AQA
Realising potential

LEVEL 3 FOUNDATION TECHNICAL LEVEL IT: SCRIPTING AND APP PROGRAMMING

360 GLH (TVQ01015)

LEVEL 3 TECHNICAL LEVEL IT: PROGRAMMING

720 GLH (TVQ01013)

Specifications
First registration September 2016 onwards

Version 5.1 November 2018

Contents

1 About these qualifications

These qualifications are Advanced (Level 3) Technical qualifications, on a par with A-levels and have been built in close collaboration with employers and professional bodies ensuring that they have both recognition and value.

They are for learners over the age of 16 who wish to specialise or progress into a specific sector or specific occupational group, through advanced/higher apprenticeships, further study or employment.

Transferable skills (sometimes known as 'soft skills') have been contextualised explicitly within the content of each qualification. These transferable skills have been prioritised by employers and professional bodies in this sector and are a mandatory part of the qualification outcome. It is important to note that learners **must** demonstrate successful achievement of the identified transferable skill(s) appropriate to the qualification on at least **one** occasion to the required standard.

The Statements of purpose (pages 12 and 19) give more detail on the likely progression for learners with these qualifications.

Each qualification is one of the three components of the new Technical Baccalaureate (TechBacc).

The TechBacc is a performance table measure which recognises the highest level of technical training. It recognises the achievement of learners taking a Technical Level qualification, a Level 3 maths qualification and an Extended Project Qualification (EPQ).

2 Qualification at a glance – overview

2.1 Level 3 Foundation Technical Level IT: Scripting and App Programming

Ofqual qualification number	601/7125/X	**AQA qualification number**	TVQ01015
First registration date	1 September 2016	**Age range**	16–18, 19+
Last registration date	31 August 2020	**UCAS points**	Information on UCAS points can be obtained from ucas.com
Last certification date	31 August 2023	**Performance table points**	Information on performance measures can be found at education.gov.uk
Total qualification time (TQT)	380 (GLH = 360) (See TQT section for more information)	**Eligibility for funding**	Yes
Unit weighting **Externally assessed** **Internally assessed**	25% each unit (2 x units) 25% each unit (2 x units)	**Entry requirements**	There are no formal entry requirements for this qualification set by AQA.

Mandatory units	All units in this qualification are mandatory.
Resits, resubmissions and retakes	The learner is permitted one resit/retake in relation to each unit of the qualification. Where a unit is examined/externally assessed, this means one resit. Where a unit is internally assessed and externally quality assured, this means one retake. Resits, resubmissions and retakes are each permitted where learners have both failed the requirements of the unit and where the learner wishes to improve on a grade received. Any resubmission of an assignment (ie a second attempt at an internally assessed unit task/assignment prior to external quality assurance) must be undertaken without further guidance from the tutor and must be completed within a defined and reasonable period of time following the learner receiving their initial result of the assessment.

Assessment model	This qualification contains externally examined and internally assessed units. Internally assessed units are externally quality assured by AQA.	**Examination sessions**	January and June each year.
Employer involvement during delivery	It is a requirement that employers are engaged meaningfully in the delivery of this qualification. Further information on this can be found in the individual units (where relevant) and the Meaningful employer involvement section.		

Grading	The units are graded Pass, Merit or Distinction The overall qualification is graded as P, M, D, D*

Transferable skills contextualised within the units of this qualification

These are the skills deemed essential by the employers and professional bodies AQA has collaborated with on the development of this qualification. We have contextualised units around these 'soft' skills. There may be more than one opportunity for each transferable skill to be evidenced to the required standard across the units within the qualification. It is important to note that learners **must** demonstrate successful achievement of the identified transferable skill(s) appropriate to the qualification on **one** occasion to the required standard in the identified unit(s). Evidence produced for the transferable skills will be internally assessed and externally quality assured.

- Research

2.2 Level 3 Technical Level IT: Programming

Ofqual qualification number	601/7129/7	**AQA qualification number**	TVQ01013
First registration date	1 September 2016	**Age range**	16–18, 19+
Last registration date	31 August 2020	**UCAS points**	Information on UCAS points can be obtained from ucas.com
Last certification date	31 August 2023	**Performance table points**	Information on performance measures can be found at education.gov.uk
Total qualification time (TQT)	760 (GLH = 720) (See TQT section for more information)	**Eligibility for funding**	Yes
Unit weighting **Externally assessed** **Internally assessed**	12.5% each unit (3 x units) 12.5% each unit (5 x units)	**Entry requirements**	There are no formal entry requirements for this qualification set by AQA.

Mandatory units	All units in this qualification are mandatory.
Resits, resubmissions and retakes	The learner is permitted one resit/retake in relation to each unit of the qualification. Where a unit is examined/externally assessed, this means one resit. Where a unit is internally assessed and externally quality assured, this means one retake. Resits, resubmissions and retakes are each permitted where learners have both failed the requirements of the unit and where the learner wishes to improve on a grade received. Any resubmission of an assignment (ie a second attempt at an internally assessed unit task/assignment prior to external quality assurance) must be undertaken without further guidance from the tutor and must be completed within a defined and reasonable period of time following the learner receiving their initial result of the assessment.

Assessment model	This qualification contains externally examined and internally assessed units. Internally assessed units are externally quality assured by AQA.	**Examination sessions**	January and June each year.
Employer involvement during delivery	It is a requirement that employers are engaged meaningfully in the delivery of this qualification. Further information on this can be found in the individual units (where relevant) and the Meaningful employer involvement section.		

Grading	The units are graded Pass, Merit or Distinction The overall qualification is graded as PP, MP, MM, DM, DD, D*D, D*D*

Transferable skills contextualised within the units of this qualification

These are the skills deemed essential by the employers and professional bodies AQA has collaborated with on the development of this qualification. We have contextualised units around these 'soft' skills. There may be more than one opportunity for each transferable skill to be evidenced to the required standard across the units within the qualification. It is important to note that learners **must** demonstrate successful achievement of the identified transferable skill(s) appropriate to the qualification on **one** occasion to the required standard in the identified unit(s). Evidence produced for the transferable skills will be internally assessed and externally quality assured.

- Research
- Problem-solving
- Teamwork
- Communication (oral and written)

3 Level 3 Foundation Technical Level IT: Scripting and App Programming: Statement of purpose

3.1 Qualification objectives

The objectives of this qualification are:

- preparing learners to progress to a qualification in the same subject area but at a higher level or requiring more specific knowledge, skills and understanding
- meeting relevant programmes of learning
- preparing learners for employment
- giving learners personal growth and engagement in learning.

This qualification is linked to the following Standard Occupational Classification (SOC)[1] to prepare learners for work in this area:

AQA Level 3 Foundation Technical Level IT: Scripting and App Programming

- 213 – information technology and telecommunications professionals

3.2 Who is this qualification for?

This technical qualification is aimed at 16 to 18 year old learners who are seeking to develop skills and access a range of junior scripting and app development job roles in a variety of sector settings, or as the first year of a two year programme where learners aspire to achieve the IT: Programming qualification.

It provides a progression pathway from a range of Level 2 qualifications and learning programmes as can be seen in the following document: gov.uk/government/publications/technical-and-vocational-qualifications-for-14-to-19-year-olds

There are no formal entry requirements for this qualification but to optimise their chances of success, learners will typically have five GCSE's at A* to C, preferably including English and maths.

This qualification could be studied alongside other Level 3 qualifications such as IT: Technical Support, or the IT: Networking or IT: User Support qualifications for a multi-discipline technical role in a small or medium enterprise (SME).

It can form part of a study programme, Technical Baccalaureate and would benefit from being studied alongside an Applied General, A-level or an EPQ.

[1] SOC code is Standard Occupational Category – a common classification of jobs based on their skill content and level – assigned by The Office for National Statistics.

3.3 What does this qualification cover?

All of the units in this qualification are mandatory and will provide a core knowledge and understanding of IT: scripting and app programming. Focusing on basic programming, website development, client and server-side scripting, plus an opportunity to explore cross-platform mobile applications development, all based on underpinning units in the fundamental principles of computing and computer programming, this will prepare learners to work in this sector.

This qualification has been developed under the guidance of The Tech-Partnership and the British Computer Society (BCS) who are the sectors primary professional bodies.

The learner will cover topics such as:

- the theory, practices and concepts associated with programming solutions, designed and developed to meet client requirements
- building high-quality coded applications for popular mobile devices across platforms
- designing and building interactive websites and cloud-based applications that professionally meet client needs demonstrating using client-side and server-side technologies, eg PHP/JSP/ASP.net, JavaScript, CSS, XML and combinations such as Ajax.

Transferable skills are those generic 'soft skills' that are valued by employers and higher education alike. The following transferable skills have been contextualised into the content of the qualification:

- research.

Units which provide opportunities to achieve these skills are listed below:

Unit code	Unit title	Transferable skill(s)
M/507/6476	Website technologies	Research

Opportunities for each available transferable skill will be highlighted in the pass criteria for the unit where appropriate.

There may be more than one opportunity for each transferable skill to be evidenced to the required standard across the units within the qualification. It is important to note that learners **must** demonstrate successful achievement of the identified transferable skill(s) appropriate to the qualification on at least **one** occasion to the required standard.

The Transferable skills standards can be found at Appendix A.

3.4 What could this qualification lead to?

Learners who achieve this qualification will have a range of options.

Progression from this Level 3 Technical qualification is designed to be to work, as a junior web or app developer. Learners would have an opportunity for further study, topping up this qualification to a Level 3 Technical Level in IT: Programming qualification. This qualification will also contribute to university entry and will provide opportunities to undertake a range of professional qualifications from vendors such as CISCO.

However, as it is studied at 16 to 19 as part of the study programme, learners will be studying additional qualifications such as an A-level, an EPQ, an AS and possibly re-sits for GCSE English and/or Maths, learners will potentially be able to access higher education – either HNCs and HNDs or degree programmes.

Therefore, studying this qualification does not restrict future progression into one particular route.

The following are examples of job opportunities within this sector:

- junior web developer
- junior mobile app developer.

Companies that might employ someone with this qualification are:

- web companies
- app development companies
- some learners may choose to work for themselves.

3.5 Who supports this qualification?

This qualification has been developed in collaboration with employers, professional bodies and key stakeholders in the IT sector. Because of this, the knowledge, skills and competencies gained will provide the best possible opportunity for progression into employment, a higher or advanced apprenticeship or higher education.

This qualification is supported by the following organisations:

bcs The Chartered Institute for IT	British Computer Society (BCS)	bcs.org
TECH PARTNERSHIP	The Tech Partnership	thetechpartnership.com
UK Cyber Security Forum	UK Cyber Security Forum	ukcybersecurityforum.com
D-RISQ SOFTWARE SYSTEMS	D-RisQ	drisq.com
fasthosts Inspiring Better Business	Fasthosts	fasthosts.co.uk
TOSHIBA Leading Innovation	Toshiba UK	toshiba.co.uk
NETGEAR®	NETGEAR	netgear.co.uk

	Weheartdigital Limited	weheart.digital
	CompTIA	comptia.org
	Microsoft	microsoft.com
	AlfaPeople UK	alfapeople.com
	RSPCA	rspca.org.uk
	CCL Group Limited	cclgroupltd.com
	Cisco	cisco.com
	VMWare IT Academy	vmware.com
	Axelos	axelos.com
	City of Wolverhampton College	wolvcoll.ac.uk
	Burton and South Derbyshire College	bsdc.ac.uk

Solihull College & University Centre	Solihull College	solihull.ac.uk
SOUTH & CITY COLLEGE BIRMINGHAM	South and City College Birmingham	sccb.ac.uk
NULC Newcastle-under-Lyme College	Newcastle-under-Lyme College	nulc.ac.uk
Edge Hill University	Edge Hill University	edgehill.ac.uk
University of Bedfordshire	University of Bedfordshire	beds.ac.uk or beds.ac.uk/howtoapply/departments/teacher-education/tt
STAFFORDSHIRE UNIVERSITY	Staffordshire University	staffs.ac.uk

3.6 What are the benefits of this qualification?

To learners

Learning to program computers can be a fascinating experience because being able to move objects with a few button clicks and a few lines of code can be tremendous fun. With more and more businesses relying more heavily on technologies such as websites and mobile applications to support their activity, this qualification will be essential in teaching you how to support employers in these areas.

On this course you will build on a foundation of programming concepts, but using these in website development, with an opportunity to create cross-platform mobile apps. Today, many apps that sell on sites such as iTunes and on similar sites selling android versions, have been built by young programmers in their spare time.

For this reason, this qualification will give you a set of skills that can be used in a variety of contexts even if you decide not to go on to be an industry programmer.

You will be working on industry focused assignments to help you to prepare for work, which will help you to talk confidently about your knowledge and skills in an interview situation.

When you have successfully completed the course you will be able to look for a job, or if you prefer, along with your other studies you will be able to apply to university or college to continue your studies.

Beginning with the fundamental principles of computing, you will study the different programming constructs, web and app development, developing solutions for employers or customers.

With experience you will eventually be able to work in a variety of roles.

To employers

This qualification has been developed in consultation with employers and professional bodies who have identified a range of technical and personal skills that are essential for a junior or entry role in this area.

All learners who have achieved this technical qualification will have learned and been assessed using the same content as there are no optional units in this programme and they will have demonstrated and achieved a range of transferable skills that are essential in this area such as teamwork, skills in problem-solving, research and in both written and oral communication, which are important sector skills.

Building on the fundamental principles of computing, learners will consider basic programming techniques and design techniques across a possible range of languages and they will understand how to create website solutions for industry. They will also be able to create apps for different platforms.

Employers can therefore be confident that learners have a comprehensive grounding in technical IT and personal skills.

To higher education institutions

Discussions with higher education institutions (HEIs) during the development of this programming qualification identified that learners with programming skills will be able to use the skills they have developed (such as problem-solving, logic, design, accuracy, process and checking/testing) to support almost any higher qualification.

Learners will be able to use their learning towards software design or programming degrees, or for more general IT programmes.

In this qualification learners will build on the fundamental principles of computing, learning basic programming techniques across a possible range of languages and they will understand how to create website solutions for industry. They will also be able to create apps for different platforms.

Combining both the research transferable skill and other skills for development through this programme, HEIs can therefore be confident that learners will be ready to study at higher levels.

3.7 Links to professional body memberships

The British Computer Society (BCS) believes employers will recruit and train professionals with AQA Tech Level IT qualifications for roles that are likely to be at Registered IT Technician level (see Letter of Support).

4 Level 3 Foundation Technical Level IT: Scripting and App Development: Unit summary

This qualification is made up of four mandatory units. All units must be successfully completed to achieve the full qualification.

	Unit title	Assessment type	Ofqual unit reference
1	Fundamental principles of computing	External examination	Y/507/6424
2	Computer programming	External examination	F/507/6465
3	Website technologies	Internally centre assessed	M/507/6476
4	Mobile applications programming	Internally centre assessed	Y/507/6486

Links with other qualifications

The following units are shared across the IT sector:

Y/507/6424 1 Fundamental principles of computing

The following unit is shared with IT: Programming:

F/507/6465 2 Computer programming

M/507/6476 3 Website technologies

Y/507/6486 4 Mobile applications programming

5 Level 3 Technical Level IT: Programming: Statement of purpose

5.1 Qualification objectives

The objectives of this qualification are:

- preparing learners to progress to a qualification in the same subject area but at a higher level or requiring more specific knowledge, skills and understanding
- meeting relevant programmes of learning
- preparing learners for employment
- supporting a role in the workplace
- giving learners personal growth and engagement in learning.

This qualification is linked to the following Standard Occupational Classification (SOC)[2] to prepare learners for work in this area:

AQA Level 3 Foundation Technical Level IT: Scripting and App Programming

- 2136 – programmers and software development professionals

5.2 Who is this qualification for?

This technical qualification is aimed at 16 to 18 year old learners who are seeking to develop programming skills to enable them to access a range of roles in a variety of sector settings, or a software development apprenticeship.

It provides a progression pathway from a range of Level 2 qualifications and learning programmes as can be seen in the following document: gov.uk/government/publications/technical-and-vocational-qualifications-for-14-to-19-year-olds

There are no formal entry requirements for this qualification but to optimise their chances of success, learners will typically have five GCSE's at A* to C, preferably including English and maths.

This qualification could be studied alongside other Level 3 qualifications such as business or enterprise for those who seek a career in programming or wish to become self-employed. Alternatively learners could study the IT: Technical Support route to broaden their IT and computing experience.

It can form part of a study programme, Technical Baccalaureate and would benefit from being studied alongside an Applied General, A-level or an EPQ.

[2] SOC code is Standard Occupational Category – a common classification of jobs based on their skill content and level – assigned by The Office for National Statistics.

5.3 What does this qualification cover?

All of the units in this qualification are mandatory and will provide a core knowledge and understanding of IT programming. This qualification focuses on the key programming paradigms of object oriented programming and event driven programming that are themselves built on a knowledge base of programming concepts, this will prepare learners to work in this sector.

This qualification has been developed under the guidance of The Tech-Partnership and the British Computer Society (BCS).

The learner will cover topics such as:

- the theory, practices and concepts associated with high quality professionally programming solutions, designed and developed to meet client requirements
- object oriented design of objects and classes using OOP design techniques and implementing the designs into working solutions that have been tested and documented to professional standards
- event driven design using EDP features and functions such as event triggers, controls, event handlers, identifiers, data structures, operators, common language constructs
- building high-quality coded applications for popular mobile devices across platforms
- mathematical concepts contextualised for computing including number systems, base conversion, logical operators and data interpretation and representation
- logic that is used every day by operating systems, networks and programmers alike
- designing and building interactive websites and cloud-based applications that professionally meet client needs demonstrating using client-side and server-side technologies, eg PHP/JSP/ASP.net, JavaScript, CSS, XML and combinations such as Ajax.

Transferable skills are those generic 'soft skills' that are valued by employers and higher education alike. The following transferable skills have been contextualised into the content of the qualification:

- communication (oral and written)
- research
- teamwork
- problem-solving.

Units which provide opportunities to achieve these skills are listed below:

Unit code	Unit title	Transferable skill(s)
A/507/6464	Industrial project	Teamwork
D/507/6487	Event driven programming	Communication (oral)
K/507/6489	Object oriented programming	Problem-solving, communication (written)
M/507/6476	Website technologies	Research

Opportunities for each available transferable skill will be highlighted in the pass criteria for the unit where appropriate.

There may be more than one opportunity for each transferable skill to be evidenced to the required standard across the units within the qualification. It is important to note that learners **must** demonstrate successful achievement of the identified transferable skill(s) appropriate to the qualification on at least **one** occasion to the required standard.

The Transferable skills standards can be found at Appendix A.

5.4 What could this qualification lead to?

Learners who achieve this qualification will have a range of options.

Progression from this Level 3 Technical qualification is designed to be into work, as a junior programmer or developer in a variety of programming settings including engineering, games and app development. Learners would have an opportunity for further study either through university or through professional qualifications from a number of vendors.

Furthermore, this technical qualification which sits under SOC 2136 will be exceptionally well placed to address the skills shortage for programmers identified in the Tier 2 Shortage Occupations List.

However, as it is studied at 16 to 19 as part of the study programme, learners will be studying additional qualifications such as an A-level, an EPQ, an AS and possibly re-sits for GCSE English and/or Maths, learners will potentially be able to access higher education – either HNCs and HNDs or degree programmes.

Therefore, studying this qualification does not restrict future progression into one particular route.

The following are examples of job opportunities within this sector:

- junior programmer
- analyst-programmer
- games programmer
- software engineer.

Companies that might employ someone with this qualification are:

- web companies
- app development companies
- software houses
- games development companies
- some learners may choose to work for themselves.

5.5 Who supports this qualification?

This qualification has been developed in collaboration with employers, professional bodies and key stakeholders in the IT sector. Because of this, the knowledge, skills and competencies gained will provide the best possible opportunity for progression into employment, a higher or advanced apprenticeship or higher education.

This qualification is supported by the following organisations:

bcs The Chartered Institute for IT	British Computer Society (BCS)	bcs.org
TECH PARTNERSHIP	The Tech Partnership	thetechpartnership.com

	UK Cyber Security Forum	ukcybersecurityforum.com
	D-RisQ	drisq.com
	Fasthosts	fasthosts.co.uk
	Toshiba UK	toshiba.co.uk
	NETGEAR	netgear.co.uk
	Weheartdigital Limited	weheart.digital
	CompTIA	comptia.org
	Microsoft	microsoft.com
	AlfaPeople UK	alfapeople.com
	RSPCA	rspca.org.uk
	CCL Group Limited	cclgroupltd.com
	Cisco	cisco.com

vmware IT ACADEMY	VMWare IT Academy	vmware.com
AXELOS GLOBAL BEST PRACTICE	Axelos	axelos.com
city of wolverhampton college Learn • Achieve • Succeed	City of Wolverhampton College	wolvcoll.ac.uk
Burton and South Derbyshire College	Burton and South Derbyshire College	bsdc.ac.uk
Solihull College & University Centre	Solihull College	solihull.ac.uk
SOUTH & CITY COLLEGE BIRMINGHAM	South and City College Birmingham	sccb.ac.uk
NULC Newcastle-under-Lyme College	Newcastle-under-Lyme College	nulc.ac.uk
Edge Hill University	Edge Hill University	edgehill.ac.uk
University of Bedfordshire	University of Bedfordshire	beds.ac.uk or beds.ac.uk/howtoapply/departments/teacher-education/tt
STAFFORDSHIRE UNIVERSITY	Staffordshire University	staffs.ac.uk

5.6 What are the benefits of this qualification?

To learners

Learning to program computers can be a fascinating experience because being able to move objects with a few button clicks and a few lines of code can be tremendous fun. It has its serious side too though, programming network and security protocols, creating websites to support businesses, mining data for analysis to give just a few examples.

There is one thing that the industry agree on, however, and that is that a sound understanding of the main programming paradigms is essential for anyone wanting to work as a systems programmer, games programmer, or analyst, app developer.

On this course you will build on a foundation of programming concepts and study object oriented programming (known as OOP) and event driven programming (EDP). Once learned, you will be able to transfer the techniques to different languages. You will also learn about website programming and scripting. The Maths unit has been included to enable you to use Maths in the context, for example manipulating objects.

You will be working on industry focused assignments to help you to prepare for work, which will help you to talk confidently about your knowledge and skills in an interview situation.

When you have successfully completed the course you will be able to look for a job, or if you prefer, along with your other studies you will be able to apply to university or college to continue your studies.

Beginning with the fundamental principles of computing, you will study the different programming constructs, you will learn about testing, web development, commercial applications, apps and developing solutions for employers or customers.

With experience you will eventually be able to work in a variety of roles.

To employers

This qualification has been developed in consultation with employers and professional bodies who have identified a range of technical and personal skills that are essential for work in this area.

All learners who have achieved this technical qualification will have learned and been assessed using the same content as there are no optional units in this programme and they will have demonstrated and achieved a range of transferable skills that are essential in this area such as teamwork, skills in problem-solving and in both written and oral communication.

Building on the fundamental principles of computing, learners will have studied a variety of programming languages in a range of contexts and will understand how to create solutions for industry. They should be versed in testing and will be able to create apps for different platforms.

All learners will have been involved in a contextualised project that will have been industry driven and will demonstrate the ability to contribute to team performance.

Employers can therefore be confident that learners have a comprehensive grounding in technical and personal skills.

To higher education institutions

Discussions with higher education institutions (HEIs) during the development of this programming qualification identified that to succeed in higher study in this area, learners would need to demonstrate competence in mathematics. Clearly this could have been studied as an additional A-level or AS level, but there are aspects of this subject that are not necessarily relevant in the sector. For this reason a mathematics unit has been included as a mandatory part of this qualification providing opportunities for contextualised maths.

In addition, learners will have built on the fundamental principles of computing, learners will have studied both object oriented and event driven programming, they will have developed apps and will be confident in testing.

All learners will have been involved in a contextualised project that will have been industry driven and that will demonstrate the ability to contribute to team performance.

HEIs can therefore be confident that learners will be ready to study at higher levels.

5.7 Links to trailblazers

This qualification has been developed to provide a comprehensive grounding in programming with a view to offer learners the opportunity to progress to a Level 4 Trailblazer Apprenticeship as a Software Developer: apprenticeships.org.uk/~/media/Apprenticeship-standards/DI%20-%20Software.ashx

There is also an opportunity to progress to the new Software Tester Level 4 Trailblazer Apprenticeship: gov.uk/government/publications/apprenticeship-standard-software-tester

5.8 Links to professional body memberships

The British Computer Society (BCS) believes employers will recruit and train professionals with AQA Tech Level IT qualifications for roles that are likely to be at Registered IT Technician level (see Letter of Support).

6 Level 3 Technical Level IT: Programming: Unit summary

This qualification is made up of eight mandatory units. All units must be successfully completed to achieve the full qualification.

	Unit title	Assessment type	Ofqual unit reference
1	Fundamental principles of computing	External examination	Y/507/6424
2	Computer programming	External examination	F/507/6465
3	Website technologies	Internally centre assessed	M/507/6476
4	Mobile applications programming	Internally centre assessed	Y/507/6486
5	Mathematics for programmers	External examination	Y/507/6469
6	Event driven programming	Internally centre assessed	D/507/6487
7	Object oriented programming	Internally centre assessed	K/507/6489
8	Industrial project	Internally centre assessed	A/507/6464

Links with other qualifications

The following units are shared across the IT sector:

Y/507/6424 1 Fundamental principles of computing

These also appear within:

AQA Level 3 Technical Level IT: User Support

AQA Level 3 Technical Level IT: Networking

The following units are shared with IT: Networking and IT: User Support:

A/507/6464 8 Industrial project

7 Meaningful employer involvement

7.1 Introduction

It is important that centres develop an approach to teaching and learning that supports the assessment of the technical focus of a Tech-level qualification. The specification contains a balance of practical skills and knowledge requirements and centres need to ensure that appropriate links are made between theory and practice in a way that is relevant to the occupational sector.

This will require the development of relevant and up-to-date training materials that allow learners to apply their learning to actual events and activity within the sector, and to generate appropriate evidence for their portfolio.

It is a requirement that employers are involved in the delivery and/or assessment of the Tech-level to provide a clear 'line of sight' to work, advanced/higher apprenticeships or higher education. Employer engagement enriches learning, raises the credibility of the qualification in the eyes of employers, parents and learners – as well as also furthering the critical collaboration between the learning and skills sector and industry.

It is therefore a requirement that all learners undertake meaningful activity involving employers during their study and this activity will be scrutinised as part of our ongoing quality assurance activities with centres.

Such is the importance of meaningful employer involvement in the delivery of this qualification, should a centre be unable to evidence this, we will impose a sanction, together with an associated action plan. Further information on this process can be found in the AQA *Centre Administration Guide for Technical and Vocational Qualifications*.

AQA will not stipulate the minimum duration or contribution of employer involvement to the overall qualification grade as it is important that centres and employers are allowed flexibility in how best to work together to support learning and in which units – but this collaboration must be significant.

7.2 Definition of meaningful employer involvement

In order to meet our requirements, meaningful employer involvement must take at least one of the following forms:

- learners undertake structured work experience or work placements that develop skills and knowledge relevant to this qualification
- learners undertake project work, exercises and/or assessments set with input from industry practitioners – such as getting employers involved in developing real life case studies, or assignments
- learners take one or more units delivered or co-delivered by an industry practitioner – this could be in the form of masterclasses or guest lectures
- industry practitioners operating as 'expert witnesses' that contribute to the assessment of a learner's work or practice, operating within a specified assessment framework. This may be for specific project work, exercises or examinations, or all assessments for a qualification.

For the purpose of clarity, the following activities, whilst valuable, would **not** be considered as meaningful employer involvement:

- employers hosting visits, providing premises, facilities or equipment
- employers or industry practitioners providing talks or contributing to delivery on employability, general careers advice, CV writing, interview training
- learner attendance at career fairs, events or other networking opportunities
- simulated or centre-based working environments
- employers providing learners with job references.

More information on employer involvement in the delivery of technical level qualifications can be found at:

- gov.uk/government/uploads/system/uploads/attachment_data/file/306280/RR341_-_Employer_Involvement_in_Qualifications_Delivery_and_Assessment_Research_Report.pdf
- gov.uk/government/uploads/system/uploads/attachment_data/file/268624/document.pdf

7.3 Employer involvement in quality assurance

We need to make sure that the assessment remains relevant and valid, and that learning outcomes are what employers and higher education institutions are expecting of a learner who has achieved a Level 3 Tech-level qualification.

Each year a panel, including representatives from employers and HE, will be brought together to review outcomes from the units and we will ask for samples of learner work from your centre at each EQA visit.

We are keen to work collaboratively with employers and HE to make sure that whatever the progression route chosen by the learner, this qualification will be recognised and valued.

If you have a local employer that would like to be involved in this review, we would be very pleased to consider them. Please email their contact details to techlevels@aqa.org.uk

8 Synoptic delivery and assessment

The definition of synoptic assessment used by AQA is:

'A form of assessment which requires a learner to demonstrate that they can identify and use effectively, in an integrated way, an appropriate selection of skills, techniques, concepts, theories, and knowledge from across the whole qualification or unit, which are relevant to a key task'.

The design of this qualification allows learners to develop knowledge, understanding and skills from some units and then evidence this learning in the performance outcomes contained within other units.

The significant amount of synoptic content within the Tech-level supports synoptic learning and assessment by:

- showing teaching and learning links between the units across the specification
- giving guidance or amplification relating to the grading criteria for the internally assessed units, about where learners could apply the knowledge and understanding from other units
- providing a coherent learning programme of related units
- allowing holistic delivery and the application of prior or concurrent learning
- providing opportunities for the learning and assessment of multiple units combined together to promote holistic delivery
- developing and assessing learners' use of transferable skills in different contexts.

Whilst we do not prescribe in which order the units should be delivered or assessed, it is important for centres to be aware of the links between the units so that the teaching, learning and assessment can be planned accordingly. This way, when being assessed, learners can apply their learning in ways which show they are able to make connections across the qualification.

It is therefore a requirement that all learners undertake meaningful synoptic learning and assessment during their study. Plans for how this will be undertaken will be scrutinised as part of our centre approval process and its implementation monitored during our ongoing quality assurance activities with centres.

Within each unit we provide references to where the unit content maps from or to other units within the qualification. This will help the learner understand where there are explicit opportunities for synoptic learning as well as synoptic assessment.

For example, learners will be able to see very clearly how they can apply the underpinning knowledge and theory from the core units into real life or work related tasks – such as projects and work experience – within the specialist units.

This approach will also enable learners to integrate transferable skills much valued by employers and HE into their assignments.

The following grid demonstrates the overall synoptic coverage in each unit of the qualification:

Unit	Assessment outcomes/ pass criteria	Synoptic links to other units	% of synoptic assessment
Fundamental principles of computing	5	Underpinning knowledge for sector	5/5 (100%)
Computer programming	4	Underpinning knowledge for pathway	4/4 (100%)
Website technologies	15	Linked to Units 1, 2, 4, 5, 6, and 7	11/15 (73%)
Mobile applications programming	15	Linked to Units 1, 2, 3, 6 and 7	14/15 (93%)
Mathematics for programmers	5	Linked to Units 1, 2, 3, 6, 7 and 8	5/5 (100%)
Event driven programming	11	Linked to Units 1, 2, 3, 4, 7 and 8	11/11 (100%)
Object oriented programming	15	Linked to Units 1, 2, 3, 4, 5, 6 and 8	14/15 (93%)
Industrial project	All units contribute to the industrial project where learners will work together to create solutions for users or a client.		

This qualification contains 82.4% synopticity calculated over all eight units, or 94% synopticity calculated over seven units (excluding industrial project).

9 Total qualification time

For any qualification which it makes available, Ofqual requires an awarding organisation to:

a assign a number of hours for total qualification time to that qualification, and

b assign a number of hours for guided learning to that qualification.

Total qualification time is the number of notional hours which represents an estimate of the total amount of time that could reasonably be expected to be required in order for a learner to achieve and demonstrate the achievement of the level of attainment necessary for the award of a qualification.

Total qualification time is comprised of the following two elements:

a the number of hours which an awarding organisation has assigned to a qualification for guided learning (GLH)

AQA has assigned GLH to the overall qualification and the individual units.

b an estimate of the number of hours a learner will reasonably be likely to spend in preparation, study or any other form of participation in education or training, including assessment, which takes place as directed by – but, unlike guided learning, not under the immediate guidance or supervision of – a lecturer, supervisor, tutor or other appropriate provider of education or training.

AQA has assigned the following GLH and TQT values to its qualifications:

Qualification	Guided learning hours (GLH)	Total qualification time (TQT)
IT: Scripting and App Programming (TVQ01015)	360	380
IT: Programming (TVQ01013)	720	760

10 Transferable skills

These valued 'employability' skills are an integral and explicit element within the design and structure of all AQA Level 3 Technical Level qualifications.

Discussions and collaboration with centres, employers and stakeholders (such as FE Colleges, UTCs, sector skills councils, professional/trade bodies and HE), made it clear that the inclusion of these skills is regarded as a priority, and that they should be included through contextualisation within the core subject content.

Employers and stakeholders prioritised the skills they required from employees in the sector as follows:

- teamworking
- communication
- problem-solving
- research.

Rather than force the inclusion of these skills across a random selection of units or across the qualification as a whole, specific units have been identified as being most appropriate and suitable for the inclusion of a transferable skill within the subject context. The skill becomes the driver for the assessment – rather than the subject content and this will be demonstrated by producing evidence to meet the required standard[3]. Not every unit within the qualification has a skill contextualised within the subject content.

Unit code	Pathway	Unit title	Transferable skill(s)
A/507/6464	Programming	Industrial project	Teamwork
D/507/6487	Programming	Event driven programming	Communication (oral)
K/507/6489	Programming	Object oriented programming	Problem-solving Communication (written)
M/507/6476	Scripting and App Programming Programming	Website technologies	Research

The skill is assessed as a performance outcome of the unit, at the Pass grade. It is assessed in the same way as any other assessment criteria within the unit.

The formal inclusion of a contextualised transferable skill does not preclude the inclusion of other 'soft' or 'employability' skills within the unit at the point of delivery, for example those which employers and HE will also value, such as critical thinking, project management, leadership, time management etc. However, these additional 'employability' skills will not be formally assessed as part of the unit performance outcomes.

[3] Please visit the specification homepage to access the transferable skills standards and associated guidance and recording documentation.

The AQA Skills statement

Upon the successful completion of a qualification, each learner will be issued with a Skills statement that will sit alongside their formal qualification certificate.

This Skills statement records the transferable skills that were contextualised within the units of the qualification and is an explicit way for learners to showcase the skills that have been formally assessed as part of the qualification. This Skills statement can then be used by a learner as evidence of this achievement within their CVs or HE applications.

11 Support materials and guidance

The following delivery resources and support materials are available from AQA.

- A full Scheme of work (SOW) has been provided for each of the units in this programme. The SOW breaks down the unit content across a 30 teaching week academic year, although this is not mandated. Suggestions have been made for activities both for the tutor and the learner, and the document also contains links to external resources such as videos, task sheets, pdfs, PowerPoint presentations etc. Opportunities to develop English and maths skills have been identified and flagged, and SOWs include some mapping for stretch and challenge and equality and diversity, although tutors will benefit from making this much more class relevant. Assignment handouts have been identified and the assessment points for transferable skills have been highlighted in the final column.
- A sample Lesson plan has also been provided against the SOW, as a guide for good practice.
- A sample assignment has been provided for each of the internally assessed units. These are not mandated in the assessment of this qualification, but do provide a good starting point to help tutors who would benefit from assessment support. These assessments do not necessarily cover all of the criteria that need to be assessed within a unit and it is the tutor's responsibility to ensure that all criteria are assessed across the unit and qualification.
- Sample question papers and mark schemes for each of the examined units.

The schemes of work and lesson plans are available at: aqa.org.uk/subjects/ict-and-computer-science/tech-level/itprogramming/teaching-resources

The sample assessment materials (question papers and mark schemes), plus the available sample assignments can be found at: aqa.org.uk/subjects/ict-and-computer-science/tech-level/it-programming/assessment-resources

12 Qualification units

12.1 Unit 1: Fundamental principles of computing

Title	Fundamental principles of computing
Unit number	Y/507/6424
Assessment	Externally assessed
Guided learning hours	90
Transferable skill(s) contextualised within this unit	N/A
Resources required for this unit	Central processing unit (CPU); memory chips; motherboards; internal disk drives; expansion cards; computer cases; cables; power supplies; cooling devices; ports; external devices; operating systems; device drivers; applications software.
Synoptic assessment within this unit	This unit provides the underpinning knowledge for all units contained in either the IT: Scripting and App Programming or the IT: Programming pathway.

Aim and purpose

This unit will provide the learner with the necessary knowledge to understand the different hardware and elements of a computer system and how these contribute to a fully functioning computer system. The learner will also develop a range of skills required to make changes to computer systems to ensure that they are fit for the particular requirements of the users.

Unit introduction

The fundamental requirement of any information system is a responsive computer system. Anyone who works in the IT industry needs to be fully conversant with the hardware and software elements that work together to meet the needs of the user.

This unit will provide the learner with understanding of the fundamental building blocks of such systems and enable them to understand how the various components can be linked together and why different possible combinations of these elements can affect the potential of the system to perform to the required standard.

The ability to test or upgrade a system to identify any problems and ensure that it continues to provide the required level of performance is a fundamental skill required of a computer technician. This unit will also provide the opportunity for the learner to develop the relevant skills to enable them to carry out a range of basic tests and make the necessary adjustments to the system for a given scenario.

While small systems may be managed and maintained by a single individual, larger systems require teams of specialists to take responsibility for one or more components in areas such as security. Whether large or small, it is necessary for all testing and adjustments to the systems to be recorded and reported to the responsible manager. As a result, this unit will provide learners with opportunities to develop their skills in teamworking, written and oral communication and problem-solving.

Computer systems use electricity and some components are heavy and/or difficult to handle, so the learner will be required to understand the correct methods for working safely with electrical equipment, and avoiding damage to components through static electricity. All computer systems use data, some (if not all) of which will be sensitive and the learners will need to understand their responsibilities in protecting the data and systems. Therefore, the learner will need to identify and apply the relevant laws and regulations governing working with electrical systems. It is essential that learners consider the safe disposal of equipment, manual lifting, data protection and computer misuse, and that they carry out risk assessments before undertaking any activities such as dismantling computers, moving computers, etc.

Unit content

Different types of computer	
Personal computers	• Micro-computers, tablets. • Single user. • Applications for personal use eg email, diary, spreadsheets, databases, word processors, web access.
Multi-user computers	• Mainframes. • Supercomputers. • Multi-user. • Applications for governments and research eg storing and manipulating large volumes of data for online bookings and enquiries, payroll, weather prediction, simulators.

Hardware components of a computer system	
The internal components of a computer CPU	• Arithmetic logic unit (ALU). • Main memory. • Cache. • Control unit. • Registers; accumulator etc. • The steps of the Fetch-Execute Cycle. • The effect of an interrupt on a Fetch-Execute Cycle. • Pipelines. • Multi-core processors.

Hardware components of a computer system	

The internal components of a computer
 	
Non-CPU components
	• Power supply units (PSUs) which convert power from alternating current (AC) to direct current (DC).
	• Cooling devices:
	  • fans
	  • heat sinks and thermal paste
	  • water-based.
	• Internal hard drives.
	• Memory chips:
	  • Random Access Memory (RAM) eg Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM)
	  • Read Only Memory (ROM)
	  • Programmable Read Only Memory (PROM)
	  • Erasable Programmable Read Only Memory (EPROM)
	  • Electrical Erasable Programmable Read Only Memory (EEPROM).
	• Basic Input Output System (BIOS) and Extensible Firmware Interface (EFI):
	  • independent of operating system
	  • instructions eg booting, identification of devices, cpu, memory, power-on self-test (post).
	• Cards or expansion cards such as sound, graphics, network cards etc.
	• Input/output controllers.

Communication methods	• Computer ports such as Universal Serial Bus (USB), FireWire, Serial Advanced Technology Attachment (SATA), parallel. • Internal and external computer buses eg systems bus, data bus, memory bus, parallel bus, serial bus.
External hardware	• Input devices eg mouse, scanner, keyboard, touch screen, web cam, microphone, barcode reader, sensors. • Biometric readers eg fingerprint, iris. • External output devices eg printers (2D, 3D), screens, speakers, slide projectors. • Secondary/backing storage eg hard disk drives, USB drives, read/writeable DVDs, removable magnet disks, fixed magnetic disks, solid state drives. • Specialist operator console.

Software requirements of a computer system	
Types of software	• Systems software. • Applications software. • Shareware. • Freeware. • Open source.

Software requirements of a computer system	

System software	• Libraries eg routines which are used by multiple programs.
	• Utility programs – such as systems backup, systems optimisers, disk formatters, disk defragging, text editor, graphic editor etc.
	• Systems management software notifying actual or impending failures, capacity issues and other systems and network events eg monitoring, controlling and reporting on status of:
	  • client devices (PC, laptop, other mobile devices)
	  • printers
	  • storage.

Operating systems	• A range of operating systems eg Microsoft Windows, Apple Mac OSX, Android, Linux, Unix. • Types of operating systems eg single user, multi-user, multiprocessing, multitasking (co-operative and pre-emptive), multi-threading operating systems. • Operating system functions eg input recognition, output device recognition, tracking files, tracking directories, managing peripheral devices, sharing resources between users, ensuring that users do not interfere with each other, managing security, access to devices, programs and data.
Device drivers	• Types of device driver. • Role eg linking devices to computer system, translating commands received from operating system. • Devices requiring drivers eg expansion cards (eg network, sound, video card), printers, monitors, scanners, mobile devices.

Applications software	• Types of application software:
	  • off-the-shelf: generic programs which provide a recognised business or personal need eg word processors, databases, computer games, spreadsheets, email, internet software.
	  • bespoke designed for specific client needs.
	  • tailored – off-the-shelf adjusted for specific client needs.

Security software	• Firewalls. • Antivirus. • Anti-spyware. • Authorisation. • Authentication. • Biometrics. • Encryption.
Software inventory	• Software name. • Software version. • Date. • Activity eg installation, test, update. • Outcomes eg successful, failure, reasons for failure (if appropriate), remedial steps taken. • Report of any other observations.

How data is converted to information	
Data	• Elements which can be processed to produce useful information eg numbers (numeric), characters (alphanumeric), images, signals. • Qualitative and quantitative data.

Information	• Organised data which delivers knowledge, clarification or proof eg reports, charts, graphs, telephone directories, text books.
	• Information characteristics:
	  • accuracy
	  • validity
	  • timeliness
	  • authority
	  • objectivity.

Data processing cycle	• Input data eg words, numbers, images, signals. • Arithmetic operations eg +, -. *, /. • Logical eg ‘and’, ‘or’, ‘not’, ‘nand’, ‘xor’. • Output information eg printed report, on-screen email, correction or operating signals to machinery.

How computers process user requirements	

Data storage units	• Bits, nibbles, bytes and words.
	• Common multiples, eg:
	  • kilobyte
	  • megabyte
	  • gigabyte
	  • terabyte
	  • petabyte.
	• International system of quantities ‘Kibibyte’, etc and conflict with inaccurate international system of units (SI) definition of ‘Kilo’, etc.

Character encoding	• Character encoding eg American Standard Code for Information Interchange (ASCII), Extended ASCII, Unicode.

Programming languages	• Natural languages, eg English, French etc.
	• Man readable vs computer readable languages (ie binary).
	• Low level languages:
	  • machine code
	  • assembly language.
	• High level languages which use commands and comments as well as characters which are easier for humans to understand eg JavaScript, C++, VB.net, Ada, Fortran, Delphi, PHP, Python.
	• Fourth Generation languages (4GL) clear human commands eg Structured Query Language (SQL), OpenEdge Advanced Business Language, PROLOG.

Converting source code to machine code	• Assemblers, including cross-assemblers.
	• Translators and their differences:
	  • interpreters
	  • compilers.

Assessment outcomes

Learners will be able to:

Assessment outcome 1: Understand the different types of computer	
a	The features of personal computers.
b	The features of applications for personal use and their uses.
c	The features of multi-user computers.
d	The features of applications for governments and research used for storing and manipulating large volumes of data.

Assessment outcome 2: Understand the hardware requirements of a computer system	
a	The internal components of a CPU including their purpose.
b	The steps of the Fetch-Execute Cycle.
c	The effect and purpose of an interrupt on a Fetch-Execute Cycle.
d	Maskable interrupts (IRQ) and non-maskable interrupts (NMI).
e	The internal components of a computer.
f	Internal and external power supply units (PSUs) which convert power from alternating current (AC) to direct current (DC).
g	Cooling devices and their purpose.
h	How internal hard drives work.
i	Types of memory chips.
j	Basic input output system (BIOS) and Extensible firmware interface (EFI) and their purpose.
k	Input/output controllers and expansion cards such as sound, graphics, network cards etc. and their purpose.
l	Computer ports and their purpose.
m	Internal and external computer buses.
n	Input devices eg mouse, scanner, keyboard, touch screen, web cam, microphone, barcode reader, sensors, biometric readers eg fingerprint, iris.
o	External output devices eg printers (2D, 3D), screens, speakers, slide projectors.
p	Secondary/backing storage eg hard disk drives, USB drives, read/writeable DVDs, removable magnetic disks, fixed magnetic disks, solid state drives (SSD).
q	The purpose of specialist operator consoles.

Assessment outcome 3: Understand the software requirements of a computer system	
a	Types of software.
b	Advantages and disadvantages of shareware, freeware and open source software.
c	The purpose of libraries eg routines which are used by multiple programs.
d	The features and purpose of utility programs.
e	The role of systems management software notifying actual or impending failures, capacity issues and other systems and network events eg monitoring, controlling and reporting on status.
f	The purpose of client devices.
g	Types of operating systems and their function.
h	The purpose of the operating systems.

Assessment outcome 3: Understand the software requirements of a computer system	
i	The purpose of access to operating systems via a command line interface (CLI).
j	Types of file storage.
k	Justify the use of different types of file storage.
l	The purpose of device drivers.
m	The features of anti-malware and their purpose.
n	Security methods and their purpose.
o	The role of the software inventory including the following records.

Assessment outcome 4: Understand how data is converted to information	
a	The terms data and information with examples.
b	Methods of conveying information.
c	What can affect the quality or validity of information.
d	Qualitative and quantitative data.
e	The input, process, output cycle.
f	Arithmetic operations +, -, *, /
g	Logical operations.
h	Truth tables using up to three logical operations.

Assessment outcome 5: Demonstrate how computers process user requirements	
a	Bits, nibbles, bytes and words.
b	Use common multiples represented by decimal numbers or powers of 10.
c	The International System of Quantities ‘Kibibyte’, etc and the International System of Units (SI) definition of ‘Kilo’ etc.
d	The features and purpose of character encoding.
e	Types of language.
f	Describe Low level languages and their purpose.
g	The features and purpose of high level languages.
h	The features and purpose of Fourth Generation languages (4GL).
i	The purpose of assemblers, including cross-assemblers.
j	The features and purpose of translators.

Assessment

This unit is assessed by an external examination set and marked by AQA. The examination takes place under controlled examination conditions and the exam date will be published at the start of each academic year.

Learners are allowed to use a non-programmable scientific calculator in the examination.

The examination consists of a written paper with two sections, A and B. Learners have to complete both sections and there are no optional questions within either section.

The examination is 2 hours duration and the total number of marks available in the examination is 80.

Section A is worth 50 marks and consists of relatively short questions based on the whole of the specification for this unit. Learners are required to answer **all** of the questions in Section A.

Section B is worth 30 marks and includes longer questions worth up to 15 marks each. The questions in Section B do not necessarily cover the whole of the specification for this unit at each assessment. Learners are required to answer **all** of the questions in Section B.

Employer engagement guidance

The organisation, its staff and learners must have access to employers and expertise. The organisation will have computer/technical staff who will understand the practical activities identified in the assessment outcomes. Local employers could be invited to discuss the skills and knowledge they require to support their IT systems, to inform the structure and specific hardware and software identified in the unit.

Employers may also be able to provide opportunities to visit IT facilities or provide placement or shadowing opportunities for assessors and/or learners to provide updating of the former and learning opportunities for the latter.

Employers could be invited to an apprenticeship forum.

The British Computer Society (BCS) and the Association of Computing Machinery are two examples of professional bodies who engage with learners.

Delivery guidance

Although, for the purposes of identifying specific assessment outcomes, areas such as hardware, software, networks etc, have been split into different elements of a computer system, it is not necessary or advisable to deliver the unit in this way.

Hardware and software could be taught together for example:

The CPU, for example, only understands '0's and '1's because at this level all instructions merely change the status of switches 'off' and 'on'. Early programmers and some of those at the forefront of microchip technology still need to understand how to change the status to produce specific results and thus they need to understand binary arithmetic and machine code. Others, work at the next level of instructions where individual codes have been assembled into simple human instructions eg 'load', 'execute', where the assembler then breaks the instructions down into machine code for the computer to understand. From here the learner can go on to consider need for devices such as input and output devices and the role of the operating system and device drivers in enabling the CPU to carryout instructions. Once these are in place then the introduction of applications software becomes a requirement for those individuals who wish to use the capabilities of the computer rather than program each instruction for themselves. Practical skills can be incorporated by the learners discussing the need for devices such as graphics cards, printers, iris scanners and then identifying and installing the appropriate range of hardware; selecting the correct drivers and testing the installation. Finally, the applications software could be selected to operate the hardware and installed and tested in its turn. This could all take place as part of a scenario for designing a new computer system.

Assessment outcome 1

It is important that learners understand that computers are not limited to laptops or tablets but that they could be faced with larger and more complex machines.

The learner should have the opportunity to **research** the different computer types and **identify** the appropriate uses for each type.

This could include small group research and presentations of findings to the larger group, visits to local organisations with larger computer systems or presentations by individuals who work with the different computer types.

As a result the learners should be able to **explain** the advantages and disadvantages of a particular computer type in a given situation ie research, data warehousing, data mining, administration.

Assessment outcome 2

These are the physical components of the computer system and can be interpreted as anything that can be touched or felt. There have been many changes to computer hardware over time and new hardware and hardware modifications are appearing all of the time. Therefore, specific examples are intended only for guidance and should be adjusted to reflect the range of hardware available at the time of delivery and assessment of the unit. Learners should be able to **explore** the most up-to-date hardware available and **analyse** their strengths and weakness in order to **illustrate** their choice of hardware in a given scenario.

Where possible, learners should have the opportunity to identify components from computers using different CPU chips and operating systems and have access to either server-based systems or those who maintain such systems eg the organisation's network and server technicians, who can explain how the hardware supports organisational requirements. It would be advantageous for learners to have physical access through opportunities to look at the inside of a computer. It is not necessary for learners to actually dismantle the computer themselves, but seeing the hardware in place does assist understanding of how buses, for example, link the various items together and what the components look like when properly installed.

Some components are separate identifiable parts, such as the motherboard or ports, whereas others such as the central processing unit (CPU), which in the case of personal computers (PCs) and servers, for example, normally contain the arithmetic logic unit (ALU), the control unit (CU) and small, fast registers of read/write memory within a single unit or 'chip'. Learners should engage in class discussions or small group research activities to identify the individual components and their role in carrying out instructions and requests.

As the various parts are identified, a class or group discussion could take place as to the purpose, communication methods and location of each one. From this individuals or groups could create annotated diagrams of their findings, which could be discussed within the larger group or class.

For example, the learner would clearly benefit from being able to see and discuss as many of the following example devices as possible:

- input devices eg mouse, scanner, keyboard, touch screen, web cam, microphone, barcode reader, sensors
- external output devices eg printers (2D, 3D), screens, speakers, slide projectors
- secondary/backing storage eg hard disk drives, USB drives, read/writeable DVDs, removable magnetic disks, fixed magnetic disks, solid state drives (SSD).

Assessment outcome 3

These are the components of the computer system and can be interpreted as anything that cannot be touched, but ensures that the computer carries out the required task quickly and accurately. There have been many changes to computer software over time and new software is always becoming available. Therefore, specific examples are intended only for guidance and should be adjusted to reflect the range of software available at the time of delivery and assessment of the unit.

Learners could be provided with a computer system for which they would need to identify the appropriate operating systems, drivers, applications software etc. They could carry this out as a group or individually feeding back to the group through presentation, video or report.

The learners could carry out a theoretical activity where they would identify the necessary software for a written computer specification.

Presentations or sessions led by software or systems technicians would be beneficial to learners as the presenters could relate the software choices to actual events and possible issues which have arisen in the work place. This would assist learners in understanding the importance of software selection in the real world.

Learners should be able to explore the most up-to-date software available and analyse their strengths and weakness in order to illustrate their choice of hardware in a given scenario, through class discussion or small group research that could be fed back to the larger group.

Learners should consider different types of file storage and understand their purposes.

They could also work in pairs or small groups to investigate and provide feedback on specific types of security software such as antivirus, anti-spam and anti-malware, demonstrating a real understanding of the differences between these, and identifying examples of commonly used software to provide this functionality.

Assessment outcome 4

Learners could carry out individual research and then have a class discussion on the difference between information and data or a research project by small groups, who could report back their findings to the larger group for discussion through presentations or poster presentations, for example. Learners could be presented with information that has been broken down in to its component data items – eg lists of dates, names, places, images – and be asked to consider how they could be organised to provide information.

The difference between qualitative and quantitative data could be discussed by the class reviewing examples of both, eg feelings, colours, preferences (qualitative) and election results, annual rainfall, age profiles (quantitative).

It is also important that learners recognise that information must be checked to see that it is accurate, valid, timely and objective. The learners could be given examples of good and poor information and instructed to consider whether they meet the criteria and provide their reasoning. Online tutorials are also available to assist understanding of the characteristics of information. Learners could produce leaflets, booklets or electronic resources after carrying out their own investigations in small groups or individually.

Learners could be given examples of data and consider the processing required to convert them into useful information. The steps that they take could be used as the basis of a class discussion of the data processing cycle. They could identify the input data and the range of operations (both arithmetic and logical) that they carried out on the data; the final output could then be drawn together as a set of instructions or diagrammatic representation of the cycle.

Assessment outcome 5

Concepts such as character encoding and assembly language often involve numbers systems such as binary and hex, eg ASCII characters encoded in binary, performing arithmetic of hexadecimal values in assembly language etc. Learners should be able to accurately **manipulate** the various number systems eg addition, subtraction, conversions as well as understand the relationships between the machine code level and the complex higher level languages written from a human rather than a machine perspective. This will include the ability to **evaluate** different character encoding systems in terms of range of characters available, for example.

It is important that learners undertake practical exercises on number conversion, from one base to another, and practice arithmetic calculations; these must include denary, binary and hexadecimal bases and fractions. It may be possible to integrate this element of the assessment outcome with maths lessons or enable learners to develop presentations on number bases for other learners. The same discussions could include the reasons for binary and hexadecimal being essential to

computer instructions, linking it to bits, bytes, nibbles and words. From this, the matching of binary or hexadecimal numbers to human understandable characters could be discussed and small group research conducted into the reasons for the development of different character codes such as ASCII, Unicode and Extended Binary Coded Decimal Interchange Code (EBCDIC) and how they differ from each other.

The learners could research the different types of programming language and produce a diagram of their position on a line from machine understandable to human understandable formats. The programs must include machine code, assemblers, high level and 4GLs. The outcomes could then be discussed in terms of the ways in which different programs can be translated in to machine code.

Useful links and resources

Books

- Hedly S and Aplin T, *Blackstone's statutes on IT and e-commerce,* 4th edition, ISBN-10 0199238219, ISBN-13 978-0199238217, Blackstone Press (2008).
- Reed C (ed.), *Computer law*, ISBN-10 0199696462, ISBN-13 978-0199696468, Oxford University Press (2012).
- Render B, Stair R and Hanna M, *Fundamentals of information systems*, 7th edition, ISBN-10 1133629628, ISBN-13: 978-1133629627, Mason, OH, South-Western College Publishing, 2013.
- Shelly GB and Cashman TJ, *Computer fundamentals for an information age*, ISBN 0-88236-125-2, Anaheim Publishing Company, Brea, CA, 2013.
- Burdett A, Bowen D, Butler D, Cumming A et al, *BCS glossary of computing and ICT*, ISBN-13 9781780171500 (2013).

Websites

- e-booksdirectory.com (online books for download or reading, some free resources).

12.2 Unit 2: Computer programming

Title	Computer programming
Unit number	F/507/6465
Assessment	Externally assessed
Guided learning hours	90
Transferable skill(s) contextualised within this unit	N/A
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms. It would be helpful for demonstration purposes, if at least the tutor's computer was given permissions that would allow access to at least one Integrated Development Environments (IDE) suitable for learning. Obviously, if learners are to carry out practical tasks in order to reinforce their theoretical learning, they would require access also.
Synoptic assessment within this unit	IT: Scripting and App Programming linked to Units 1, 3 and 4. IT: Programming linked to Units 1, 3, 4, 5, 6, 7 and 8. Unit 1 provides complementary coverage of programming as a specialised aspect of computer system usage. Unit 5 provides underpinning knowledge of number systems and Boolean logic which is critical to the storage of program data and the creation of conditional expressions. Units 3, 4, 6 and 7 build on the foundations of the computer programming unit to pursue different programming approaches and paradigms, encouraging the learner to explore their varied concepts, solution types and development pathways. To complete an industrial project (Unit 8), learners will make use of an extensive selection of programming skills and techniques gained from studying the units in this programme to help them develop a solution for a client or end user. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

To equip learners with the necessary knowledge and understanding of generic theory, practices and concepts associated with high quality professional programming solutions, designed and developed to meet client requirements.

Unit introduction

There are a number of approaches to the design and development of coding solutions to real world problems, such as traditional algorithm-oriented solutions, event driven and object oriented programming. The key to having the ability to successfully design and develop programs using any of these approaches is for the learner to possess a thorough, fundamental understanding of the general theories, practices and concepts that they are based on.

Learners will build extensive knowledge about different types of programming, the software design life cycle, and the design and development tools used by developers such as flowcharts and structure diagrams. Learners will also gain understanding about the key coding features and techniques that are utilised in various industry-standard programming languages: searching and sorting algorithms, constructs including sequence, selection and iteration, as well as functions and data types. The unit content focuses on the principles of good coding practice and intuitive user interface design.

Unit content

Different types of computer programming and the key features	
Different types of computer programming	• Low-level languages. • Machine code. • Assembly language. • High-level languages.

Different types of high-level programming paradigms and associated languages	• Concept of a programming paradigm.
	• Procedural paradigm:
	  • BASIC languages, Pascal, Fortran, C, Java, C++ etc.
	• Functional paradigm:
	  • LISP, Logo, Scheme etc.
	• Object oriented paradigm:
	  • Characteristic (eg polymorphism)
	  • C++, Ruby, Python, Java etc.
	• Event driven paradigm:
	  • Characteristics (eg triggers)
	  • MS Visual Basic .NET, JavaScript, Java, Objective-C etc.
	• Scripting paradigm:
	  • AppleScript, VBScript, ColdFusion etc.
	• Logic paradigm:
	  • Prolog, Oz, CLACL etc.

Common uses	• Web browsers. • Web servers. • Enhanced web page interactivity. • Scientific software. • Operating systems. • Game development. • Cross platform development. • Graphical user interfaces (GUIs). • Mobile platforms.

Tools and techniques for planning, design and development	

Software design lifecycle	• What is the software design lifecycle and what is its purpose?
	• Different development approaches:
	  • waterfall model
	  • spiral model
	  • incremental model
	  • agile model
	  • v-model
	  • iterative model
	  • modular.
	• Advantages/disadvantages of the different development approaches.
	• Different phases of the software design lifecycle; purposes and outcomes.
	• Requirement gathering.
	• Understanding the problem.
	• Design.
	• Implementation or coding.
	• Testing.
	• Deployment.
	• Maintenance.
	• Advantages/disadvantages of the software design lifecycle.

Flowcharts	• What is a flowchart and what is its purpose?
	• Key terms and elements of a flowchart:
	  • sequences of actions
	  • inputs and outputs
	  • processes
	  • decisions
	  • direction of flow
	  • on/off page linkage.
	• When to use flowcharts.
	• Basic flowchart development procedure.
	• Advantages/disadvantages of flowcharts.

Structure diagrams	• What is a structure diagram and what is its purpose?
	• Key terms and elements of a structure diagram:
	  • sequence
	  • selection
	  • iteration.
	• Functional decomposition.
	• When to use a structure diagram.
	• What a structure diagram describes.
	• Advantages/disadvantages of structure diagrams.

Tools and techniques for planning, design and development	
Pseudocode	• What is pseudocode and what is its purpose? • Key terms and elements of pseudocode. • How is pseudocode written? • Advantages/disadvantages of pseudocode.

Storyboards	• What is storyboarding and what is its purpose?
	• Key features of storyboarding:
	  • sequence of scenes
	  • connections.
	• When is storyboarding used?
	• Storyboarding in user interface design.
	• Advantages/disadvantages of storyboarding.

Pair/quad programming	• What is pair/quad programming and what is its purpose? • Organisation of pair/quad programming. • How is pair/quad programming used? • Advantages/disadvantages of pair/quad programming.
Data tables	• What are data tables and what are their purposes? • Key features of data tables. • How are data tables used? • Advantages/disadvantages of data tables.

Action tables	• What is an action table and what is its purpose?
	• Key features of action tables:
	  • code blocks
	  • decisions
	  • local variable
	  • while loops/do-while loops/for loops
	  • return blocks
	  • break blocks.
	• How are action tables used?
	• Advantages/disadvantages of action tables.

Unified modelling language (UML)	• What is UML and what is its purpose?

Features and techniques used in computer programming	

Key features of computer programming languages	• Variable or object types:
	  • single character
	  • integer
	  • real
	  • strings
	  • arrays
	  • pointers
	  • structures or records.
	• Variables or objects scope:
	  • global
	  • local.
	• Declarations.
	• Statements.
	• Expressions.
	• Assignments.
	• Constructs:
	  • sequence
	  • selection
	  • simple if
	  • nested if
	  • case type statement.
	• Iteration:
	  • pre-condition loops
	  • post-condition loops.
	• Modularity and functions:
	  • arguments and parameters
	  • functions with no arguments and no return value
	  • functions with no arguments but do have a return value
	  • functions with arguments but no return value
	  • functions with arguments and a return value.
	• Data structures:
	  • array
	  • First In, First Out (FIFO), eg Queue
	  • Last In, First Out (LIFO), eg Stack
	  • data files, eg text files, binary files (record) etc.
	• Source files:
	  • valid program statements as defined by the language
	  • what are they used for?

Features and techniques used in computer programming	

Algorithms	• What are algorithms and what are their purposes?
	• Classes of algorithm:
	  • simple recursive algorithms
	  • backtracking algorithms
	  • divide and conquer algorithms
	  • dynamic programming algorithms
	  • greedy algorithms
	  • branch and bound algorithms
	  • brute force algorithms
	  • randomised algorithms.

Testing – compiling and debugging computer programs	• Testing techniques:
	  • black box vs white box testing
	  • open and closed beta testing
	  • workflow testing.
	• Selecting test data, eg normal, extreme, invalid.
	• Code coverage, eg percentage, logical pathways.
	• Bench run and test run
	  • actual vs expected results
	  • screen captures etc
	  • remedial actions, eg code modification, bug fixes.
	• Debug tools.
	• Watches.
	• Traces and step into.
	• Breakpoints.
	• Code inspection.
	• Trace tables.
	• Expected and actual results:
	  • releases and versioning, eg major/minor releases, patching, concurrent versioning system (CVS)
	  • recording outcomes and recommendations, eg known bugs, common vulnerabilities and exposures (CVEs).

Reviewing	• Importance of regular project reviewing against specification requirements. • Iterative project design and implementation process.
Technical documentation	• Clear, concise technical writing. • Appropriate use of technical terminology. • Suitable document structure and layout. • Applicable for target audience.

Features and techniques used in computer programming	
User documentation	• Clear, concise 'Plain English' writing. • Screen captures. • Use cases. • Lack of jargon. • Use of only necessary technical terminology. • Suitable document structure and layout. • Applicable for target audience.

The principles of good program practice and user interface design	

Good practice in application design and programming	• Good programming principles:
	  • avoid repetition of code
	  • abstraction principle
	  • KISS – Keep It Simple Stupid!
	  • avoid adding functionality until you need it
	  • always opt for the simplest solution that will work
	  • always make your code as easy to read and as understandable as possible
	  • open/closed principle
	  • always write code for the person who is going to maintain it
	  • always follow standard conventions and practice. Don't build in surprises!
	  • single responsibility principle
	  • always try to code to minimise dependencies in other areas of the program
	  • place code that has similar functionality within the same component
	  • avoid optimising until the code is working
	  • always try to reuse code where possible
	  • familiarity with existing systems
	  • separate different areas of functionality into distinct modules of code.

Assessment outcomes

Assessment outcome 1: Understand the different types of computer programming, languages and the common uses	
a	The features and application of high and low-level languages.
b	Paradigmatic structures that guide the design of high-level programming languages.
c	Programming languages associated with each paradigm and their common uses.

Assessment outcome 2: Analyse the tools and techniques for planning, design and development	
a	The software design lifecycle, its purposes and outcomes.
b	Software development models and the merits or otherwise of different development approaches.

Assessment outcome 2: Analyse the tools and techniques for planning, design and development	
c	The purpose of Unified Modelling Language (UML) and its application throughout the development lifecycle.
d	Compare and contrast the modular nature of structure diagrams with the representation of logic in flowcharts and identify key terms, elements and appropriate application.
e	The principles and use of pseudocode to describe algorithms.
f	The principles of storyboarding and pair/quad programming in the design and development process.
g	The utility of data and action tables identifying their key features and purposes.

Assessment outcome 3: Evaluate the key features and techniques used in computer programming	
a	The elements and data structures of programming languages.
b	The purpose and features of source files
c	Classes of algorithms and their utility for a given situation.
d	Testing methods and techniques as a process of exercising software to verify it satisfies requirements and to detect errors.
e	The process of analysing a software item to detect the differences between existing and required conditions.
f	The importance of regular project reviewing against specification requirements, including iterative design.
g	The conventions of software versioning and the software release lifecycle.
h	The features of technical and user documentation for a target audience.

Assessment outcome 4: Demonstrate the principles of good program practice and user interface design	
a	The principles of good programming in application design and the efficiencies gained.
b	The purpose and application of the principles of modular application development.

Assessment

This unit is assessed through an external examination set and marked by AQA. The examination takes place under controlled examination conditions and the exam date will be published at the start of each academic year.

Learners are allowed to use a non-programmable scientific calculator in the examination.

The examination consists of a written paper with two Sections, A and B. Learners have to complete both sections and there are no optional questions within either section.

The examination is 2 hours duration and the total number of marks available in the examination is 80.

Section A is worth 50 marks and consists of relatively short questions based on the whole of the specification for this unit. Learners are required to answer **all** of the questions in Section A.

Section B is worth 30 marks and includes longer questions worth up to 15 marks each. The questions in Section B do not necessarily cover the whole of the specification for this unit at each assessment. Learners are required to answer all of the questions in Section B.

Employer engagement guidance

The organisation, its staff and learners must have access to employers and expertise. The organisation will have computer/technical staff who will understand the practical activities identified in the assessment outcomes. Local employers could be invited to discuss the skills and knowledge they require to support their IT systems, to inform the structure and specific hardware and software identified in the unit.

Employers may also be able to provide opportunities to visit IT facilities or provide placement or shadowing opportunities for assessors and/or learners to provide updating of the former and learning opportunities for the latter.

Employers could be invited to an apprenticeship forum.

The British Computer Society (BCS) and the Association of Computing Machinery are two examples of professional bodies who engage with learners.

Delivery guidance

Assessment outcome 1

This unit introduces learners to a broad range of aspects of computing. Through tasks such as individual research and report writing, learners should study the different levels of computer programming, eg machine code, assembly language and high level languages.

Learners could be presented with group-work opportunities to research and review common high level languages (such as C#, Pascal, Java, C++ and MS Visual Basic).

Learners could work in groups or individually to compare different aspects and functionality of the languages and discover the common uses for these languages, (eg for web browsers, game development, graphical user interfaces or mobile platforms etc).

Assessment outcome 2

It is important that learners study as many different planning, designing and development tools and techniques as possible. These should reflect the tools and techniques that are commonly used within the software industry to design and produce new products.

Exercises could take the form of small group project tasks, where learners are given scenarios of increasing complexity. Learners should select and utilise different tools and techniques to produce effective preliminary designs of programs that would be created by a software design team prior to the coding stage of a project. This may be an opportunity to engage with a local software company, who could provide learners with industry-relevant scenarios that they can investigate and design solutions for.

Assessment outcome 3

It is important that learners gain a thorough understanding and a competent ability to utilise the range of common algorithms. This could be done via group classroom exercises and small individual projects involving writing algorithms in pseudocode.

Learners should be encouraged to research a variety of syntax and constructs associated with different programming languages. With a view to maintaining consistency, it may be necessary to illustrate aspects of coding predominantly using examples from one particular language. However, it should be emphasised that, as this is a generic unit, learners should be taught that other languages have different syntactic approaches to building the same programming structures.

Learners should also be taught generally about aspects and processes regarding the testing of software; they should understand (and be able to explain) terms, tools and techniques associated with testing procedures. This could be done via individual research projects and writing test plans and policies for different purposes.

Learners should understand the importance of reviewing projects against the original project specifications and the iterative approach, which is commonplace in modern software project management. Learners should understand the need for clear, concise and instructive user and technical documentation. They could be given tasks involving the instruction of everyday processes but emphasising the differences in instructing technicians compared to non-technical users. This could lead to learning about the differing audiences of processes and aspects associated with software installations.

There should be an opportunity for learners to engage with industry specialists, particularly so learners can understand that modern program development is a team-led activity.

Assessment outcome 4

Learners should be taught to understand the importance of maintaining good practice in application design and programming. They could start by researching what particular aspects are regarded as good or bad practice and explaining the reasons why. This could be in the form of an essay or report.

Learners could then research good practice in user interface design and programming by designing on paper good features of user interface design and using callouts and annotations to explain why it is important to include such aspects within designs.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

AO1: Understand the different types of computer programming and the key features

There are many types of programming language; each has unique features and uses. This assessment outcome lets the learner explore the most common programming languages (eg C++, C#, Java, PHP, VB.Net etc) and appreciate their application in industry.

Unit 1 – AO5: Demonstrate how computers process user requirements
This complementary learning outcome discusses different types of language (natural and computer-based), low- and high-level languages and the roles of interpreters, compilers and assemblers. Learners should benefit from examining programming languages as part of a computer system and the role they play in creating operating systems, bespoke applications and utility software.

Unit 3 – PO1: Understand the key features of website technologies

Unit 4 – PO1: Understand the key features of mobile application development

Unit 6 – PO1 : Understand the key features of event driven programming languages

Unit 7 – PO1: Understand object oriented programming (OOP)
Although this assessment outcome provides a thorough overview of the different types of programming languages available and their typical applications, each unit shown above has a performance outcome that examines this in much greater depth, providing the learner with an expanded appreciation of each distinct programming language they study and in which circumstances they should be used.

AO2: Analyse the tools and techniques for planning, design and development

This assessment outcome introduces the learner to the software design lifecycle and a range of different tools that can be used to design a programmed solution.

Unit 4 – PO3: Demonstrate the ability to design mobile applications

Unit 6 – PO3: Demonstrate the ability to design event driven applications

Unit 7 – PO2: Design software solutions using an object oriented approach
Each of these performance outcomes exercises the selection and use of different design tools to create working solutions. The learner will quickly appreciate which design tool is most appropriate depending on the programming paradigm being used.

AO3: Evaluate the key features and techniques used in computer programming

This assessment outcome focuses on various elements that comprise a modern programming language. This includes the statements, constructs, operators, data types, structures, algorithms that form the language and tools used to compile and debug solutions. In addition learners discover the requirements of creating successful user and technical documentation.

Unit 5 – AO1: Understand and manipulate data in common number systems

Unit 5 – AO2: Understand and apply the foundations of computer logic
Many programming tools and techniques rely on understanding how computers work with different number systems (eg variables and data types) and apply Boolean logic (eg IF statement conditions, loop conditions etc). Learners typically apply number system and Boolean logic theory successfully when creating working programs.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Many of the tools and techniques encountered in this assessment outcome are practically implemented through selected programming languages in this unit's performance outcomes, typically in JavaScript and PHP respectively. Learners' understanding of these tools and techniques are developed and reinforced through practical application.

Unit 4 – PO2: Apply the key features and functions of mobile application programming languages

Unit 4 – PO4: Demonstrate the proficient use of mobile development tools

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
Many of the tools and techniques encountered in this assessment outcome are practically implemented through the selected programming language in this unit's three performance outcomes, typically in Java (for Android) or Objective-C (for Apple iOS). Learners' understanding of these tools and techniques are developed and reinforced through practical application.

Unit 6 – PO2: Demonstrate the use of event driven language features and functions

Unit 6 – PO4: Implement event driven applications to a professional standard
Many of the tools and techniques encountered in this assessment outcome are practically implemented through the selected programming language in this unit's two performance outcomes, typically in VB.Net. Learners' understanding of these tools and techniques are developed and reinforced through practical application.

Unit 7 – PO3: Implement object oriented applications to a professional standard

Unit 7 – PO4: Understand how to test and maintain programs
Many of the tools and techniques encountered in this assessment outcome are practically implemented through the selected programming language in this unit's two performance outcomes, typically in C++, Java or C#. Learners' understanding of these tools and techniques are developed and reinforced through practical application.

A04: Demonstrate the principles of good program practice and user interface design

This assessment outcome encourages the learner to apply good principles when creating program code (eg using simplest solution, reusable code, readability etc) and the user interface.

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development

Unit 6 – PO4: Implement event driven applications to a professional standard

Unit 7 – PO5: Understand how to produce documentation

These performance outcomes apply the learner's understanding of good program practice by tasking them with documenting their code correctly (ie comments, layout etc), for users, for technical purposes and, in the case of mobile applications specifically, for digital distribution.

Useful links and resources

Books

- Wang W, *Beginning programming for dummies*, 4th edition, ISBN-10 0470088702, ISBN-13 9780470088708, Wiley Publishing Inc., Hoboken, NJ (2007).
- Wang W, *Beginning programming all-in-one desk reference for dummies*, 1st edition, ISBN-10 0470108541, ISBN-13 9780470108543, Wiley Publishing Inc., Hoboken, NJ (2008).

Websites

- 10 classes of algorithms every programmer must know about: hbfs.wordpress.com/2008/12/23/the-10-classes-of-algorithms-every-programmer-must-know-about
- Best practices for software development projects: ibm.com/developerworks/websphere/library/techarticles/0306_perks/perks2.html
- Code.org: code.org/learn
- Code Academy: codecademy.com
- UML diagrams: smartdraw.com/resources/tutorials/uml-diagrams
- What is an algorithm: wisegeek.org/what-is-an-algorithm.htm

12.3 Unit 3: Website technologies

Title	Website technologies
Unit number	M/507/6476
Unit assessment type	Centre assessed and externally quality assured
Recommended assessment method	Practical assignment This is the preferred assessment method for this unit. A centre may choose an alternative method of assessment, but will be asked to justify as part of the quality assurance process.
Guided learning hours	90
Transferable skill(s) contextualised within this unit	Research[4]
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms, which offer Integrated Development Environments suited to learning and exploring the various website technologies listed here. These include, but are not limited to: Microsoft's Active Server Pages (ASP.net), PHP, Java, JavaScript, multiple web browsers (and, ideally, versions), SSH client, syntax highlighted text editors, FTP server and client, Firewall software and Web server software such as Apache or IIS. In addition some aspects of the unit will require user rights on the development and target platforms to compile programs and install executable code. Firewalls may also have to be configured to permit data transfers on selected protocols (eg FTP, SSH) between the server and its clients. Learners should also have access to suitable offline and online learning material, manuals, help sheets and coded examples in order to encourage self-sufficiency.

[4] Please visit the specification homepage to access the transferable skills standards and associated guidance and recording documentation.

Synoptic assessment within this unit	IT: Scripting and app programming linked to Units 1, 2 and 4. IT: Programming linked to Units 1, 2, 4, 5, 6 and 7. Unit 1 provides the learner with a solid grounding in the different types of software and hardware that are likely to form part of a website technology solution. In addition this unit also introduces the concept of programming languages and relational database systems, which, when combined, form powerful website solutions. Unit 2 can be seen as the introductory unit for programming and as such introduces many of the concepts used in website technologies, from an overview of the different types of programming languages available to the key features common to them all and the intricate differences in usage and concept that set them apart. Unit 5 gives the learner the opportunity to sharpen their mathematical skills, proving essential for learners when tackling the logic, arithmetic and algorithms needed to get the very best out of website technologies. Units 4, 6 and 7 focus on mobile applications programming, event driven programming and object oriented programming. Web technology languages that work client-side often exercise event driven programming features and concepts while server-side solutions are often best implemented by using object oriented techniques. Despite their differences these three units collectively form a cohesive approach that reinforces the learner's problem-solving skills and awareness of common elements such as code syntax, functions and debugging, which are crucial to leveraging website technology solutions effectively. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

To equip learners with the necessary knowledge and practical ability to design and build interactive websites and cloud-based applications that professionally meet client needs through their applied understanding and blending of client-side and server-side technologies.

Unit introduction

The rise of website applications and, in particular, global e-commerce has greatly altered the chosen platform for many commercial systems. There is no longer a need to distribute bulky CDs and DVDs: it is now possible to run services directly from the internet through server-based applications that can be maintained, updated and secured centrally. This method always offers the client the most up-to-date copy of the software with only a small footprint impinging on the valuable resources of their client PC, Apple Mac or mobile device.

In this unit, learners will discover not just how to deploy secure web services, but how to design and create cloud-based applications, typically through a hybrid of client-side technologies (such as HTML, CSS and JavaScript) and server-side technologies (such as PHP or Microsoft ASP .Net). Emphasis is placed on understanding the roles of each component, to produce data-driven code that meets modern standards and expectations and to ensure that the solutions created are both robust and secure, protecting their users and host platforms from unwanted intrusion.

It is a skill set that every software developer should have and be self-assured in its use and practice.

This unit provides an opportunity to evidence achievement of the transferable skill of research.

Unit content

Key features of website technologies	

Supporting technologies	• Hardware:
	  • server-side
	  • web server
	  • relational database server
	  • client-side
	  • desktop, eg PC, Apple Mac etc
	  • mobile device, eg smart phone, tablet etc.
	• Software application:
	  • server-side
	  • web server, eg Apache HTTP, Microsoft
	  • internet information services (IIS), Nginx, Google GWS etc
	  • relational database system, eg MSSQL, SQL etc
	  • client-side
	  • web browser, eg Internet Explorer (IE), Google Chrome, Mozilla Firefox, Apple Safari, Opera etc
	  • file transfer protocol (FTP) client, eg Filezilla, WS_FTP etc
	  • secure shell client, eg PuTTY
	  • text editors, eg Notepad++, TextEdit, TextWrangler etc
	  • web development ('WYSIWYG') tools, eg Adobe Dreamweaver.
	• Pre-packaged bundles, eg LAMP, WAMP, MAMP etc.
	• Protocols and ports:
	  • file transfer protocol (FTP), port 21
	  • secure shell (SSH), port 22
	  • hypertext transfer protocol – HTTP, port 80
	  • hypertext transfer protocol – secure sockets layer (SSL), port 443.

Website technologies	• Separation of content from formatting:
	  • markup languages eg HTML
	  • style sheet languages, eg CSS.
	• Client-side scripting languages, eg JavaScript, Adobe Flash, Oracle Java.
	• Server-side scripting, eg PHP hypertext pre-processor (PHP), active server pages (ASP.net), java server pages (JSP) etc.
	• Third party libraries, eg jQuery.
	• Data description methods, XML (eXtensible Markup Language), JSON (JavaScript Object Notation).

Key features of website technologies	
Website standards	• World wide web consortium (W3C). • HTML 5. • CSS 2 and 3. • JavaScript. • Markup validation service for HTML, CSS etc. • Client compliance and testing, eg web browser HTML5 test, Acid3.
Market share and penetration	• Web server share. • Web browser client share. • Server-side programming share, eg PHP vs ASP .net.

Key features and functions of a markup language	
HTML elements and syntax	• Purpose (content). • Tags, nested tags. • Tag attributes. • DOCTYPES (standards), eg HTML 4.01 transitional or strict, XHTML1.1, HTML5 etc. • Case sensitivity.
HTML page structure	• <html>...</head> HTML document. • <head>..</head> HTML title, meta tags, eg keywords, description, author, refresh etc. • <body>...</body> HTML body.

Key features and functions of a markup language	

HTML basic tags	• Structure and text tags:
	  • ... Paragraph
	  • <h>...</h> Headline
	  • <pre>...</pre> Pre-formatted text
	  • Horizontal rule.
	• Comments.
	• Links <a>...:
	  • hyperlink to another resource (page or domain)
	  • image link
	  • mailto link
	  • same page target
	  • same page hyperlink.
	• Lists:
	  • ... ordered
	  • ... unordered
	  • ... items.
	• Tables:
	  • <table>...</table> table
	  • <th>...</th> header cell
	  • <tr>...</tr> row
	  • <td>...</td> cell
	  • <caption>...</caption> caption
	  • <colgroup>...</colgroup> group columns
	  • <col>...</col> column properties
	  • <thead>...</thead> group header content
	  • <tbody>...</tbody> group the body content
	  • <tfoot>...</tfoot> group the footer content.
	• Images <img> and appropriate file formats.
	• Embedded objects, eg Adobe Flash.

HTML forms	• <form> Forms.
	• Form methods (post, get).
	• Form actions.
	• Form elements
	  • <input type=”text” /> single line text box
	  • <input type=”checkbox” /> checkbox
	  • <input type=”radio”/> radio button
	  • <input type=”file”/> file upload
	  • <textarea>...</textarea> multiline text box
	  • <select>...</select>, <option> pulldown menu
	  • <input type=”button”/> action button
	  • <input type=”submit”/> submit button.
	• Form name and ID attributes.

Key features and functions of a markup language	
HTML5	• New elements, eg new media (audio, video etc). • Removed elements, eg frames, <font>, , <applet> etc. • Web browser support.

Key features of a style sheet language	
CSS elements and syntax	(eg) • Purpose (format and layout). • Selectors: types (element, id, class), grouping. • Declaration (property, value pairs). • Comments. • Pseudo classes and elements, eg <a> link, visited, active, hover, before, after etc. • Cross-browser issues, eg rendering variances.
CSS units	• Measurement values, eg %, in (inch), cm (centimetre), mm (millimetre), em, ex, pt (point), pc (pica), px (pixels).
CSS box model	(eg) • Content. • Padding. • Border. • Margin. • Effect on height and width of content.
CSS insertion methods	(eg) • External style sheet <link> • Internal style sheet <style>...</style> • Inline styles. • Inheritance priority from multiple style sheets.
CSS common properties	(eg) • Background. • Border. • Colour. • Font. • Height. • Margin. • Padding. • Text-align. • Width. • Combination of common properties to produce visual themes. • Circumstances when alternate visual themes are necessary, eg high contrast for visually impaired users.
CSS colours	(eg) • Colour value formats, eg Hexadecimal, RGB (Red Green Blue), RGBA (Red Green Blue Alpha channel) etc. • Colour names.

Key features of a style sheet language	
CSS 3	• Web browser support (browser and version). • Cross-browser differences.

Key features and functions of a client-side scripting language	

JavaScript elements and syntax	• Purposes (adding web page behaviour), eg:
	  • change HTML element content via Document Object Model (DOM)
	  • change HTML element attributes via DOM
	  • change HTML CSS styles via DOM
	  • validate HTML form input
	  • add interactive elements, event handling.
	• Declaration section <script>...</script>
	• Comments.
	• Case sensitivity.

JavaScript syntax	• Identifiers, variables (local, global) and data types.
	• Arrays.
	• Literals.
	• Operators:
	  • arithmetic, eg +, -, *, ., %, ++, --
	  • assignment, eg =, +=, -=, *=, /=, %=
	  • concatenation, eg +
	  • relational, eg ==, ===, !=, !==, >=, <=, <, <
	  • logical, eg &&, ||, !
	• Keywords and statements.

Common language constructs	• Sequence. • Selection, eg if...else, nested if, switch. • Iteration, eg pre-conditioned, post-conditioned, for...each mechanism.

JavaScript functions	• Declarations (name, parameters, code to be executed, return value).
	• Call and execution:
	  • hoisting
	  • self-invoking.
	• Third party libraries, eg jQuery.

JavaScript objects	• Objects (including primitive data types, eg Numbers, Booleans, Strings etc). • Methods. • Properties.
JavaScript events	• Common HTML events, eg onblur, onchange, onclick, onmouseover, onmouseout, onkeydown, onload, onsubmit etc. • Adding events to HTML elements, eg Form elements (buttons, text box, etc).

JavaScript debugging	• Debugging tools:
	  • built-in web browser, eg Chrome Developer Tools, JavaScript Console
	  • third party, eg Firebug.
	• Exception handling, try...throw...catch block.

Key features and functions of a server-side scripting language	
Server-side script elements and syntax	• Purpose (creation of dynamic web page content, particularly when linked to a backend relational database system), validate HTML form input etc. • Case sensitivity.
Identifiers	• Variables and constants (programming and system defined). • Data types, eg integer, decimal, Boolean, string, character, date/time, currency etc. • Value range. • Declaration and initialisation. • Scope and visibility, eg public, private, local (module-level), global, namespaces etc.
Data structures	• Array, eg one-dimensional and multi-dimensional. • Lists, stacks, queues. • Text files, XML files. • Relational databases (SQL, MSSQL etc), tables, records and fields.
Operators	• Arithmetic. • Relational. • Logical. • Concatenation.
Common language constructs	• Sequence. • Selection, eg if...else...endif, nested if, switch/case. • Iteration, eg pre-conditioned, post-conditioned, for...each mechanism.

Functions	• Programmer-defined functions:
	  • naming and declaration
	  • formal and actual parameters
	  • function calls
	  • return type.
	• Common library functions:
	  • input and output
	  • arithmetic
	  • string
	  • date
	  • file.
	• Third party library functions.

Server-side classes and objects	• Classes and objects (including primitive data types, eg Numbers, Booleans, Strings etc). • Methods. • Properties.

HTTP request headers	• Header content, eg posted values, URL encoded parameters.
	• Accessibility of header content, eg:
	  • PHP Superglobals, eg $_POST, $_GET
	  • ASP.net request object.

Accessing backend databases	• Relational database management system (RDMS) overview.
	• RDMS DDL (Data Definition Language) commands.
	• RDMS DML (Data Manipulation Language) commands.
	• RDMS DCL (Data Control Language) commands.
	• Accessing backend database via web server scripting to:
	  • select data (with where clause)
	  • insert data
	  • delete data
	  • update data.
	• Stored procedures.
	• Encryption and decryption, eg advanced encryption standard (AES).

Vulnerabilities and counter threats to website technologies	

Common client-side threats	• Current trends and attack vectors.
	• Current and classic examples, eg:
	  • clickjacking
	  • HTML injection and cross-site scripting (XSS)
	  • client-side cookies
	  • password autocomplete.

Common server-side threats	• Current trends and attack vectors.
	• Current and classic examples, eg:
	  • SQL injection
	  • directory traversal (eg downloading files)
	  • source code reveal (bad server configuration)
	  • remote file inclusion (untrusted sources)
	  • session hijacking
	  • non-patching of flawed code, eg 'Heartbleed'.

Client-side protection	• Current and recommended techniques, eg:
	  • cache options (eg no cache, no autocomplete)
	  • client browser updates
	  • active cookie management.

Server-side protection	• Current and recommended techniques, eg:
	  • server configuration and patching for web server, server-side script engine and relational database
	  • secure sockets layer (SSL) certification
	  • validation and stripping of HTML tags/JavaScript from inputs
	  • HTTP only cookies
	  • web vulnerability scanner, eg Acunetix
	  • awareness, eg common vulnerabilities and exposures (CVE) list.

Performance outcomes

On successful completion of this unit learners will be able to:

Performance outcome 1:	Understand the key features of website technologies.
Performance outcome 2:	Demonstrate key features and functions of a markup language.
Performance outcome 3:	Demonstrate key features of a style sheet language.
Performance outcome 4:	Demonstrate the key features and functions of a client-side scripting language.
Performance outcome 5:	Demonstrate the key features and functions of a server-side scripting language.
Performance outcome 6:	Recognise vulnerabilities and counter threats to website technologies.

Grading criteria

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO1: Understand the key features of website technologies	**P1** **Research** the **four** supporting technologies that enable web technologies to function.		
	P2 Using **research** describe the main characteristics and properties of the four supporting technologies that enable web technologies to function.	**M1** Compare and contrast the **five** different types of available website technologies.	**D1** Assess using well-considered examples, the importance of web standards and market share when selecting and using website technologies.
PO2: Demonstrate key features and functions of a markup language	**P3** Create a simple multi-page website for a specific user using HTML elements.	**M2** Enhance a simple website by adding form-based content for a specific user need.	**D2** Assess and evaluate opportunities presented by HTML forms.
	P4 Add media content for a specific user need using HTML elements.	**M3** Enhance a simple website by adding HTML5 features to meet a specific user need.	

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO3: Demonstrate key features of a style sheet language	**P5** Use CSS elements to format a simple multi-page website using a selected visual theme.	**M4** Enhance a simple website's format by providing a user with a choice of **two** different visual themes.	**D3** Assess and evaluate opportunities when other themes might be used to enhance the user experience.
	P6 Identify and demonstrate style sheet features which may cause cross-browser differences.	**M5** Enhance a simple website by creating style sheet content which compensates for cross-browser differences.	
PO4: Demonstrate the key features and functions of a client-side scripting language	**P7** Use JavaScript elements to add interactive elements to a selected website.	**M6** Enhance a simple website by adding JavaScript validation to its form-based content.	
	P8 Use debugging facilities to resolve errors in JavaScript elements.		
	P9 Create a JavaScript function to perform a selected complex task.	**M7** Apply a third party JavaScript library function to perform a selected complex task.	
PO5: Demonstrate the key features and functions of a server-side scripting language	**P10** Use server-side scripting elements to add dynamic content to a selected website.	**M8** Enhance a simple website by adding server-side validation and processing to its form-based content.	
	P11 Create and use a bespoke server-side scripted function to solve a set task.	**M9** Create and use a bespoke server-side scripted class to solve a set task.	
	P12 Use server-side scripting to query data in a relational database system.		

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
	P13 Use server-side scripting to manipulate data in a relational database system.		
PO6: Recognise vulnerabilities and counter threats to website technologies	**P14** Identify **four** common client-side threats.	**M10** Apply suitable measures to protect against a client-side threat in a selected website solution.	
	P15 Identify **four** common server-side threats.	**M11** Apply suitable measures to protect against a server-side threat in a selected website solution.	**D4** Interpret threat analysis data and justify which attack vectors pose the most vital threat to website integrity and data security.

Assessment amplification

This section provides amplification of what is specifically required or exemplification of the responses learners are expected to provide.

There is no mandatory division of grading criteria although, as can be seen from the Grading criteria grid, some are naturally linked either by learning outcome or by developmental theme. Connections between the various pass, merit and distinction criteria can therefore suggest logical assessment grouping, but do not necessarily have to be followed.

Finding a single real world scenario that can cover all criteria is difficult, but not impossible. The key to assessment in this unit, as in others, is to make assessment a natural outcome of the learning experience, with skills being assessed at a level appropriate to the learner's level of experience. The nature of the assessment technique is tutor-dependent, but some in this unit may only be achieved through generating working program code to a good standard.

Assessments that connect this unit to others via connected tasks are also encouraged and it is possible that evidence produced during work for this unit may be used to provide evidence for criteria in other units.

For **M4** learners should offer a choice of different formatting preset styles for a user.

For **P9** learners are required to create their own JavaScript functions to solve a complex task.

For **M7** learners should identify, understand and use a suitable third party library function to perform a complex task. The task may be the same as for P8 and it must involve a multi-stage process, ie not be merely a simple one-line calculation; many third party libraries exist for JavaScript and none in particular are mandated.

Employer engagement guidance

If learners are in the workplace then the centre could ask the employer whether there are any suitable projects that learners could work on as part of the team. It would helpful for the employer to be made aware of the sort of skills that learners have to practice.

Delivery guidance

It is recommended that this unit is taught after the conclusion of Unit 2: Computer programming, which introduces (in overview only) the concept of generalised programming techniques that support the demands of the various website technologies found in this unit. In addition, many elements have parallels in both the object oriented programming unit (eg PHP) and event driven programming unit (eg JavaScript); indeed, many of their respective programming languages (eg Visual Basic .Net, Objective-C, Java etc) can form a critical foundation.

Although it is suggested that the content is delivered to follow the order of the learning outcomes in this unit specification, it is not the only sequence that could be used and tutors are encouraged to consider the holistic nature of the learner's programme and the scheme itself. This is particularly evident for this unit where tutors may interleave various aspects (HTML, CSS, JavaScript, Server-side scripting) to create more rounded and industry representative solutions.

Learners must have access to the necessary hardware and software facilities in order to generate evidence for all of the grading criteria listed. As such, if centres cannot guarantee these resources, the unit should not be attempted.

It is suggested that the delivery of tools and techniques follows the pattern of a succession of small, but well chosen, exercises. These form a developmental toolkit that learners may use to build more complex solutions. This type of approach gently builds confidence and self-sufficiency in learners and is critical in promoting good problem-solving skills. Although remote web servers in shared environments may be used, it is beneficial for the learner to be able to observe server and client behaviour at close hand. Therefore, it is preferential that a web server is installed on a local area network that learners can access and inspect from the backend. This will permit access to some of the web server settings that are often obscured or unavailable in a shared hosting environment. Similar restrictions and levels of access should also apply to the relational database system that will be used by learners.

Performance outcome 1

This outcome focuses on the learner demonstrating their knowledge and understanding of the key principles of website technologies, essentially breaking them down into supporting technologies (hardware, software, protocols and ports etc) and website technologies (client-side, server-side languages etc). As part of this outcome learners will develop and be assessed on their research skills. Learners should examine the current market trends and penetration levels of the various hardware and software technologies in order to evaluate them and select the best possible combination for deployment.

The remaining five performance outcomes lend themselves to a substantive real world, complex task that is solved by using a secure cloud application that is designed and implemented by learners using a combination of client-side and server-side technologies.

Performance outcome 2

This outcome centres on a study of how to use hypertext markup language to build a multi-stage website. This website should include scope for the majority of the content from Performance outcomes 2.1 to 2.3 and will generate evidence for assessment. Learners should be taught how to enhance their system by including a data form and learners should assess their markup language in comparison to worldwide standards in order to evaluate the level of good practice they have demonstrated.

Learners should use their skills in a variety of contexts including the embedding of media content and elements from the new HTML5 standards.

Performance outcome 3

Learners should explore the use of CSS and add CSS elements to a multi-page website. They should be able to use and adapt preset styles. Learners must also explore the complexity of compensating for the different levels of style-sheet compatibility and compliance found in various web browsers. They should think of ways to adjust their style-sheet code to compensate for any variances.

Performance outcome 4

Focusing on client-side script through the use of JavaScript (although other client-side technologies exist, its practical use is explicitly named here because of its ubiquity in the worldwide website development process), learners will add interactive elements to a website and enhance a hypertext markup language data form by adding client-side validation of its entries. They will develop skills in using debugging tools to identify and resolve errors in their JavaScript code.

Using inbuilt functions and creating their own JavaScript functions will help learners to solve a complex task. They are asked to consider suitable third-party library functions to perform a complex task and should be able to identify them, judge their suitability and understand how they work.

Performance outcome 5

Performance outcome 5 switches the learner's focus to server-side scripting technologies. Here there is no mandated server-side scripting language as the market share demonstrates multiple popular development choices. Centres are advised to select and build a learner's skills in just one particular server-side scripting language, although learners are required to be aware of competing alternatives that may exist. Learners will create interactive website content via the chosen server-side scripting language, possibly adding or refining functionality. Bespoke functions and classes should also become familiar as they can contribute to solving a set task.

Learners will explore the connectivity and use of backend relational database systems, which are particularly important in commercial websites.

Performance outcome 6

To complete the unit, learners will need to explore, recognise and counter threats to websites. Generally these are divided between identifying and resolving client and server-side issues while also considering the learner's ability to understand the level of threat involved and evaluate the factors that represent the most pressing danger.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

P1: Research the four supporting technologies that enable web technologies to function
P2: Using research describe the main characteristics and properties of the four supporting technologies that enable web technologies to function

Unit 1 – AO1: Understand the different types of computer

Unit 1 – AO2: Understand the hardware requirements of a computer
One of the supporting technologies for web technologies to function is computer hardware. Learners should benefit from the focus spent in this unit's two assessment outcomes where types of computer hardware are examined. Learners could apply this background information to their discussion on the hardware used for both server-side and client-side systems.

Unit 1 – AO3: Understand the software requirements of a computer system
Another supporting technology necessary for web technologies to function is the actual software used on both the client and server sides. This assessment outcome introduces the learner to the different types of software available including the concept of operating systems, applications software and utilities. Many of the examples that the learner may discover here (eg web server, web browser, FTP client, text editor, secure shell etc) will be cornerstones of this unit.

P7: Use JavaScript elements to add interactive elements to a selected website

Unit 1 – AO5: Demonstrate how computers process user requirements
This unit introduces the key features of high level languages such as JavaScript and examples provided for this client-side scripting language could support the learner when adding interactive elements to a selected website, depending on the depth of coverage chosen.

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
This unit's assessment outcome examines different programming paradigms; aspects of the event driven and scripting paradigms when combined through JavaScript can be used to add interactive elements to a website. The learner may be able to draw upon previous sample code and concepts from this unit to assist in the completion of this performance outcome.

Unit 6 – PO1: Understand the key features of event driven programming languages
This unit's performance outcome showcases JavaScript as an example of a commercial programming language that reflects event driven design. Ideally the learner should be able to develop interactive elements by examining the coded samples this unit typically provides as it explores this paradigm's key features. In addition, coverage includes common uses of event driven programming languages through enhancing web page interactivity; this may provide the learner with some good ideas.

P8: Use debugging facilities to resolve errors in JavaScript elements

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
The assessment outcome in this unit introduces the concept of testing and debugging and examines the use of debug facilities such as watches, traces, breakpoints and code inspection. The learner should be able to apply tools and techniques such as these (present in a modern web browser to resolve errors in their JavaScript.

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
This performance outcome also focuses on debugging applications albeit in a mobile application development environment. The learner should be able to apply similar tools and techniques learnt here to resolve similar syntax and semantic errors found in their JavaScript.

Unit 6 – PO4: Implementing event driven applications to a professional standard
This performance outcome also focuses on debugging applications albeit in an event driven programming environment. The learner should be able to apply similar tools and techniques learnt here to resolve similar syntax and semantic errors found in their JavaScript.

P9: Create a JavaScript function to perform a selected complex task

Unit 1 – AO5: Demonstrate how computers process user requirements
This unit introduces the key features of high level languages such as JavaScript and examples provided for this client-side scripting language could support the learner when adding interactive elements to a selected website, depending on the depth of coverage chosen.

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
JavaScript functions are a form of modularity and this topic is firmly detailed in this unit's third assessment outcome where functions, arguments and parameters are introduced. This should provide a firm foundation, allowing the learner to understand the requirements of the complex task and develop a suitable JavaScript function to solve it using the correct syntax.

Unit 4 – PO2: The key features and functions of mobile application programming languages
This unit's second performance outcome also introduces the concept of functions and differentiates programmer-defined functions (as must be created by the user here to solve a complex task) from the common library functions that also exist. Although the languages used may be different, the problem-solving concepts are similar and the learner will benefit from the comparison and contrast that modular design in another programming language provides.

Unit 5 – AO2: Understand and apply the foundations of computer logic
Most functions that perform complex tasks will typically involve the application of Boolean logic in order to process the data successfully. Learners may rely on the introduction to this concept that is provided through the second performance outcome of this unit.

Unit 6 – PO2: Demonstrating the use of event driven language features and functions
This unit's second performance outcome also introduces the concept of functions and differentiates programmer-defined functions (as must be created by the user here to solve a complex task) from the common library functions that also exist. Although the languages used may be different, the problem-solving concepts are similar and the learner will benefit from the comparison and contrast that modular design in another programming language provides.

P10: Use server-side scripting elements to add dynamic content to a selected website

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
This unit's outcome examines different programming paradigms and aspects of the object oriented and scripting paradigms when combined through a suitable server-side scripting language (eg PHP or ASP.net) can be used to add dynamic content to a website. The learner may be able to draw upon previous sample code and concepts from this unit to assist in the completion of this performance outcome.

Unit 7 – PO1: Understand object oriented programming (OOP)
Server-side scripting languages such as ASP.net and PHP are viewed from an object oriented programming perspective in this unit. The learner should be able to utilise code samples and concepts introduced here to create the appropriate dynamic content.

P11: Create and use a bespoke server-side scripted function to solve a set task

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Server-side functions are a form of modularity and this topic is firmly introduced in this unit's third assessment outcome where functions, arguments and parameters are introduced. This should provide a firm foundation, allowing the learner to understand the requirements of the complex task and develop a suitable server-side function (in PHP or ASP.net, for example) to solve it using the correct syntax.

Unit 4 – PO2: The key features and functions of mobile application programming languages
This unit's second performance outcome also introduces the concept of functions and differentiates programmer-defined functions (as must be created by the user here to solve a complex task) from the common library functions that also exist. Although the languages used may be different, the problem-solving concepts are similar and the learner will benefit from the comparison and contrast that modular design in another programming language provides.

Unit 5 – AO2: Understand and apply the foundations of computer logic
Most functions that perform complex tasks will typically involve the application of Boolean logic in order to process the data successfully. Learners may rely on the introduction to this concept that is provided through the second performance outcome of this unit.

P12: Use server-side scripting to query data in a relational database system
P13: Use server-side scripting to manipulate data in a relational database system

Unit 1 – AO3: Understand the software requirements of a computer system
This unit's third assessment outcome introduces the concept of application software and would, typically, include the role of the relational database system in modern computers. As a direct consequence, learners should understand the concept of a relational database system and how it is structured and queried.

Unit 1 – AO5: Demonstrate how computers process user requirements
This unit introduces the key features of fourth generation languages (4GLs) such as Structured Query Language (SQL). Server-side scripting languages typically use a combination of library functions and SQL syntax to successfully query data in a relational database system. Examples provided for this particular 4GL could support the learner when generating the appropriate server-side scripting code needed for this performance outcome, depending on the depth of coverage chosen.

Unit 7 – PO3: Implement object oriented applications to a professional standard
Program connectivity with relational database systems is examined in this unit's third performance outcome, albeit through object oriented programming languages such as Java. Although the syntax will be markedly different, the underlying SQL that is used to query the database will be very similar and will provide the learner with additional resources they can use to tackle the problem.

P14: Identify four common client-side threats
P15: Identify four common server-side threats

Unit 1 – AO3: Understand the software requirements of a computer system
Many of the common client-side threats (eg tracking cookies) can be detected by the type of security software that is introduced in this unit's third assessment outcome. Antivirus and anti-spyware software are particularly prevalent in the identification of common client-side threats and may provide the learner with additional information about the nature and spread of the risk.

Useful links and resources

Books

- McGrath M, *JavaScript in easy steps*, 5th revised edition, ISBN-10 1840785705, ISBN-13 978-1840785708, In Easy Steps Limited (2013).
- McGrath M, *PHP and MySQL in easy steps*, ISBN-10 1840785373, ISBN-13 978-1840785371, In Easy Steps Limited (2012).
- McGrath M, *SQL in easy steps*, 3rd edition, ISBN-10 1840785438, ISBN-13 978-1840785432, In Easy Steps Limited (2012).
- Powell TA, HTML and CSS: the complete reference, ISBN-10 0071496297, ISBN-13 978-0071496292, McGraw-Hill Osborne (2009).
- Spaanjaars I, *Beginning ASP.Net 4.5 in C# and VB*, ISBN-10 1118311809, ISBN-13 978-1118311806, John Wiley and Sons, (2012).
- Valade J, *PHP and MySQL for dummies*, 4th edition, ISBN-10 0470527587, ISBN-13 978-0470527580, John Wiley and Sons (2009).

Websites

- Code Academy: codecademy.com
- JSFiddle: jsfiddle.net
- MSDN ASP .Net: msdn.microsoft.com/en-us/library/aa286485.aspx
- MySQL: mysql.com
- PHP: hypertext preprocessor: php.net
- Sitepoint: reference.sitepoint.com
- W3C learn HTML: w3.org/wiki/HTML
- W3C markup validation service: validator.w3.org
- W3Schools online web tutorials: w3schools.com
- Webplatform.org: webplatform.org

12.4 Unit 4: Mobile applications programming

Title	Mobile applications programming
Unit number	Y/507/6486
Unit assessment type	Centre assessed and externally quality assured
Recommended assessment method	Practical assignment This is the preferred assessment method for this unit. A centre may choose an alternative method of assessment, but will be asked to justify as part of the quality assurance process.
Guided learning hours	90
Transferable skill(s) contextualised within this unit	N/A
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms that offer IDEs suited to learning and developing event mobile application development techniques. These include, but are not limited to: Microsoft's Visual Basic.Net; Apple's xCode for iOS; IBM's Java using NetBeans IDE; Eclipse (with Android Development Tools (ADT) plug-in) etc. Alternative development platforms could be considered, eg MIT App Inventor for Android, although this may limit some scope for learner development and achievement. In addition some aspects of the unit will require user rights on the development and target platforms to compile programs and install executable code. Learners should also have access to suitable offline and online learning material, manuals, help sheets and coded examples in order to encourage self-sufficiency.

Synoptic assessment within this unit	IT: Scripting and app programming linked to Units 1, 2 and 3. IT: Programming linked to Units 1, 2, 3, 6 and 7. Unit 1 provides the learner with a solid grounding in the different types of software and hardware that are likely to form part of a mobile application solution. This unit also introduces the concept of programming languages and cross-compilation which, when combined, enable the creation of mobile application solutions. Unit 2 introduces many of the concepts used in mobile applications programming, from an overview of the different types of programming languages available to the key features common to them all and the intricate differences in usage and concept that set them apart. Unit 3 has strong links to this unit as many mobile applications are hybridised native and web-based solutions. This unit provides the background on the myriad of technologies that form a modern web-based solution. Units 6 and 7 focus on event driven programming and object oriented programming. Despite their differences these two units collectively form a cohesive approach that reinforces the learner's problem-solving skills and awareness of common elements such as code syntax, functions and debugging that are crucial to leveraging mobile applications solutions effectively. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

To equip learners with the necessary knowledge and practical ability to build high-quality coded applications for popular mobile devices and meet user expectations through their applied understanding of the design, development and deployment process.

Unit introduction

The rise of mobile technology, particularly mobile telephones and tablets, has reinvigorated and transformed the software development market, with a myriad of smaller applications written on a daily basis that inform, educate or simply entertain.

The development of mobile applications can be achieved in a number of ways, but principally it involves cross-platform development – designing an application on one computer platform and publishing it for use on another. The concepts involved can be challenging, but the tools and support available are often free and easy to access.

In this unit learners will discover how to design and create a mobile application, typically through a hybrid of basic event driven and object oriented techniques, and then publish on their target mobile platform. Focus is also placed on the interface techniques of the target device (including voice recognition and gesture control) and on the correct use of third party libraries that assist development and making money in the worldwide application marketplace.

Unit content

The key features of mobile application development	
Mobile platforms	• Commercial deployment platforms, eg iOS (iPhone, iPad, iPod touch), Android, BlackBerry, Windows Phone, Symbian, Python etc. • Free deployment platforms, eg Java ME, Web browser etc. • Special hardware features, eg multi-touch screen, camera, microphone, accelerometer, finger print sensor, tilt sensor etc. • Special software, eg voice recognition, gesture recognition etc. • Screen resolution (modes) and orientation. • PPI (pixels per inch), DPI (dots per inch). • Colour depth.
Mobile application programming technologies	• Programming languages, eg Objective-C, Java, Microsoft C#, C, C++, Objective Pascal, Lua, Visual Basic.net etc. • Scripting languages, eg JavaScript etc. • Style-sheet languages, eg CSS3. • Markup languages eg HTML5. • SDK (Software Development Kit), eg Android SDK. • API (Application Programmers Interface), eg. • Configuration, eg XML (eXtensible Markup Language).
Types of mobile application	• Native applications. • Mobile web applications. • Hybrid applications.
Categories of mobile application	• Computer games. • Media content players, eg BBC iPlayer, YouTube etc. • Productivity applications, eg Office suites etc. • Utilities, eg calculator, calendar etc. • Social networking, eg Facebook, Twitter etc. • Lifestyle, eg cooking recipes, price comparison, review guides etc.
Development tools	• Commercial, eg Adobe Flash Builder, Flash Professional. • Free, eg Adobe AIR SDK, Eclipse, MIT App Inventor for Android, NetBeans, Text editor, XCode etc. • Online interactive, eg MIT App Inventor for Android. • Plug-ins, eg Android Development Tools (ADT) for Eclipse. • Emulators and virtual devices.
Publishing methods and issues	• Installer packages, eg APK, JAD/JAR. • OTA (over the air). • Digital distribution platform, eg iOS App Store, Google Play Store, BlackBerry World, Windows Phone Store etc. • Licensing, eg free, commercial, GNU GPL etc. • Security bypassing, eg sideloading, jailbreaking. • Digital signing, eg developer key, vendor lock-in.

The key features and functions of mobile application programming languages	
Mobile application development workflow	• Setup. • Development. • Debugging and Testing. • Publishing.
Event handling features	• Event loop. • Event trigger. • Event queue. • Event dispatcher. • Event handlers and listeners. • Delegates and interfaces.

Event triggers	• User events:
	  • touch screen, eg click, drag etc
	  • physical button presses
	  • gesture-based controls, eg pinch, swipe, multi-finger swipes etc
	  • motion-based controls, eg rotation, shake.
	• Time-based events:
	  • real time clock (RTC)
	  • timers and interval measurement, eg milliseconds
	  • temporal, eg loading of forms, application exit.
	• Software exceptions, eg OnError.

Classes and objects	• Classes. • Object. • Methods and overriding. • Properties.
User interface components	• Screen layouts and views. • Pre-defined chrome, eg navigation bar, status bar etc. • Common control widgets, eg Textbox, Button, Label, Listbox, Checkbox, Radio button etc. • Common containers, eg Form, Panel, Tabs. • Common dialogs, eg File, Print, Colour, Font etc. • Properties, eg enable, visible, colour, font, size, position, values, etc • Methods, eg clear, refresh/repaint etc. • Custom controls (classes, methods, properties). • Third party controls.

Event handlers	• Common event handlers, eg on Load, Click, got Focus, lost Focus, Key down, Key press, Text changed etc.
	• Custom event handlers:
	  • adding event handler
	  • raising event handler
	  • removing event handler.

The key features and functions of mobile application programming languages	
Identifiers	• Variables and constants (programming and system defined). • Data types, eg integer, decimal, Boolean, string, character, date/time, currency etc. • RAM storage allocation. • Value range. • Declaration and initialisation. • Scope and visibility, eg public, private, local (module-level), global, namespaces etc.
Data structures	• Array, eg one-dimensional and multi-dimensional. • Lists, stacks, queues. • Text files, XML files. • Relational databases, tables, records and fields, eg SQLite.
Operators	• Arithmetic. • Relational. • Logical. • Concatenation.
Common language constructs	• Sequence. • Selection, eg if...else...endif, nested if, switch/case. • Iteration, eg pre-conditioned, post-conditioned, for...each mechanism.

Functions	• Programmer-defined functions:
	  • naming and declaration
	  • formal and actual parameters
	  • function calls
	  • return type.
	• Common library functions:
	  • input and output
	  • arithmetic
	  • string
	  • date
	  • file.
	• Third party library functions.

Demonstrating the ability to design mobile applications	
Design tools	• Storyboards. • Metrics and grids, physical size and screen density (DPI). • User interface design templates. • User Interface design software. • Action charts.
Style considerations	• Platform style guides, eg Apple iOS. • Iconography. • Typography. • Themes and colour. • Touch feedback. • Layout and consistency. • Personal or organisational branding.

Accessibility considerations	• Accessibility concerns:
	  • vision
	  • hearing
	  • physical and motor skills
	  • learning and literacy.
	• Accessibility adaptations:
	  • captioning
	  • voiceover
	  • speech
	  • guided access
	  • high contrast colour.

Selecting appropriate controls	• Identifying available controls and objects. • Mapping storyboard to controls. • Identification of required properties and methods.
Selecting appropriate events	• Identifying available event handlers. • Mapping storyboard to events.

Demonstrating the proficient use of mobile development tools	
Installing and configuring mobile development tools	• Download developer packages. • Install developer packages. • Configure developer packages. • Test developer packages.

Demonstrating the proficient use of mobile development tools	

Using the features of an integrated development environment	• Integrated development environment (IDE):
	  • workspaces.
	• Customisation eg layout, docking
	• Restoring defaults:
	  • projects.
	• Project types eg form-based, console.
	• Creating a new project.
	• Navigation and storage.
	• Adding files to a project.
	• Resources, eg images, sound files etc.
	• Closing a project.
	• Moving a project safely.
	• Using project templates.
	• File operations:
	  • load
	  • save.
	• Coding conventions and standards.
	• Compilation/building.
	• Executables (debug, release).
	• Cleaning intermediary files.

Deploying a working mobile application using cross platform development	
Cross platform development	• Features, eg cross-compilation. • Challenges, eg connectivity, tools, different architecture, platform neutrality and platform dependence.
Creating a mobile application	• Coding conventions and standards. • Cross compilation/building. • Executables or portable code (eg Java bytecode). • Cleaning intermediary files.

Debugging a mobile application	• Compilation/build problems:
	  • Nnotices
	  • deprecations
	  • warnings
	  • errors.
	• Debugging tools:
	  • watch
	  • breakpoints, eg setting, clearing
	  • tracing and step over
	  • output and immediate window
	  • log files
	  • virtual devices and emulators
	  • transfer to mobile device.

Deploying a working mobile application using cross platform development	

Documenting a mobile application	• Internal documentation:
	  • onscreen help eg user prompts, tool tips etc
	  • identifier naming conventions.
	• Potential benefits, eg metadata, consistency, clarity.
	• Common formats eg upper camel case, lower camel case, Hungarian notation etc.
	• Code layout and indentation.
	• Comments.
	• Best practice and purpose.
	• XML and HTML documentation.
	• User documentation (paper and online).
	• Technical documentation.
	• Digital distribution application description.

Deploying a mobile application	• Application properties:
	  • icon
	  • start-up form
	  • application versioning, eg major, minor
	  • digital signatures.
	• Application setting persistency:
	  • configuration.
	• Separate configuration files, eg .ini, .conf text files, XML etc.
	• Operating System database, eg Windows Registry.
	• Packaging dependencies, eg required external components.
	• Installation tools.
	• End user license agreement (EULA).
	• Digital distribution methods.

Reviewing a mobile application	• Approval process, eg Apple SDK agreement. • Client interim reviews, eg iterative process. • Final review against client specification. • Customer reviews.
Maintaining a mobile application	• Client and customer feedback process. • Bug reporting. • Update strategies and frequency.

Performance outcomes

On successful completion of this unit learners will be able to:

Performance outcome 1:	Understand the key features of mobile application development.
Performance outcome 2:	Apply the key features and functions of mobile application programming languages.
Performance outcome 3:	Demonstrate the ability to design mobile applications.
Performance outcome 4:	Demonstrate the proficient use of mobile development tools.
Performance outcome 5:	Create and deploy a working mobile application using cross platform development.

Grading criteria

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO1: Understand the key features of mobile application development	**P1** Describe **six** features of mobile platforms and **six** programming technologies.	**M1** Compare and contrast the different types and categories of mobile application.	**D1** Assess the different methods of mobile application distribution.
	P2 Describe **five** development tools used in mobile applications programming.		
PO2: Apply the key features and functions of mobile application programming languages	**P3** Provide an annotated diagram that represents the mobile application development workflow.		
	P4 Demonstrate the correct usage of different types of interface elements, event handling features and triggers.	**M2** Compare and contrast the potential use of **five** different user interface elements to meet a defined user need.	
	P5 Demonstrate the correct usage of basic data types, operators, constructs and functions to meet a defined user need.	**M3** Develop a programmer-defined function or class to meet a defined user need.	**D2** Interpret the correct usage of a complex third party function or class in order to deploy correctly in a mobile application.
	P6 Demonstrate the implementation of simple identifiers and data structures to meet a defined user need.	**M4** Prove the importance of accessing database structures to solve set problems.	

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO3: Demonstrate the ability to design mobile applications	**P7** Select an appropriate design tool and plan a mobile application to meet a defined user need.	**M5** Interpret a design correctly to deduce the user interface components and event handlers required to meet a user defined need.	**D3** Justify the selection of accessibility adaptations made in the solution to cater for potential user concerns.
	P8 Select appropriate controls and events to meet a defined user need.	**M6** Apply suitable style considerations to a mobile application, using guidelines for the target platform.	
PO4: Demonstrate the proficient use of mobile development tools	**P9** Carry out mobile application development using a range of IDE facilities with good working practices.	**M7** Demonstrate the ability to install and configure mobile development tools correctly.	
PO5: Create and deploy a working mobile application using cross platform development	**P10** Explain the challenges to cross platform development and the role of the cross compiler.		
	P11 Select and use appropriate features and functions to create a mobile application to meet a defined user need using traditional coding standards and conventions.		
	P12 Use appropriate debugging tools to identify programming faults in a selected application.		

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
	P13 Document a mobile application.		
	P14 Deploy a mobile application to a virtual device or emulator for debug purposes.	**M8** Review a mobile application against the client specification and customer feedback.	**D4** Evaluate the mobile application using client feedback to identify an effective update and maintenance strategy.
	P15 Deploy a mobile application to a mobile device from a standard computer platform.		

Assessment amplification

This section provides amplification of what is specifically required or exemplification of the responses learners are expected to provide.

There is no mandatory division of grading criteria although, as can be seen from the Grading criteria grid, some are naturally linked either by learning outcome or by developmental theme. Connections between the various pass, merit and distinction criteria can therefore suggest logical assessment grouping but do not necessarily have to be followed.

Finding a single real world scenario that can cover all criteria is difficult, but not impossible. The key to assessment in this unit, as in others, is to make assessment a natural outcome of the learning experience, with skills being assessed at a level appropriate to the learner's level of experience. The nature of the assessment technique is tutor-dependent, but some outcomes in this unit may only be achieved through generating working program code to a good standard.

Integrative assessments that connect this unit to others via connected tasks are also encouraged and it is possible that evidence produced during work for this unit may be used to provide evidence for criteria in other units.

Evidence for this unit could be generated through web pages, leaflets, electronic slideshows, wikis, presentations, podcasts or a formal report. Learners will work with a substantive real world, complex task, that is solved by using a mobile application that is designed and implemented by learners.
A learner's ability to install and configure development software in their target environment should include a comprehensive demonstration of the use of its Integrated Development Environment (IDE) to complete the tasks associated with normal application workflow. This could be evidenced through a combination of photographic or video evidence, presentation, witness or observation statements.

For **P1** learners should demonstrate an understanding and appreciation of the different commercial (and free) platforms available.

For **D1** learners should use examples to evaluate the different methods of distribution for mobile application; this could include (but is not limited to) metrics such as speed, cost, ease-of-deployment, requirement for digital signage or vendor approval.

For **M2** learners should consider the relevant merits of using different interface elements for a particular mobile application.

For **D2** learners should consider what qualifies as a complex function and it is suggested that the processing undertaken by the function is multi-stage, with at least two formal parameters and a return type being used.

For **P5** and **M4** the suggested data structure implementations should include an event driven stack (last in, first out – LIFO) or queue (first in, first out – FIFO). Relational databases systems such as SQLite may provide suitable sources of data for **M4**.

For **P7**, **M5** and **D3** learners will demonstrate their understanding and use of a particular design tool (eg storyboard or user interface templates) to solve this complex task. **M5** requires learners to interpret the design created in P7 to identify the appropriate user interface elements and event and handlers that are needed; an action chart is an ideal method for evidencing this.

For **M6** learners are required to demonstrate that they are aware of platform considerations in terms of meeting appropriate style guides; these aspects should be identified and applied in their design.

For **D3** learners focus on including accessibility features at the heart of the design in order to address a particular user need; learners must demonstrate they understand the concern and can incorporate appropriately judged adaption to their solution to deal with the accessibility issue.

For **P12** learners should use good coding standards and conventions in the creation of their code. This simply means that functional code isn't sufficient: it should be written to a professional standard using meaningfully named identifiers, good layout and indentation, and be appropriately documented.

For **P13** learners should demonstrate their debug skills to identify and remove warnings and errors from their code (semantic or syntactic); this should be documented to reflect good practice and could form part of P14's technical and user documentation which could be presented in printed format or electronically.

Employer engagement guidance

If learners are in the workplace then the centre could ask the employer whether there are any suitable projects that learners could work on as part of the team. It would be helpful for the employer to be made aware of the sort of skills that learners have to practice.

Delivery guidance

It is recommended that this unit is taught after the conclusion of Unit 2: Computer programming because that unit introduces (in overview only) the concept of generalised programming techniques that support the demands of learning mobile application development. In addition, many elements have parallels in both the Object Oriented and Event Driven programming units; indeed, many of their respective programming languages (eg Visual Basic.Net, Objective-C, Java etc) can form a critical foundation.

Although it is suggested that the content is delivered to follow the order of the learning outcomes in this unit specification, it is not the only sequence that could be used; tutors are encouraged to consider the holistic nature of the learner's programme and the scheme itself.

Learners must have access to the required hardware and software facilities in order to generate evidence for all of the grading criteria listed. In the case of mobile applications, it is perfectly possible

that a learner's device may be used if they accept the risks associated with running (in some cases) unsanctioned code. As such, if centres cannot guarantee these resources, the unit should not be attempted.

It is suggested that the delivery of tools and techniques follows the pattern of a succession of small, but well chosen, exercises. These form a developmental toolkit that learners may use to build more complex solutions. This type of approach gently builds confidence and self-sufficiency in learners and is critical in promoting good problem-solving skills. The deployment target of solutions may be mobile platforms such as smartphones, tablets and personal data assistants. No particular language, development platform or deployment platform is mandated; centres may focus on one or more languages during delivery, if they choose, in order to aid comparison and contrast.

Performance outcome 1

Learners should begin their studies by developing their knowledge and understanding of the key principles of event mobile application development, essentially breaking them down into platforms, technologies, tools, applications and methods of deployment. Group work could explore these concepts and learners could present their research, demonstrating an understanding and appreciation of the different commercial (and free) platforms available. A compare-and-contrast exercise would also be useful, particularly considering (but not limited to) metrics such as speed, cost, ease of deployment, requirement for digital signage or vendor approval.

Learners should also consider emerging technologies that might not have been available at the time of writing this guidance.

Performance outcome 2

Learners should become familiar with event driven tools and techniques. This could easily be achieved through a suite of small development tasks, each focusing on a different aspect. Examples may use one technology or a suite of technologies, depending on the approach of the tutor.

Learners will subsequently need to demonstrate a range of different interface elements, events, triggers and handlers, considering the relevant merits of using different interface elements for a particular mobile application.

Learners should have multiple opportunities to create program code that encompasses basic data types, constructs and functions. They should develop their own programmer-defined functions or classes and become familiar with the use of complex third party functions or classes correctly in order to solve set problems.

Furthermore learners should consider the suitability of different data structure implementations including an event driven stack (last in, first out – LIFO) or queue (first in, first out – FIFO). Relational database systems such as SQLite may provide suitable sources of data for appropriate apps.

Learning outcomes 3, 4 and 5 lend themselves to a substantive real world, complex investigation that is solved by using a mobile application designed and implemented by learners.

Performance outcome 3

Undertaking mobile applications programming is enhanced by competent design. Therefore, this outcome allows learners to study particular design tools (eg storyboard or user interface templates) that they will use to underpin solutions. Learners will need to provide sufficient detail to enable a programmer to interpret the design, which will mean that the following should be identified:

- interface elements
- event handlers
- action charts.

Learners must be able to select and justify style choices and would benefit from investigating a range of mobile applications across different platforms.

They should consider accessibility and how the app meets the needs of the target user group. To ensure that they grasp the complexities of different user groups, they should investigate a range of apps for different groups (young, middle-aged, older; male, female etc).

Performance outcome 4

This outcome enables learners to install and configure development software in their target environment and to develop the skills required to demonstrate the comprehensive use of its Integrated Development Environment (IDE) to complete the tasks associated with normal application workflow.

Performance outcome 5

Learners should be able to independently design, implement and deploy a working solution. In particular learners would benefit from creating applications across multiple platforms, so that they can understand the challenges and considerations when working in a cross-platform environment and, in particular, the use of a cross-compiler.

Learners should ensure at all times, even when practicing their coding techniques, that good coding standards and conventions are upheld in the creation of their code. This simply means that functional code isn't sufficient: it should be written to a professional standard using meaningfully named identifiers, good layout and indentation, and be appropriately documented.

Skills in testing and debugging, identifying and removing warnings and errors from code (semantic or syntactic) should be developed. This should also be documented to reflect good practice and often forms part of the technical and/or user documentation for the application.

To complete the unit, learners should be assessed in a way that reflects the linked sequence of the learner deploying their solution (initially to a virtual device or emulator), reviewing it and evaluating it successfully. Real world scenarios, particularly with an external client, are ideal for capturing impartial critical feedback, so opportunities to work with employers or commercial outlets would be recommended.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

P1: Describe six features of mobile platforms and six programming technologies

Unit 1 – AO1: Understand the different types of computer
Many of the features associated with mobile platforms are introduced as part of this unit's examination of computer hardware (internal and external), including specialised hardware and software that offer more specialised input (eg biometrically via a user's voice and fingerprint). Therefore, learners should be familiar with such technologies before they consider their practical use in mobile applications.

Unit 1 – AO5: Demonstrate how computers process user requirements
Many different programming technologies can be used to create mobile applications. Unit 1 introduces many of these languages and should provide a fundamental introduction for the learner, easing their re-introduction in this unit (which is likely to follow).

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
Unit 1 introduces many of the languages used to create mobile applications. Unit 2 expands this coverage and also examines common uses of these languages including mobile platforms and cross platform development.

Unit 3 – PO1: Understand the key features of website technologies

Unit 3 – PO2: Demonstrate key features and functions of a markup language

Unit 3 – PO3: Demonstrate key features of a style sheet language

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Many mobile applications are categorised as being hybridised; this means that they are effectively a combination of native and web-based applications. The majority of Unit 3 introduces the key technologies used to create web-based applications and consequently should provide a solid reference point for learners wishing to discuss this aspect of mobile applications programming.

P3: Provide an annotated diagram that represents the mobile application development workflow

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
The four stage mobile application workflow is in reality a specialised version of the better known (but generic) software design lifecycle that is introduced in Unit 2's first assessment outcome. It is likely that learners will be familiar with the stages of the full software design lifecycle before tackling this workflow, and this may help them to differentiate and sequence the four mobile applications workflow stages much faster.

P4: Demonstrate the correct usage of different types of interface elements, event handling features and triggers

Unit 6 – PO1: Understand the key features of event driven programming languages

Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 provides a firm foundation on the event driven features that most mobile application programming languages use.

These linked performance outcomes should provide the learner with a checklist of the key features to consider and help them differentiate the various event triggers, controls and handlers that are available in the target mobile application programming language.

P5: Demonstrate the correct usage of basic data types, operators, constructs and functions to meet a defined user need
P6: Demonstrate the implementation of simple identifiers and data structures to meet a defined user need

Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 focuses on event driven programming, a paradigm that is typically at the heart of most mobile application programming. As with most programming languages, an event driven language also makes extensive use of basic data types, structures, operators, constructs and functions. Learners should be able to draw parallels between the similar concepts demonstrated in the two units and use them to reinforce their understanding.

P7: Select an appropriate design tool and plan a mobile application to meet a defined user need

Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2's second assessment outcome introduces the learner to many different design techniques. Some of these, such as storyboards, flowcharts, action tables etc, are more appropriate than others for designing mobile applications. Providing the learner with a wide range of design tools (and their associated features) should offer plenty of insight for selecting a preferred method.

Unit 6 – PO3: Demonstrate the ability to design event driven applications
Although there are often unique challenges to designing a mobile application, many of the tools (eg action charts and storyboards) used to plan and design can also be used to design generic event driven applications. Using these familiar tools for potentially different target platforms should help the learner to make informed choices when selecting their preferred design tools and further hone their abilities when using them.

P8: Select appropriate controls and events to meet a defined user need

Unit 6 – PO1: Understand the key features of event driven programming languages

Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 provides a firm foundation on the event driven features that most mobile application programming languages use.

These linked performance outcomes should provide the learner with a checklist of the key features to consider and help them differentiate the various event triggers, controls and handlers that are available in the target mobile application programming language.

P9: Carry out mobile application development using a range of IDE facilities with good working practices

Unit 6 – PO4: Implement event driven applications to a professional standard
Integrated Development Environments (IDEs) are typically at the heart of any modern programming experience. This performance outcome focuses on providing the learner with an experience of using an IDE to build an application through an event driven programming language. Many of the skills learnt in this performance outcome should be easily transferred to creating mobile applications, including the application of good working practices (eg coding conventions and standards, project management etc).

P10: Explain the challenges to cross platform development and the role of the cross compiler

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
The concept of cross platform development is discussed in Unit 2's first assessment outcome and, paired with mobile platforms, necessitates the learner having an appreciation of the role of a cross compiler.

P11: Select and use appropriate features and functions to create a mobile application to meet a defined user need using traditional coding standards and conventions

Unit 6 – PO1: Understand the key features of event driven programming languages

Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 provides a firm foundation on the event driven features that most mobile application programming languages use.

These linked performance outcomes should provide the learner with a checklist of the key features to consider and help them differentiate the various event triggers, controls and handlers that are available in the target mobile application programming language.

P12: Use appropriate debugging tools to identify programming faults in a selected application

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This assessment outcome introduces the concept of testing and debugging and examines the use of debug facilities such as watches, traces, breakpoints and code inspection. The learner should be able to apply tools and techniques such as these (present in a modern web browser) to resolve errors in their mobile applications.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
These performance outcomes focus on debugging applications albeit as part of an interactive website, through the use of either client-side (JavaScript) or server-side (eg PHP) scripting tools. The learner should be able to apply similar tools and techniques learnt here to resolve similar syntax and semantic errors found in their mobile applications.

Unit 6 – PO4: Implement event driven applications to a professional standard
This performance outcome also focuses on debugging applications, albeit in an event driven programming environment. The learner should be able to apply similar tools and techniques learnt here to resolve similar syntax and semantic errors found in their mobile applications.

P13: Document a mobile application

Unit 6 – PO4: Implement event driven applications to a professional standard

Unit 7 – PO5: Understand how to produce documentation
Documentation of an application is a recurring theme for most programming paradigms, and event driven and object oriented programming proves no exception. These linked performance outcomes mirror the content and delivery for mobile applications and may help to reinforce the learner's understanding of the required documents, their content and standards.

P14: Deploy a mobile application to a virtual device or emulator for debug purposes

Unit 6 – PO4: Implement event driven applications to a professional standard
Many event driven applications have a similar deployment path to mobile applications. Although it is less common, learners may have previous experience with deploying applications to virtual devices or emulators and this could ensure that the process is achieved confidently and correctly.

P15: Deploy a mobile application to a mobile device from a standard computer platform

Unit 6 – PO4: Implement event driven applications to a professional standard
Many event driven applications have a similar deployment path to mobile applications. Some event driven programming IDEs permit deployment to mobile devices thereby providing learners with previous experience; this should ensure that the process is achieved confidently and correctly.

Useful links and resources

Books

- Bennett G, Fisher M and Lees B, *Objective-C for absolute beginners: iPhone, iPad and Mac programming made easy*, ISBN-10 1430236531, ISBN-13 978-1430236535, Apress (2011).
- Burd B, *Java programming for Android developers for dummies*, ISBN-10 1118504380, ISBN-13 978-1118504383, John Wiley and Sons, Hoboken, NJ (2013).
- Felker D, *Android application development for dummies*, ISBN-10 047077018X, ISBN-13 978-0470770184, John Wiley and Sons, Hoboken, NJ (2010).
- Goldstein N, *iPhone application development for dummies*, ISBN-10 0470487372, ISBN-13 978-0470487372, John Wiley and Sons, Hoboken, NJ (2009).
- Ray J and Johnson S, *Sams teach yourself iPhone application development in 24 hours*, ISBN-10 0672330849, ISBN-13 978-0672330841, Sams Publishing (2009).

Websites

- Apple iOS Dev Center: developer.apple.com/devcenter/ios/index.action
- iOS Dev Tools: ios.devtools.me
- Official site for Android developers: developer.android.com/index.html
- Sketch[3]: bohemiancoding.com/sketch/features
- The iOS design cheat sheet: ivomynttinen.com/blog/the-ios-7-design-cheat-sheet

12.5 Unit 5: Mathematics for programmers

Title	Mathematics for programmers
Unit number	Y/507/6469
Assessment	Externally assessed
Guided learning hours	90
Transferable skill(s) contextualised within this unit	N/A
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms that offer access to the low levels of the operating system, eg debug or monitor facilities where RAM address content may be examined. This is often achievable through a virtualised platform. In addition, pathway-relevant software should be available that demonstrates the use of different number systems, eg networking software for subnet masks, graphics software for RGB colour codes, email clients for multi-purpose internet mail extensions (MIME) base 64 attachments etc. Optional resources are indicated in each assessment outcome that may enhance learners' experience in this unit. Learners should also have access to suitable offline and online learning material, manuals, help sheets and coded examples in order to encourage self-sufficiency.
Synoptic assessment within this unit	IT: Programming linked to Units 1, 2, 3, 6, 7 and 8. Draws on the hardware components of a computer system (specifically the CPU), underlying principles of data concepts and how user requirements are processed, as discussed throughout Unit 1. Unit 2 introduces the concepts of programming and makes practical use of concepts such as Boolean logic, different number bases, algorithms and matrices. These are then amplified and developed in Units 3, 6, 7 and 8 where demonstrations of many of the mathematic topics found in this unit can be practically applied. Units 6, 7 and 8 build on the basic programming concepts introduced in Unit 2 and expand the learner's technical skills by demonstrating the power and flexibility offered by different programming paradigms. Each programming language studied and every solution written offers opportunities to create ever more complex solutions that require in-depth testing and quantifiable analysis of actual results. The industrial project (Unit 8) relies on planning a solution to a set problem using an organised methodology. Problems are often provided in terms of quantifiable data that needs gathering and analysing, a key part of this unit's goals. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

This unit will provide the learner with the necessary knowledge to understand the mathematical concepts that enable computers to store, process, communicate and transmit data. The learner will also develop a range of skills required to recognise its practical application in everyday computing, process numerical data correctly and construct the appropriate logical and arithmetic expressions to solve common problems.

For programmers it introduces many of the concepts that enable them to abstract real world situations into the calculations and functions that can be easily translated to effective program code.

Unit introduction

Mathematics is at the very heart of any computer system; from storing its data to displaying complex imagery on screen or simply loading a web page – mathematics is the 'magic' that makes it happen.

In this unit learners will gain an understanding of how computers internally represent data and make decisions, along with applying common mathematical techniques for creating algorithms that solve set problems, whether these are presented in terms of system, networking or programming situations.

The unit begins with a sound foundation in the number systems that are prevalent in the operation of computer hardware, software and systems. It then examines the underlying logic used in the digital circuits that power the micro-processors inside the computer – logic that is used every day by operating systems, networks and programmers alike.

Once the fundamental building blocks have been locked into place, it is possible to investigate the practical use of mathematics in computing, focussing on popular applications such as number series, recursion, algebra, functions and matrix manipulation. Many of these can then be implemented using suitable programming languages.

Unit content

Common number systems	

Number systems fundamentals	• Concept of a positional numeral system, eg positional weights.
	• Base or radix terms; correct mathematic notational use, eg 12_2.
	• Base notation in computing, eg # or 0x prefix for hexadecimal.
	• Common computer bases:
	  • denary (Base 10)
	  • binary (Base 2)
	  • octal (Base 8)
	  • hexadecimal (Base 16)
	  • larger, eg multipurpose internet mail extensions (MIME) Base 64.

Base conversion	• Base conversion methods:
	  • denary to binary and vice versa
	  • binary to hexadecimal and vice versa
	  • denary to hexadecimal and vice versa.

Common number systems	

Fractional and negative integers in binary	• Binary integer signing techniques:
	  • unsigned
	  • sign and magnitude
	  • one's complement
	  • two's complement.
	• Representing fractional values in:
	  • denary
	  • binary
	  • fixed number storage in binary
	  • floating point number storage notation in binary, ie significand/ mantissa, exponent etc
	  • single and double precision.

Base arithmetic	• Performing basic arithmetic operations in other bases; binary, hex etc:
	  • addition
	  • subtraction
	  • multiplication
	  • division.

Computer use of bases	• Common use of different bases in modern computing, eg:
	  • binary, eg:
	  • internal data representation
	  • Boolean logic and flags
	  • internet protocol subnet masks
	  • Classless Inter Domain Routing (CIDR) notation
	  • character representation, eg 7 or 8 bit American Standard Code for Information Interchange (ASCII) characters etc.
	  • octal, eg:
	  • Unix/Linux CHMOD file permissions
	  • escape strings, UTF-8 etc.
	  • Hexadecimal, eg:
	  • error codes
	  • network (Media Access Control) MAC physical addresses
	  • memory (eg RAM) addresses
	  • 8-bit per channel RGB (Red Green Blue) colour codes, eg #FF0000 = red.
	  • internet Uniform Resource Identifier (URI).
	• Binary vs Gray code, rationale and use.

Computer logic	
Foundation of computer logic	• Boole, Boolean logic/algebra (true, false; 1,0).

Basic logical operators	• Processing and outputs of basic logical operations:
	  • AND (conjunction)
	  • OR (disjunction)
	  • NOT (negation).
	• Common logical operator symbols.

Composition and derived operators	• Processing and outputs of derived logical operations:
	  • Exclusive or (EOR or XOR)
	  • Negative or (NOR)
	  • Negative and (NAND)
	  • Exclusive Negative or (XNOR).
	• Common logical operator symbols.

Visual representation of logical operations	• Representing logic in diagrammatic form:
	  • truth tables for each using x and y as inputs.
	  • digital logic gate diagrams for circuits
	  • Venn diagrams
	  • creating circuits using only one type of logic gate, eg NOR or NAND.

Working with Boolean algebra	• Notation and symbols
	• Interpreting simple equations, eg $(x.y) + x$
	• Common techniques used to simplify complex Boolean expressions, (eg):
	  • Boolean identities
	  • De Morgan's Laws
	  • Karnaugh maps
	  • truth tables.

Sets, sequences, series, probability and recursion	

Sequence and series	• Sequences:
	  • type, eg finite or infinite
	  • notation, n^{th} term
	  • rules and finding rules
	  • term and term number
	  • arithmetic and geometric sequence.
	• Series:
	  • definition, contrast to a sequence
	  • use of sigma (Σ) notation.

Sets, sequences, series, probability and recursion	

Probability	• Probability terminology and usage, eg:
	  • probability definition
	  • probability line
	  • experiment or trial
	  • sample space and sample point
	  • events and event types (dependent, independent, mutually exclusive).
	• Space diagrams eg 'double' values when two dice are thrown.
	• Visualising events, eg using Venn diagrams (mutually exclusive events); tree diagrams (dependent events).

Recursion	• Series eg Fibonacci, factorial etc
	• Rules of recursion:
	  • have a base case
	  • state change toward base case,
	  • must call itself.
	• Common recursive algorithms used in computing, eg:
	  • factorial
	  • quicksort
	  • binary search
	  • directory traversal
	  • Sierpinski triangle.
	• Instances where simple iterative techniques may be more efficient.

Apply arithmetic expressions to abstract real world ideas	

Arithmetic expressions	• Numbers and operations (operators)
	• Vinculum
	• Brackets (parenthesis)
	• Orders (exponents or square roots)
	• Division and Multiplication
	• Addition and Subtraction
	• Order of operations (operator precedence)
	• Use of mnemonics, eg BODMAS, PEMDAS etc
	• Notation (eg):
	  • common notation, ie infix notation
	  • Polish notation, ie prefix notation
	  • reverse Polish notation (RPN), ie postfix notation
	  • rationale for Polish and RPN, eg speed, computer-based interpretation
	  • interpretation and writing of postfix and prefix expressions.

Apply arithmetic expressions to abstract real world ideas	
Algebra	• Purpose of algebra and abstraction. • Use of letters in algebra, including Greek symbols. • Variables. • Replacement of multiplication symbol, 3 . x or 3(x) or 3x • Coordinate plane, axes and origin, Cartesian coordinates. • Writing expressions to represent real world ideas and situations. • Interpreting expressions. • Linear expressions. • Algebraic expressions. • Evaluating expressions. • One variable. • Two variables. • Algebraic expressions with fractions. • Simplification using combination of like terms (including negative coefficients). • Examples of differentiation of variables, expressions and equations. • Dependent and independent variables.
Functions	• Purpose. • Range, eg non-mathematical. • Definition. • Input. • Output. • Function notation, eg f, x etc. • Relationship to equation notation, eg $y = x + 1$ or $y = f(x) = x + 1$ • Difference between equation and function, ie inability to associate two outputs with one input. • Evaluation of a function when given its formula. • Evaluation of a function when given its graph. • Creating a function from an equation. • Interpreting an expression with function notation. • Composition. • Recursive function.

Apply matrix methods to solve problems	

Matrices	• Matrices to represent ordered data.
	• Matrix terminology, eg rows, columns, elements or entries, subscripts, ‘m-by-n’ size definitions.
	• Forms of matrix:
	  • row vector
	  • column vector
	  • square matrix.

Apply matrix methods to solve problems	

Common matrix applications and techniques	• Basic operations include:
	  • addition
	  • scalar multiplication
	  • transposition.
	• Common practical applications of matrices, eg:
	  • solving simultaneous equations
	  • transforming vectors; shear, flip, scale and rotate etc.

Assessment outcomes

Assessment outcome 1: Understand and manipulate data in common number systems	
a	Concept of computer bases in information systems.
b	Base conversions and basic arithmetic operations using four rules of number.
c	Representation and manipulation of binary numbers.
d	Describe common uses of different bases in computing.

Assessment outcome 2: Understand and apply the foundations of computer logic	
a	The foundation of Boolean logic.
b	Common logical and derived operator symbols and operations.
c	Interpret expressions using Boolean algebraic.
d	Manipulate diagrammatic representation of functions, gates and circuits.

Assessment outcome 3: Understand and interpret information using sets, sequences, series, probability and recursion	
a	The use of sequencing and selection in computing.
b	Sequences and series and results interpretation.
c	Principles of probability and recursion.

Assessment outcome 4: Apply arithmetic expressions to abstract real world ideas	
a	Arithmetic expressions.
c	Forms of notation.
d	Algebraic expressions.
e	Functions including inputs and outputs.

Assessment outcome 5: Apply matrix methods to solve common problems	
a	The functions and common matrix forms.
b	Basic matrix operations.
c	Use matrices to perform common tasks.

Assessment guidance

This unit is assessed by an external examination set and marked by AQA. The examination takes place under controlled examination conditions and the exam date will be published at the start of each academic year.

Learners are allowed to use a non-programmable scientific calculator in the examination but, where appropriate, full working should be shown.

The examination consists of a written paper with two sections, A and B. Learners have to complete both sections and there are no optional questions within either section.

The examination is 2 hours duration and the total number of marks available in the examination is 80.

Section A is worth 50 marks and consists of relatively short questions based on the whole of the specification for this unit. Learners are required to answer **all** of the questions in Section A.

Section B is worth 30 marks and includes longer questions worth up to 15 marks each. The questions in Section B do not necessarily cover the whole of the specification for this unit at each assessment.

Learners are required to answer **all** of the questions in Section B.

AQA will ensure that the full content of the unit is covered equally over the life of the qualification.

Employer engagement guidance

Where possible set problems encountered by the learner should have a strong vocational context; engaging with local SMEs (small and medium-sized enterprises) can provide a rich source of real world scenarios to investigate and solve.

A range of organisations exist that may help centres engage local employers in the delivery and potential assessment of this unit, for example The Tech-Partnership.

Delivery guidance

General point

This unit may be delivered by either a mathematics practitioner with some computing experience or a computer practitioner confident with the mathematics content presented herein.

It should be taught, where possible, in conjunction with other units such as those covering computer programming or scripting. For example, Assessment outcome 1 integrates well with delivery in Unit 1, Fundamental principles of computing.

Although the individual components have been identified, the order in which they are delivered does not have to follow the sequence provided. Although underpinning concepts are introduced in Assessment outcomes 1, 2 and ,3 a case can be made for delivering Assessment outcomes 4 and 5 in either order, especially if they can be planned to coincide with thematically relevant learner activities in other units.

A single case study, if sufficiently complex and contextualised for learners' pathway, could be used to deliver this unit; this should allow learners to discover and appreciate how important mathematical skills are when solving real world problems. However, a number of small projects, each targeting a separate assessment outcome, could also be used to good effect.

Presentations or discussions by those working in the computing/IT sector pertaining to learners' pathway would be advantageous, especially if the importance and relevance of their underpinning mathematical skills could be highlighted.

Assessment outcome 1

Learners should consider different number systems and their vital use in computing, particularly when contextualised, eg Hexadecimal notation for error codes that system support technicians will be all too familiar or the use of binary to work out subnet masks in networking. Each learning pathway has many suitable examples; a sample is provided for illustration.

Learners could locate examples of different number systems being used in computing and be challenged to interpret these both in terms of the converted denary number, but also its application. A simple example of this might be to examine a computer's memory address, read the hexadecimal or binary value present, convert it to a denary code and then discover the ASCII character that is being stored.

To summarise learners must be able to interpret, convert and perform basic arithmetic on these values, demonstrating accuracy, correct notation and a working knowledge of how these values may be used or encountered by them in their computing career.

Assessment outcome 2

Boolean logic can be taught traditionally but comes alive when contextualised through the use of simple logic circuits, whether these are realised in physical kit form or via circuit design software tools (eg http://logic.ly).

There is opportunity here for active learning connections to be made with the delivery of computer programming (specifically 'if' statements and conditions involving logical operators), in networking (eg firewall rules) and system support (processor architecture, ALU etc).

Learning opportunities can be driven through creating simple circuits for simple real world applications, eg traffic light systems, lift doors, vending machines etc.

Assessment outcome 3

As this outcome aligns most closely with the developmental needs of a software developer, understanding these topics can be contextualised with little difficulty. For example, system practitioners may discover the probability of particular brands of hard disks failing by examining fault logs, network specialists may learn how DNS queries may be handled in a recursive manner etc.

Recursion is best demonstrated through solving simple computer programming tasks, examples of which are listed in the content although for more able learners, differentiated examples could include concepts such as finding square roots using Newton's method.

Assessment outcome 4

This assessment outcome benefits from being delivered in tandem with any programming unit, particularly where the focus is on functions and problem-solving. Arithmetic expressions are a core requirement for effective problem-solving and being able to construct these correctly; abstracting information from a real world scenario is a key skill for any software developer. Moreover functions described in this unit can typically be implemented in most modern programming languages with little modification, allowing the developer to set function inputs via the keyboard and showing the resulting outputs clearly on the screen for greater clarity and flexibility.

Assessment outcome 5

It is recommended that the standard matrix theory is delivered in a generic manner through practice of the basic operations listed in the content. However, concepts may be contextualised for software development learners through the use of one and multi-dimensional arrays. It is likely that simultaneous equations may be creatively engineered for each discipline to solve problems with two unknowns, eg

for networking it may be 'time' and 'packets lost', for system support it may be 'processor speed' and 'price'. The use of matrix transformation can provide a visual aspect to learning which provides both a good insight and a practical application, eg manipulating 3D models for computer games, modification of 2D images in graphic editing software.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

AO1: Understand and manipulate data in common number systems

Unit 1 – AO2: Understand the hardware components of a computer system
This assessment outcome should present the learner with opportunities to explore the architecture of the computer's internal components. Use of low level machine code or assembly by the learner may demonstrate how denary unsigned values are stored and coded in binary and hexadecimal formats.

Unit 1 – AO5: Demonstrate how computers process user requirements
ASCII and other character set types are used to represent denary digits in a computer system. Learners will see these encodings written in binary, octal and hexadecimal formats, which may help them work with the three different bases correctly.

This assessment outcome presents the learner with opportunities to explore the architecture of the computer's internal components. Use of low level machine code or assembly language will demonstrate how denary signed values are stored and coded in binary and hexadecimal formats.

Additional examination of these low level languages will practically demonstrate how typical opcodes such as ADD, SUB, MUL and DIV are used to perform common arithmetic in these bases.

Learners should also encounter applications such as American Standard Code for Information Interchange (ASCII) and extended ASCII character codes in binary in Unit 1 – AO5: Demonstrate how computers process user requirements or Unit 6 – PO2: Demonstrate the use of event driven language features and functions, where position is an important aspect for understanding the effect of having signed or unsigned integers as data types and the effect that has on the value range that can be stored in the computer's RAM.

Unit 3 – PO3: Demonstrate key features of a style sheet language
Although understanding the value of a positional numeral system is at the heart of any unit where numbers are used to measure sizes, speeds, quantities etc, it is possible to identify specific incidences where position is key. A simple application of this by the learner could be the use of RGB colour codes when they are written in the hexadecimal '#RRGGBB' notation; in this system the learner should appreciate that number pairs in different positions help to reveal the red, green and blue balance of the colour.

AO2: Understand and apply the foundations of computer logic

Unit 1 – AO4: Understand how data is converted to information
This unit's assessment outcome considers the use of logical operations (such as AND, OR, NOT etc) as part of the data processing cycle. Contextualising specific data processing examples, eg online banking, can provide real world examples of logic use.

Unit 7 – PO3: Implement object oriented applications to a professional standard
Many learners find mathematical content difficult to understand without suitable contextualisation. Demonstrating how simple Boolean logic operators such as AND, OR etc can be used to form the simple conditions used in selection and iteration constructs provides concrete examples that aid understanding. This is similarly the case for complex Boolean expressions that can be simplified to form easier-to-read (and easier-to-code) conditions used in selection and iteration constructs. Crucially, the learner should also benefit from seeing these concepts threaded consistently throughout all programming units.

AO3: Understand and interpret information using sets, sequences, series, probability and recursion

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This unit's third assessment outcome requires the learner to analyse different classes of algorithm and evaluate their utility for a given situation. One of the key algorithm types that the learner will investigate performs a simple recursive function. A typical practical example of this is the Fibonacci sequence, eg 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 etc.

By demonstrating how to take a mathematic process and implement it in pseudocode (or a chosen programming language) the learner should be able to see more clearly how it works and, hopefully, improve their chances of being able to identify and describe this arithmetic sequence to others. Similar implementations could be performed for geometric number sequences (eg grains of rice on a chess board).

Unit 7 – PO4: Understand how to test and maintain programs
This performance outcome tasks the learner with the practical testing of object oriented programs for faults. As part of this activity they will gather data from an implemented test plan that can be used to discover the underlying cause of the faults found. It is likely that this data can be quantified in terms of output values, calculation speed, accuracy or error occurrences etc, which provides ample opportunity for interpretation.

Comparisons can easily be made between two data sets if these are recorded before and after code modifications have been made, ultimately proving or disproving performance increases.

Unit 8 – PO2: Plan a project with others to meet a specified outcome
It is likely that planning an IT project with a strong industry focus will involve investigation and problem-solving. Gathering appropriate data is essential for the success of this type of endeavour.

AO4: Apply arithmetic expressions to abstract real world ideas

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming

Unit 6 – PO2: Demonstrate the use of event driven language features and functions

Unit 7 – PO3: Implement object oriented applications to a professional standard
The use of arithmetic expressions and bespoke functions is at the heart of many programmed solutions. Learners should be able to use the algebraic skills honed in this unit to model the calculations needed in both event driven and object oriented programs to solve real world problems.

AO5: Apply matrix methods to solve common problems

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Most programming languages support the use of two dimensional (2D) arrays of data. Learners should be able to design simple pseudocode (or program code) to build algorithms that use these 2D arrays to perform basic matrix operations, eg matrix multiplication. The results of the algorithm can be used to support and articulate the learner's understanding of the manual process and, as an added bonus, confirm the accuracy of their answers.

Useful links and resources

Books

- Anonymous, *AS-Level Maths AQA complete revision and practice*, ISBN-10 1847625819, ISBN-13 978-1847625816, Coordination Group Publications Ltd, Broughton-in-Furness (2011).
- Hanrahan V, Porkess R and Pritchard D, AQA *Certificate in Further Mathematics*, ISBN-10 1444181122, ISBN-13 978-1444181128, Hodder Education, London (2013).
- Rumsey J, *Probability for dummies*, ISBN-10 0471751413, ISBN-13 978-0471751410, John Wiley and Sons, Hoboken, NJ (2006).
- Rumsey J, *Statistics for dummies*, 2nd edition, ISBN-10 0470911085, ISBN-13 978-0470911082, John Wiley and Sons, Hoboken, NJ (2011).

12.6 Unit 6: Event driven programming

Title	Event driven programming
Unit number	D/507/6487
Unit assessment type	Centre assessed and externally quality assured
Recommended assessment method	Practical assignment This is the preferred assessment method for this unit. A centre may choose an alternative method of assessment, but will be asked to justify as part of the quality assurance process.
Guided learning hours	90
Transferable skill(s) contextualised within this unit	Communication (oral)[5]
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms that offer IDEs suited to learning and developing event driven programming techniques. These include, but are not limited to: Microsoft's Visual Basic.Net, Apple's xCode for iOS, IBM's Java using NetBeans IDE. In addition some aspects of the unit will require user rights on the development and target platforms to compile programs and install executable code. Learners should also have access to suitable offline and online learning material, manuals, help sheets and coded examples in order to encourage self-sufficiency.

[5] Please visit the specification homepage to access the transferable skills standards and associated guidance and recording documentation.

Synoptic assessment within this unit	IT: Programming linked to Units 1, 2, 3, 4, 7 and 8. Unit 1 provides the learner with a solid grounding in the different types of software and hardware that are likely to form part of an event driven solution. Unit 2 introduces many of the concepts used in event driven programming, from an overview of the different types of programming languages available to the key features common to them all and the intricate differences in usage and concept that set them apart. Unit 3 has strong links because many modern web-based solutions utilise client-side scripting that uses event driven features to manipulate website format and content. Units 4 and 7 focus on mobile applications programming and object oriented programming. Despite their differences these two units collectively form a cohesive approach that reinforces the learner's problem-solving skills and awareness of common elements such as code syntax, functions and debugging that are crucial to leveraging event driven solutions effectively. Unit 7 refines essential communication skills that are needed by the programmer to successfully present their final solution to their customers. In addition, to complete the industrial project, learners will make use of an extensive selection of programming skills and techniques gained from studying the units in this programme to help them develop a solution for a client or end user. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

This unit will equip learners with the necessary knowledge and practical ability to build high-quality coded solutions that meet client needs through their applied understanding of the Event Driven Programming paradigm.

Unit introduction

Event driven programming (EDP) is a distinct approach used to design and build coded solutions to real world problems. Solutions that are developed using the event driven method differ from traditional algorithm-oriented solutions. The key to success is for the learner to understand what kinds of system and user-defined events may be triggered and how the program can best respond to them.

For some learners, EDP will be the first approach they encounter and it offers an accessible and rewarding introduction to the professional discipline of software development.

Learners will discover how to analyse real world problems, identifying the appropriate event driven tools and techniques, that can be used to design and develop a professional solution in a critical and creative fashion. Focus is also placed on the correct use of third party libraries and the practical skills required to organise and write user and technical documentation, using the appropriate style, language and subject vocabulary that would be expected by industry.

This unit provides an opportunity to evidence achievement of the transferable skill of communication (oral).

Unit content

The key features of event driven programming languages	
Key features of event driven programming	• Event loop. • Event trigger. • Event queue. • Event dispatcher. • Event handlers and listeners. • Delegates and interfaces. • Events vs interrupts. • Suitability. • Advantages eg ease of development, rapid prototyping, gentle learning curve etc. • Disadvantages eg harder to determine program flow, increase complexity of testing etc.
Other paradigms	• Object oriented. • Procedural.
Commercial programming languages	• Commercial languages, eg Microsoft Visual Basic (classic), Microsoft Visual Basic.net, Objective C, Microsoft Visual Basic for Applications (VBA), Java (AWT, Swing), JavaScript (DOM) etc.
Common uses	• Graphical User Interfaces (GUIs). • Game development. • Mobile applications. • Enhancing web page interactivity.

Using event driven language features and functions	
Event triggers	• Hardware exceptions, eg interrupts (maskable or non-maskable) or faults. • Operating System driven events. • Changes to system settings, eg date, desktop. • Network events, eg listening to incoming connections. • User events. • Keyboard, eg key press etc. • Mouse, eg click, drag, drop, resize etc. • Touch screen, eg click, drag, drop, resize etc. • Time-based events. • Real time clock (RTC). • Timers and interval measurement, eg milliseconds. • Temporal, eg loading of forms, application exit. • Software exceptions, eg OnError.

Using event driven language features and functions	
Event controls and objects	• Common control widgets, eg textbox, button, label, listbox, checkbox, radio button etc. • Common containers, eg form, panel, tabs. • Common dialogs, eg file, print, colour, font etc. • Properties, eg enable, visible, colour, font, size, position, values etc. • Methods, eg clear, refresh/repaint etc. • Custom controls (classes, methods, properties). • Third party controls.
Event handlers	• Common event handlers, eg on load, click, got focus, lost focus, key down, key press, text changed etc. • Custom event handlers. • Adding event handler. • Raising event handler. • Removing event handler.
Identifiers	• Variables and constants (programming and system defined). • Data Types, eg integer, decimal, Boolean, string, character, date/time, currency etc. • RAM storage allocation. • Value range. • Declaration and initialisation. • Scope and visibility, eg public, private, local (module-level), global, namespaces etc.
Data structures	• Array, eg one-dimensional (simple) and multi-dimensional (complex). • Lists, stacks, queues. • Text files, XML files. • Relational databases, tables, records and fields.
Operators	• Arithmetic. • Relational. • Logical. • Concatenation.
Common language constructs	• Sequence. • Selection, eg if...else...endif, nested if, switch/case. • Iteration, eg pre-conditioned, post-conditioned, for...each mechanism.

Using event driven language features and functions	

Functions	• Programmer-defined functions:
	  • naming and declaration
	  • formal and actual parameters
	  • function calls
	  • return type.
	• Common library functions:
	  • input and output
	  • arithmetic
	  • string
	  • date
	  • file.
	• Third party library functions.

Design event driven applications	
Design tools	• Quad model, eg inputs, processes, outputs, storage. • Storyboards. • Data tables. • Action charts.
Selecting appropriate controls	• Identifying available controls and objects. • Mapping storyboard to controls. • Identification of required properties and methods.
Selecting appropriate events	• Identifying available event handlers. • Mapping storyboard to events.

The professional standards for event driven applications	
Creating an application	• Integrated development environment (IDE) • Workspaces. • Customisation eg layout, docking. • Restoring defaults. • Projects. • Project types eg form-based, console. • Creating a new project. • Navigation and storage. • Adding files to a project. • Resources, eg images, sound files etc. • Closing a project. • Moving a project safely. • File operations. • Load. • Save. • Coding conventions and standards. • Compilation/Building. • Executables. • Cleaning intermediary files.
Debugging an application	• Compilation/build problems. • Notices. • Deprecations. • Warnings. • Errors. • Debugging tools. • Watch. • Breakpoints, eg setting, clearing. • Tracing and step over. • Output and immediate window. • Log files.

Documenting an application	• Internal documentation:
	  • onscreen help eg user prompts, tool tips etc
	  • identifier naming conventions
	  • potential benefits, eg metadata, consistency, clarity
	  • common formats eg upper camel case, lower camel case, Hungarian notation etc
	  • code layout and indentation
	  • comments
	  • best practice and purpose
	  • XML and HTML documentation.
	• User documentation (paper and online).
	• Technical documentation.

The professional standards for event driven applications	

Deploying an application	• Application properties:
	  • icon
	  • start-up form
	  • application versioning, eg major, minor
	  • digital signatures.
	• Application setting persistency:
	  • configuration
	  • separate configuration files, eg .ini, .conf text files, XML etc
	  • operating system database, eg Windows Registry.
	• Packaging dependencies, eg required external components.
	• Installation tools.
	• End User License Agreement (EULA).

Reviewing an application	• Client interim reviews, eg iterative process. • Final review against client specification.
Maintaining an application	• Client feedback process. • Bug reporting. • Update strategies and frequency.

Performance outcomes

On successful completion of this unit learners will be able to:

Performance outcome 1:	Understand the key features of event driven programming languages.
Performance outcome 2:	Demonstrate the use of event driven language features and functions.
Performance outcome 3:	Demonstrate the ability to design event driven applications.
Performance outcome 4:	Implement event driven applications to a professional standard.

Grading criteria

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO1: Understand the key features of event driven programming languages	**P1** Outline **each of** the key features of event driven programs.	**M1** Compare and contrast the approach of event driven programming with other paradigms.	**D1** Assess which types of application are most suited to an event driven approach.

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO2: Demonstrate the use of event driven language features and functions	**P2** Demonstrate the correct usage of **four** different types of each of the following: • event controls • triggers • handlers.	**M2** Develop a custom event handler to meet a defined user need.	
	P3 Demonstrate the correct usage of basic identifiers, constructs and functions to meet a defined user need.	**M3** Develop a programmer-defined function to meet a defined user need.	**D2** Interpret the correct usage of a complex third party function in order to deploy correctly in an event based solution.
	P4 Implement simple data structures to meet a defined user need.	**M4** Explain the importance of accessing database structures to solve set problems.	
PO3: Demonstrate the ability to design event driven applications	**P5** Select an appropriate design tool and plan an event driven solution to meet a defined user need.	**M5** Interpret a design to deduce the correct event driven controls and handlers required to meet a user defined need.	
PO4: Implement event driven applications to a professional standard	**P6** Carry out event driven programming development using a range of IDE facilities using correct coding conventions and standards.		
	P7 Select and use appropriate features and functions to create an event driven application to meet a defined user need using good coding standards and conventions.		

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
	P8 Use appropriate debugging tools to identify and resolve programming faults.		
	P9 Document an event driven programming solution.		
	P10 Present and demonstrate the final solution to an audience.		
	P11 Deploy an event driven programming solution as a professional installation.	**M6** Review an event driven programming solution against the client specification.	**D3** Evaluate the event driven programming solution using client feedback to calculate an effective maintenance and update strategy.

Assessment amplification

This section provides amplification of what is specifically required or exemplification of the responses learners are expected to provide.

In completing performance criteria P10, learners will be presented with an opportunity to demonstrate the transferable skill of communication (oral).

There is no mandatory division of grading criteria although, as can be seen from the Grading criteria grid, some are naturally linked either by learning outcome or by developmental theme. Connections between the various pass, merit and distinction criteria can therefore suggest logical assessment grouping but do not necessarily have to be followed.

Finding a single real world scenario that can cover all criteria is difficult, but not impossible. The key to assessment in this unit, as in others, is to make assessment a natural outcome of the learning experience, with skills being assessed at a level appropriate to the learner's level of experience. The nature of the assessment technique is tutor-dependent, but some outcomes in this unit may only be achieved through generating working program code to a good standard.

Integrative assessments that connect this unit to others via connected tasks are also encouraged and it is possible that evidence produced during work for this unit may be used to provide evidence for criteria in other units.

Evidence for this unit could be generated through web pages, leaflets, electronic slideshows, wikis, presentations, podcasts or a formal report. Learners will work with a substantive real world, complex task that is solved using an event driven solution that is designed and implemented by learners.

For **D2** learners should identify and use a complex third party function. In terms of what qualifies as a complex function it is suggested that the processing undertaken by the function is multi-stage, with at least two formal parameters and a return type being used.

Employer engagement guidance

If learners are in the workplace then the centre could ask the employer whether there are any suitable projects that learners could work on as part of the team. It would be helpful for the employer to be made aware of the sort of skills that learners have to practice.

Delivery guidance

It is recommended that this unit is taught after the conclusion of Unit 2: Computer Programming which introduces (in overview only) the concept of Event Driven Programming. In addition, many elements have parallels in the Object Oriented Programming unit; indeed, many programming languages (eg Visual Basic.net, C#, Java) contain many aspects of both paradigms so there may be some merit in considering corresponding delivery for these.

Although it is suggested that the content is delivered to follow the order of the learning outcomes in this unit specification, it is not the only sequence that could be used. Tutors are encouraged to consider the holistic nature of the learner's programme and the scheme itself.

Learners must have access to the hardware and software facilities necessary for the opportunity to generate evidence for all of the grading criteria listed. As such, if centres cannot guarantee these resources, the unit should not be attempted.

It is suggested that the delivery of tools and techniques follows the pattern of a succession of small, but well chosen, exercises. These form a developmental toolkit that learners may use to build more complex solutions. This type of approach gently builds confidence and self-sufficiency in learners and is critical in promoting good problem-solving skills. This unit does not specify any named event driven programming language and centres may choose to focus on one or more languages during delivery to aid comparison and contrast. The deployment target of solutions may be desktop GUI, command line, mobile platforms, game consoles or interactive web-based.

Performance outcome 1

Outcome 1 focuses on the learner developing their knowledge and understanding of the key principles of event driven programming, how it compares and contrasts to other paradigms (eg object oriented, procedural etc) and which type of real world applications it is particularly suited to.

Investigating real world solutions will help learners to identify the appropriateness of different approaches.

Performance outcome 2

Learners will focus on the development of their skills in using different event driven tools and techniques. This could easily be achieved through a suite of small development tasks, each focusing on a different aspect.

Learners should learn how to use and demonstrate a range of different events, triggers and handlers. They should also create a custom event handler and experiment with small tasks that allow them to create program code that encompasses basic data types, constructs and functions.

Learners could work in pairs or individually to create their own programmer-defined function and explore complex third party functions and how they should be used correctly in order to solve a set problem.

Performance outcome 3

As with other programming approaches, good design is essential for creating efficient and effective programmed solutions and any design tools could be used to facilitate this. Learners would certainly benefit from using different tools in different contexts.

Action charts and the identification of event driven controls and handlers will also be essential learning.

Performance outcome 4

Learners should have as much opportunity as possible to develop designs, implement and deploy them, reflecting the learner's implicit use of the integrated development environment used to build their solution.

Good coding standards and conventions should be evident in the creation of their code. This means that functional code isn't sufficient: it should be written to a professional standard using meaningfully named identifiers, good layout and indentation, and be accurately documented.

Debugging, the identification and removal of warnings and errors from code (semantic or syntactic) and the documentation of the process to reflect good practice could be included in technical and/or user documentation.

To complete the unit, learners should prepare to be assessed reflecting the linked sequence of the learner deploying their solution (initially to a virtual device or emulator), reviewing it and evaluating it successfully. Real world scenarios, particularly with an external client, are ideal for capturing impartial critical feedback, so opportunities to work with employers or commercial outlets would be recommended.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

P1: Outline each of the key features of event driven programs
P2: Demonstrate the correct usage of four different types of each of the following:

- **event controls**
- **triggers**
- **handlers.**

Unit 4 – PO2: Apply the key features and functions of mobile application programming languages

Unit 4 provides a contextualised view of the event driven features that the learner is being asked to outline here.

These linked performance outcomes should provide the learner with a checklist of the key features to consider and help them differentiate the various event triggers, controls and handlers that are available in the target event driven programming language.

P3: Demonstrate the correct usage of basic identifiers, constructs and functions to meet a defined user need
P4: Implement simple data structures to meet a defined user need

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This introductory programming focuses on common data structures such as arrays and different types of data files. Its focus on LIFO and FIFO data structures (stacks and queues respectively) is useful as these are popular choices for common processing tasks and the learner may opt to implement these to meet a defined user need.

This unit's third assessment outcome introduces many of the key features that are common to all programming languages, irrespective of the paradigm represented. Concepts such as identifiers, constructs and functions form the basic building blocks of all programming solutions. Learners may find coverage in this unit to be a useful primer when demonstrating their skills in the selected event driven language.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Website technologies such as client and server-side scripting languages also have rich collections of features and functions that may provide the learner with suitable comparative examples that may aid understanding and demonstration.

Unit 4 – PO2: Apply the key features and functions of mobile application programming languages
Mobile applications programming also relies on the correct usage of basic identifiers, constructs and functions to solve set problems. Much of the syntax demonstrated throughout this unit will have direct equivalents in event driven programming; learners could benefit from being able to compare and contrast the two directly.

Mobile applications often use a variety of data structures including strings, arrays, lists, stacks, queues, text files and relational databases. Implementing these in different programming languages should reinforce the learner's appreciation of their features, functions and typical uses. This may in turn improve the learner's ability to select and implement the most appropriate data structures to meet the identified user need.

Unit 7 – PO8: Implement object oriented applications to a professional standard
Object oriented programming also relies on the correct usage of basic identifiers, constructs and functions to solve set problems. Much of the syntax demonstrated throughout this unit will have direct equivalents in event driven programming units; learners could benefit from being able to perform direct comparison and contrast between the two.

P5: Select an appropriate design tool and plan an event driven solution to meet a defined user need

Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2's second assessment outcome introduces the learner to many different design techniques. Some of these such as storyboards, flowcharts, action tables etc are more appropriate than others for designing event driven programs. Providing the learner with a wide range of design tools (and their associated features) should offer plenty of insight for selecting a preferred method.

Unit 4 – PO3: Demonstrate the ability to design mobile applications
Many of the recognised tools (eg action charts and storyboards) used to plan and design event driven programs can also be used to design mobile applications. Using these familiar tools for potentially different target platforms should help the learner to make informed choices when selecting their preferred design tools and further hone their abilities when using them.

P6: Carry out event driven programming development using a range of IDE facilities using correct coding conventions and standards

Unit 4 – PO4: Demonstrate the proficient use of mobile development tools
Integrated Development Environments (IDEs) are typically at the heart of any modern programming experience. This performance outcome focuses on providing the learner with an experience of using an IDE to build a mobile application. Many of the skills learnt in meeting this performance outcome should be easily transferred to the creation of event driven programs on other target platforms, along with enforcing good working practices (eg coding conventions and standards, project management etc).

P7: Select and use appropriate features and functions to create an event driven application to meet, a defined user need using good coding standards and conventions

Unit 4 – PO4: Demonstrate the proficient use of mobile development tools
This performance outcome ideally covers the process of the user selecting and using appropriate features and functions, in the event driven programming language and its IDE, to meet a defined user need. At the same time learners are encouraged to follow recommended standards for coding and workflow. This unit's performance outcome follows a similar path but from a specific mobile development viewpoint. Learners may find this useful as they continue to hone their skills.

P8: Use appropriate debugging tools to identify and resolve programming faults

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This assessment outcome introduces the concept of testing and debugging and examines the use of debug facilities such as watches, traces, breakpoints and code inspection. The learner should be able to apply tools and techniques such as these to resolve programming faults in their event driven applications.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
These performance outcomes focus on debugging applications (albeit as part of an interactive website), through the use of either client-side (JavaScript) or server-side (eg PHP) scripting tools. The learner should be able to apply similar tools and techniques to resolve similar syntax and semantic errors found in their event driven applications.

Unit 7 – PO4 Understand how to test and maintain programs
The learner may supplement the use of debugging tools by following recommended testing strategies. The object oriented programming examines the power of traditional testing methods that use test plans, coverage and trace tables to identify and resolve programming faults.

P9: Document an event driven programming solution

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
Documentation produced for mobile applications often appears to be very similar to that created for an event driven program. The learner should benefit from documenting applications that are designed for different types of platform, eg mobile devices.

Unit 7 – PO5: Understand how to produce documentation
Documentation of an application is a recurring theme for most programming paradigms and object oriented programming prove no exception. This linked performance outcome generally mirrors the content and delivery for mobile applications and may help to reinforce the learner's understanding of the required documents, their contents and standards.

P10: Present and demonstrate the final solution to an audience

Unit 8 – PO3: Follow a project plan as part of a team to meet a specified project outcome
Although it is likely that opportunities for learners to exercise presentation skills will be threaded throughout this complete technical programme, it is felt that most will benefit from a review of effective communication techniques; written, verbal and non-verbal. Preparation for the industrial project unit can provide ample scope for the learner to hone these skills and aid in the demonstration of a final coded solution to a target audience.

P11: Deploy an event driven programming solution as a professional installation

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
Many mobile applications have a similar deployment path to event driven programs; the deployment process often offers similar challenges to the user, particularly in terms of packaging the solution for the target platform. Greater levels of practice may ensure that the process is achieved confidently and correctly.

Useful links and resources

Books

- Bond M, Law D, Longshaw A, Haywood D and Roxburgh P, *Sams teach yourself J2EE in 21 days*, 2nd edition, ISBN-10 0672325586, ISBN-13 978-0672325588, Sams (2004).
- McGrath M, *Java in easy steps*, 5th edition, ISBN-10: 1840786213, ISBN-13: 978-1840786217, In Easy Steps (2014).
- Newsome B, *Beginning Visual Basic 2012*, ISBN-10 1118311817, ISBN-13 978-1118311813, John Wiley and Sons, Hoboken, NJ (2012).
- Palmer G, *Java event handling*, ISBN-10 0130418021, ISBN-13 978-0130418029, Prentice Hall (2001).
- Schildt H, Java, *a beginner's guide*, 5th edition, ISBN-10 0071606327, ISBN-13 978-0071606325, McGraw-Hill Osborne (2011).
- Sempf B, *Visual Basic 2008 for dummies*, ISBN-10 0470182385, ISBN-13 978-0470182383, John Wiley and Sons, Hoboken, NJ (2008).
- Walkenbach J, *Excel VBA programming for dummies*, 3rd edition, ISBN-10 1118490371, ISBN-13 978-1118490372, John Wiley and Sons, Hoboken, NJ (2013).

Websites

- Free Visual Basic .Net course: homeandlearn.co.uk/NET/vbNet.html
- Microsoft Developer Network: msdn.microsoft.com/en-gb/default.aspx
- Visual Basic .Net programming tutorial: tutorialspoint.com/vb.net

12.7 Unit 7: Object oriented programming

Title	Object oriented programming
Unit number	K/507/6489
Unit assessment type	Centre assessed and externally quality assured
Recommended assessment method	Practical assignment This is the preferred assessment method for this unit. A centre may choose an alternative method of assessment, but will be asked to justify this as part of the quality assurance process.
Guided learning hours	90
Transferable skill(s) contextualised within this unit	Problem-solving Communication (written)[6]
Resources required for this unit	Suitable Windows PC, Linux, Apple Macintosh OS X, Apple iOS or other suitable platforms. For learners to have every opportunity to achieve success in this unit, they will require full access to at least one IDE suitable for writing, editing and testing code in the program language selected by the centre. Up-to-date and suitable computer equipment will need to be made available to learners to allow them to employ their theoretical knowledge and understanding in a practical setting.

[6] Please visit the specification homepage to access the transferable skills standards and associated guidance and recording documentation.

Synoptic assessment within this unit	IT: Programming linked to Units 1, 2, 3, 4, 5, 6 and 8. Unit 1 provides the learner with a solid grounding in the different types of software and hardware that are likely to form part of an object oriented solution. Unit 2 introduces many of the concepts used in object oriented programming, from an overview of the different types of programming languages available to the key features common to them all and the intricate differences in usage and concept that set them apart. Unit 3 has strong links because many modern web-based solutions utilise client-side and server-side scripting that uses object oriented features to manipulate website format and content. Units 4 and 6 focus on mobile applications programming and event driven programming, respectively. Despite their differences, these two units collectively form a cohesive approach that reinforces the learner's problem-solving skills and awareness of common elements such as code syntax, functions and debugging that are crucial to leveraging object oriented solutions effectively. Unit 5 gives the learner the opportunity to sharpen their mathematical skills, which will prove essential when tackling the logic, arithmetic and algorithms needed to get the very best out of object oriented programming. Unit 8 explores different project methodologies; popular object oriented approaches such as Unified modelling language (UML) may also be explored. To complete an industrial project, learners will make use of an extensive selection of programming skills and techniques gained from studying the units in this programme to help them develop a solution for a client or end user. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

This unit will equip learners with the necessary knowledge and practical ability to design and build high-quality coded solutions that professionally meet client needs through their applied understanding of the object oriented programming paradigm.

Unit introduction

Object oriented programming (OOP) is a software development method used to design and program robust modular solutions to real world problems. It is an efficient approach, that differs from traditional procedural programming, as it is organised around objects and data rather than actions and logic. Its purpose is to simplify the development and management of complicated applications.

As learners are guided through the object oriented method, they acquire the knowledge and ability that will allow them to analyse real world problems. They will also gain the understanding to enable them to choose the correct tools and techniques within a structured approach to the design and development of reliable solutions.

Emphasis is placed on the importance of the learner's ability to organise and write user and technical documentation that is clear and comprehensive and uses the appropriate style, language and subject vocabulary.

This unit provides an opportunity to evidence achievement of the transferable skill of problem-solving and communication (written).

Unit content

Object oriented programming (OOP)	

Key aspects	• Object oriented programming:
	  • definition
	  • issues with procedural programming
	  • differences between OOP and procedural programming
	  • advantages/disadvantages of OOP compared with procedural programming and event driven programming.
	• The concept of objects, eg modelling real world objects (physical or logical).
	• Constituents of objects:
	  • object name
	  • properties
	  • methods
	  • constructor
	  • destructor.

Common coding terms and concepts of object oriented programming	• Core concepts of OOP:
	  • abstraction
	  • classes
	  • encapsulation
	  • inheritance (single and multiple)
	  • objects and instances
	  • polymorphism.
	• Other key terms and concepts:
	  • association
	  • aggregation
	  • pre-defined classes:
	  • class libraries
	  • imported
	  • downloaded
	  • subclass
	  • superclass
	  • transient object
	  • virtual class.

Object oriented programming (OOP)	

Variables and methods	• Main properties as variable types:
	  • global
	  • local
	  • static.
	• Instance and class variables and methods:
	  • instance variables:
	  • behaviour
	  • where they occur
	  • instance methods:
	  • behaviour
	  • the programming elements on which they act
	  • limitations
	  • class variables:
	  • behaviour
	  • where they occur
	  • class methods:
	  • behaviour
	  • the programming elements on which they act
	  • limitations.

Popular object oriented programming languages	(eg) • ASP.net (Active Server Pages). • C++. • Java. • Microsoft C#. • Objective-C. • PHP: Hypertext Preprocessor (PHP). • Python. • Ruby. • Typical commercial applications.

Software solutions designed using an object oriented approach	

Data modelling concepts and standard techniques	• Data modelling:
	  • purposes
	  • different approaches to modelling:
	  • conceptual
	  • enterprise
	  • logical
	  • physical.
	• Unified modelling language (UML):
	  • what it is (Blueprint analogy)
	  • purpose
	  • concepts
	  • benefits
	  • use cases
	  • user Stories
	  • UML Diagrams
	  • Use Case Diagrams
	  • sequence diagrams
	  • class diagrams.
	• Class-Responsibility-Collaboration cards (CRC):
	  • purposes
	  • ideal group size
	  • use of CRC cards in project groups
	  • layout of CRC cards.

Design	• Use modelling and standard techniques to design:
	  • classes
	  • dependencies
	  • inheritance
	  • methods
	  • scope of attributes and methods
	  • definition of relationships between objects
	  • passing messages between objects.

The professional standards for object oriented applications	

Syntax elements of object-orientated applications	• Classes:
	  • declaring classes
	  • predefined classes
	  • declaring methods
	  • defining methods
	  • overloading methods
	  • constructors
	  • destructors
	  • passing information to methods/constructor
	  • understanding class members
	  • member access control within classes
	  • field initialisation.
	• Container classes:
	  • static nested classes
	  • inner classes
	  • shadowing
	  • serialisation.
	• Objects:
	  • creating objects ('instantiation')
	  • using objects
	  • reusing objects.
	• Enumeration types.
	• Annotations:
	  • declaring annotations types
	  • predefined annotation types.
	• Interfaces:
	  • defining interfaces.
	• Inheritance:
	  • multiple inheritance
	  • overriding and hiding methods
	  • polymorphism
	  • hiding fields
	  • super/sub class objects
	  • abstraction methods.
	• Numbers:
	  • number classes.
	• Strings:
	  • number/string conversion
	  • character manipulation within strings
	  • comparison of strings and string portions.
	• Data primitives and types.

The professional standards for object oriented applications	

Syntax elements of object-orientated applications continued	• Variables:
	  • instance variables
	  • class variables
	  • local variables
	  • parameters
	  • naming rules and conventions:
	  • reserved or keywords
	  • Hungarian notation
	  • Camel case
	  • Selection, declaration and initialisation of variables.
	• Arrays:
	  • creating, initialising and accessing
	  • copying
	  • manipulating.
	• Operators:
	  • assignment operators
	  • arithmetic operators
	  • unary operators
	  • equality and relational operators
	  • logical operators
	  • type comparison operators.
	• Expressions.
	• Statements.
	• Blocks.
	• Control Flow Statements:
	  • 'if-then' statements
	  • 'if-then-else' statements
	  • 'switch' statements
	  • 'while' statements
	  • 'while-do' statements
	  • 'for' statements
	  • 'break' statements
	  • 'continue' statements
	  • 'return' statements.
	• Library functions.

Automatic source code generators	• CodeDom (C#, VB.NET). • WebPerformer(Java). • PyXB(Python).

The professional standards for object oriented applications	

Recommended practice standards and standard libraries for reliable and secure programming	• The CERT Oracle Secure Coding Standard for Java.
	• High Integrity C++ (HIC).
	• Coding standards for Cocoa.
	• Style guides for OOP languages:
	  • C++
	  • Java
	  • Objective-C
	  • Python.

Database connectivity	• Linking to databases such as the Java Development Kit (JDK).
	• Linking with Structured Query Languages.
	• Drivers for connectivity such as the Java Database Connectivity Driver (JDBC):
	  • connectivity objects
	  • statement objects.

How to test and maintain programs	

Testing	• Importance of testing strategies.
	• Test coverage:
	  • logical pathways
	  • try…catch
	  • handling runtime errors.
	• Designing and implementing a test plan:
	  • test
	  • date
	  • expected results
	  • actual results
	  • actions taken.
	• Compiling and debugging within an integrated development environment (IDE).
	• Validation of data.
	• Error management.

Reviewing and maintaining	• Importance of regular project reviewing against specification requirements. • Iterative project design and implementation process.

How to produce documentation	

User documentation	• Good practice:
	  • clear, concise 'Plain English' writing
	  • lack of jargon
	  • use of only necessary technical terminology
	  • suitable document structure and layout
	  • applicable for target audience
	• Content:
	  • user instructions
	  • install
	  • normal usage
	  • uninstall
	  • diagrams
	  • screen shots
	  • FAQ
	  • troubleshooting.

Technical documentation	• Good practice:
	  • clear, concise technical writing
	  • appropriate use of technical terminology
	  • suitable document structure and layout
	  • applicable for target audience
	• Content:
	  • requirements specification
	  • form design
	  • diagrams (eg UML)
	  • data flow diagrams
	  • entity relationship models
	  • data dictionary
	  • class schemas
	  • testing strategy and results
	  • debugging and actions taken
	  • commented program code
	  • documenting identifiers
	  • maintaining and identifying system decay.

Performance outcomes

On successful completion of this unit learners will be able to:

Performance outcome 1:	Understand object oriented programming (OOP).
Performance outcome 2:	Design software solutions using an object oriented approach.
Performance outcome 3:	Implement object oriented applications to a professional standard.
Performance outcome 4:	Understand how to test and maintain programs.
Performance outcome 5:	Understand how to produce documentation.

Grading criteria

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO1: Understand object oriented programming (OOP)	**P1** Compare and contrast object oriented programming and procedural programming.		
	P2 Model a real world object as a class with methods and properties.	**M1** Derive new class from existing class to model real world objects.	
	P3 Describe the **three** main variable types giving examples.	**M2** Compare and contrast instance and class variables.	**D1** Evaluate the use of instance and class methods explaining their limitations.
	P4 Describe the commercial applications of **three** different object oriented languages.		
PO2: Design software solutions using the object oriented approach	**P5** Explore a user-defined **problem** and identify the requirements for an object oriented solution.		
	P6 Use relevant techniques (including **two** diagramming techniques) to design an OOP solution to **resolve the problem**.	**M3** Describe the techniques used in the design of an OOP solution for a defined user need.	**D2** Justify the choice of the techniques used.

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO3: Implement object oriented applications	**P7** Plan and implement the design to solve the problem.		
	P8 Use an automatic source code generator to provide **one** aspect of the coded **solution to the problem.**		
	P9 Use two library functions in the coded **solution to the problem.**		
	P10 Implement database connectivity in the coded **solution to the problem.**		
	P11 Check if the **problem** has been solved and review your approach to problem-solving.	**M4** Evaluate the program in terms of usability.	**D3** Justify **two** ways in which the program could be technically improved and the resulting benefits to the user.
PO4: Understand how to test and maintain programs	**P12** Produce a plan to test the program.	**M5** Analyse the results of testing and document actions taken.	**D4** Evaluate the effectiveness of the test plan.
	P13 Design a comprehensive review strategy to update and maintain the program.	**M6** Explain the reasons for reviewing and maintaining the program.	
PO5: Understand how to produce documentation	**P14** Create **user documentation** for the program.		

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
	P15 Create **technical documentation** for the program.		

Assessment amplification

This section provides amplification of what is specifically required or exemplification of the responses learners are expected to provide.

In completing performance criteria P14 and P15, learners will be presented with an opportunity to demonstrate the transferable skill of communication (written).

In completing performance criteria P5 to P11, learners will be presented with an opportunity to demonstrate the transferable skill of problem-solving.

Assessment tasks can take a number of forms within this unit, such as presentations, wikis, podcasts, video or audio recordings, reports and essays, case studies, testing and user documentation and programming activities.

For **P2** the real world object could be any physical object such as a human being, a car or a pot of paint.

For **P6** learners should prepare a full design of their solution including diagrams and narrative as appropriate.

For **P7** the design should be implemented and learners would benefit from keeping an implementation log or diary to support later activity.

For **P8** and **P9** learners should investigate automatic source code generators and library functions, then consider which would be most appropriate for inclusion in the solution.

For **M4** usability is focussed on the user's (or users') experience.

For **P14** and **P15** learners will produce written guidance for the user and a technical document. This will require some extended writing, but could be produced as a paper or as an electronic document.

Employer engagement guidance

If learners are in the workplace then the centre could ask the employer whether there are any suitable projects that learners could work on as part of the team. It would be helpful for the employer to be made aware of the sort of skills that learners have to practice.

Delivery guidance

One of the main purposes of this unit is to provide the IT industry with a robust and reliable qualification that confidently assesses the modern technical knowledge, skills and understanding that are demanded from the broad range of roles that make up the profession.

It is an intended theme of this unit that learners begin to write simple programs at the earliest opportunity and that exercises and assignments gradually increase in complexity in order to strengthen and build on the foundation of learners' knowledge gained in previous lessons.

With this in mind, it is essential that learners have access to high quality, up-to-date software and hardware that will allow them to study for this qualification unhindered.

Tutors delivering this award must emphasise to all learners the importance of adopting a professionally disciplined and logical approach to their object oriented projects, using industry proven planning, design and programming procedures demonstrated within the course.

It is essential that programming examples are utilised whenever an opportunity arises in order that learners grasp the importance of fully understanding the concepts of object oriented programming. To enable this to occur, tutors should choose one commonly used object oriented language for delivery, to demonstrate examples and for use in assignment projects. However, when creating applications learners may wish to work on a variety of platforms such as web-based, or on a games console or mobile device.

When issuing assignments to learners, tutors should urge learners to adopt an industry-inspired approach by breaking down elements of projects in to separate, defined stages such as planning, design, class documentation, production of code, testing/debugging and production of user/technical documentation.

With the above in mind learners should be provided with as many opportunities as possible to work and collaborate together in project teams, as would be the normal industry approach, in order to plan, design and program solutions to problems.

Below is an example of how the unit could be delivered by following the prescribed performance outcomes; however, it is by no means the only approach to the unit and tutors may adopt their own delivery schedule.

Performance outcome 1

Introduces learners to the object oriented paradigm – it's purpose and principles. Tutors should aim to deliver comprehensive content focusing on object oriented concepts, procedures and languages.

Performance outcome 2

In this outcome, learners should be introduced to a variety of industry-proven procedures and best practices within the areas of problem analysis, project planning and solution design. Design tools, methodologies and programming languages selected by learners should be suitable for their chosen platform.

Performance outcome 3

This outcome concentrates on the writing of code. Tutors should provide assignments that are structured in such a manner that learners competently use an applicable IDE and a range of tools and features such as standard coding libraries, automatic source code generators and database connectivity. Learner implemented code should be directly developed from a range of object oriented concepts utilised within the design stage. Good quality program syntax should be evident within solutions.

Performance outcome 4

Good programming practice and high quality documentation go hand-in-hand and this outcome focuses on these elements. Within their program solutions, learners should demonstrate good code layout, the importance of indentation for clarity and the use of comments.

Learners should adopt and, within their created solutions, demonstrate their awareness of best practice and established programming principles through competent use of object oriented programming concepts within their programs to avoid issues such as code duplication etc.

Learner developed solutions must also include the writing of good quality technical and user documentation that is clearly written for the appropriate audience.

With regard to consistency and the simplifying of assessment, tutors may wish to provide all learners with the same assignment; however, it is acceptable for tutors to encourage learners to come up with their own projects as long as they embrace the unit performance outcomes.

Tutors may also wish to consider approaching external companies or organisations to suggest appropriate project activities or even encourage learners to use their knowledge and skills to contribute to ongoing open source development projects.

Performance outcome 5

The ability to write clear and concise documentation for technical and non-technical audiences is important, and in this outcome learners focus on their knowledge and skills to produce both user and technical literature.

Learners will embrace the differing approaches (including structure, layout and terminology) and varying content (such as diagrammatic, illustrative and technical jargon (or lack of) specific to each audience).

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

P1: Compare and contrast object oriented programming and procedural programming

Unit 1 – AO5: Demonstrate how computers process user requirements
This unit introduces learners to key features of high level languages such as C++ (which is object oriented) and others that are likely to be procedural in nature eg PHP. Learners should be able to draw upon this coverage to assist their completion of this performance outcome.

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
This assessment outcome typically introduces and examines aspects of the differences and similarities of the object oriented and procedural programming paradigms. The learner may be able to draw upon previous sample code and concepts from this unit and may compare and contrast these.

Unit 6 – PO1: Understand the key features of event driven programming languages
In a bid to distinguish event driven programming from other paradigms there are opportunities in this unit to explore object oriented and procedural programming. The learner could utilise this coverage to meet the needs of this performance outcome.

P2: Model a real world object as a class with methods and properties

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
It is likely that the learner will encounter classes and many of their common characteristics through this assessment outcome. An understanding of the principles and application of encapsulation should enable the learner to tackle this performance outcome with confidence.

P3: Describe the three main variable types giving examples

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This unit's third assessment outcome introduces many of the key features that are common to all programming languages, irrespective of the paradigm represented. Concepts such as identifiers, constructs and functions form the basic building blocks of all programming solutions. Learners may find coverage in this unit acting as a useful primer when demonstrating their skills in the selected object oriented programming language.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Website technologies such as client and server-side scripting languages also have rich collections of features and functions; these may provide the learner with suitable comparative examples to aid understanding and demonstration.

Unit 4 – PO2: Apply the key features and functions of mobile application programming languages
Mobile applications programming also relies on the correct usage of basic identifiers, constructs and functions to solve set problems. Much of the syntax demonstrated through this unit will have direct equivalents in the object oriented programming unit; learners could benefit from comparing and contrasting the two directly when tackling this type of task.

P4: Describe the commercial applications of three different object oriented languages

Unit 2 – AO1: Understand the different types of computer programming, languages and the common uses
This Unit 2 discusses many of the common uses of programming languages, particularly commercial applications. Based on previous work on the different paradigms, learners should be able to draw conclusions about which commercial applications are more suited to an object oriented approach.

P5: Explore a user-defined problem and identify the requirements for an object oriented solution

Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2 explores the software development lifecycle and should prepare learners for gathering the requirements and understanding the problem.

Unit 8 – PO1: Understand the project planning process
Unit 8 discusses different project methodologies; it is quite common for clients to prefer object oriented methods, eg UML, to investigate requirements and solve problems. Some UML design aspects dovetail neatly into object oriented programming, and the learner may benefit from the synthesis opportunities presented.

P6: Use relevant techniques (including two diagramming techniques) to design an OOP solution to resolve the problem

Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2's second assessment outcome introduces the learner to many different design techniques. Some of these, such as pseudocode, flowcharts, structure diagrams etc are more appropriate than others for designing aspects of object oriented programming programs. Providing the learner with a wide range of design tools (and their associated features) should offer plenty of insight for selecting a preferred method.

Unit 8 – PO1: Understand the project planning process
Unit 8 discusses different project methodologies and it is quite common for clients to prefer object oriented methods, eg Unified Modelling Language (UML) to investigate requirements and solve problems. Some UML design aspects dovetail neatly into object oriented programming and the learner may benefit from the synthesis opportunities presented.

P7: Plan and implement the design to solve the problem

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language

Unit 4 – PO2: Apply the key features and functions of mobile application programming languages

Unit 6 – PO2: Demonstrate the use of event driven language features and functions
All of these outcomes focus on the general building blocks (identifiers, operators, constructs, functions etc) that form the basis of any programming language. An object oriented solution also relies on using these so learners should benefit from the synthesis of language skills and their reapplication to solve the set problem.

P9: Use two library functions in the coded solution to the problem

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Library functions are a form of modularity; this topic is firmly detailed in this unit's third assessment outcome, where functions, arguments and parameters are introduced. This should provide a firm foundation, allowing the learner to understand the requirements of the complex task and select suitable functions to solve it using the correct syntax.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
These performance outcomes also examine the use of library functions to solve set problems albeit in a web technologies framework. As most programming languages require similar sets of library functions it is highly probable that examples discussed in this unit would provide analogous for object orient programming solutions, eg finding the length of a string in characters, a ubiquitous task in any programming paradigm.

Unit 4 – PO2: The key features and functions of mobile application programming languages
This unit's second performance outcome also introduces the concept of functions and differentiates programmer-defined functions (as must be created by the user here to solve a complex task) from the common library functions that also exist. Although the languages used may be different, the problem-solving concepts are similar and the learner will benefit from comparing and contrasting of modular design in another programming language.

Unit 5 – AO2: Understand and apply the foundations of computer logic
Most functions that perform complex tasks will typically involve the application of Boolean logic in order to process the data successfully. Learners may rely on the introduction to this concept that is provided here.

Unit 6 – PO2: Demonstrating the use of event driven language features and functions
This performance outcome also introduces the concept of functions and differentiates programmer-defined functions (as must be created by the user here to solve a complex task) from the common library functions that also exist. Although the languages used may be different, the problem-solving concepts are similar and the learner will benefit from comparing and contrasting of modular design in another programming language.

P10: Implement database connectivity in the coded solution to the problem

Unit 1 – AO3: Understand the software requirements of a computer system
This assessment outcome introduces the concept of application software and would, typically, include the role of the relational database system in modern computers. As a direct consequence, learners should understand the concept of a relational database system and how it is structured, connected to and queried.

Unit 1 – AO5: Demonstrate how computers process user requirements
This unit introduces the key features of fourth generation languages (4GL) such as Structured Query Language (SQL). Programming languages typically use a combination of library functions and SQL syntax to successfully query data in a relational database system. Examples provided for this particular 4GL could support the learner when generating the appropriate object oriented code needed for this performance outcome, depending on the depth of coverage chosen.

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
A core objective of server-side scripting is accessing a backend database to create dynamic content. This unit provides learners with the tools and techniques necessary to connect to a database to solve set programming problems. Although the target language's syntax will be different, skills exercised here should prove to be highly transferable.

P11: Check if the problem has been solved and review your approach to problem-solving

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This assessment outcome introduces different testing and debugging techniques. An examination of the difference between black-box and white-box testing and the selection of meaningful test data should prepare the learner successfully for creating a test plan with good coverage. The learner should be able to apply tools and techniques such as these to solve programming faults in their objected oriented programs and to determine that the problems have been solved.

P12: Produce a plan to test the program

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
This assessment outcome introduces different testing and debugging techniques. An examination of the difference between black-box and white-box testing and the selection of meaningful test data should prepare the learner successfully for creating a test plan with good coverage. The learner should be able to apply tools and techniques such as these to solve programming faults in their objected oriented programs.

Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language

Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
These performance outcomes focus on debugging applications, albeit as part of an interactive website, either through the use of client-side (JavaScript) or server-side (eg PHP) scripting tools. The learner should be able to apply similar tools and techniques to resolve similar syntax and semantic errors found in their object oriented programs.

Unit 6 – PO4: Implement event driven applications to a professional standard
Event driven applications also need to be tested effectively and learners should benefit from testing tools and techniques introduced in this unit.

P13: Design a comprehensive review strategy to update and maintain the program

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Reviewing is a key aspect of this unit's third assessment outcome and could assist the learner in designing a suitably detailed update and maintenance strategy.

P14: Create user documentation for the program

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Learners should encounter user documentation in this unit, providing them with a checklist of desirable features (eg use cases, screen captures, plain English etc) and practices that should be avoided (eg unnecessary jargon).

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
Documentation produced for mobile applications often appears to be very similar to that created for an event driven program. The learner should benefit from documenting applications that are designed for different types of platform, eg mobile devices.

Unit 6 – PO4: Implement event driven applications to a professional standard
Documentation of an application is a recurring theme for most programming paradigms and event driven programming proves no exception. This linked performance outcome generally mirrors the content and delivery for object oriented programs and may help to reinforce the learner's understanding of the required documents, their contents and standards.

P15: Create technical documentation for the program

Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Learners should encounter technical documentation in this unit, providing them with a checklist of desirable features (eg clear, technical writing etc) and practices that should be avoided (eg inappropriate use of terminology).

Unit 4 – PO5: Create and deploy a working mobile application using cross platform development
Documentation produced for mobile applications often appears to be very similar to that created for an event driven program. The learner should benefit from documenting applications that are designed for different types of platform, eg mobile devices.

Unit 6 – PO4: Implement event driven applications to a professional standard
Documentation of an application is a recurring theme for most programming paradigms and event driven programming also proves no exception. This linked performance outcome generally mirrors the content and delivery for object oriented programs and may help to reinforce the learner's understanding of the required documents, their contents and standards.

Useful links and resources

- 10 Object oriented design principles for the Java programmer:
  javacodegeeks.com/2012/08/10-object oriented-design-principles.html
- A tour of the standard library:
  stroustrup.com/3rd_tour2.pdf
- How to write software documentation:
  wikihow.com/Write-Software-Documentation
- Introduction to object oriented programming concepts and more:
  codeproject.com/Articles/22769/Introduction-to-Object oriented-Programming-Concep
- Microsoft .Net CodeDom technology:
  codeguru.com/vb/gen/article.php/c19573/Microsoft-NET-CodeDom-Technology.htm
- Standard coding, OOP techniques and code reuse:
  slideshare.net/rayhan_wm/standard-coding-oop-techniques-and-code-reuse

12.8 Unit 8: Industrial project

Title	Industrial project
Unit number	A/507/6464
Unit assessment type	Centre assessed and externally quality assured
Recommended assessment method	Practical assignment This is the preferred assessment method for this unit. A centre may choose an alternative method of assessment, but will be asked to justify as part of the quality assurance process.
Guided learning hours	90
Transferable skill(s) contextualised within this unit	Teamwork[7]
Resources required for this unit	Prior learning and technical resources appropriate for the employer/client brief.
Synoptic assessment within this unit	The Industrial project unit allows centres to demonstrate explicit synoptic assessment. Principally it represents the key opportunity for learners to demonstrate that they can identify and use effectively, in an integrated way, an appropriate selection of skills, techniques, concepts, theories and knowledge from across many of the units and outcomes delivered on the IT: Programming pathway. It also affords centres with the chance to creatively exercise the teaching and learning links between the different units and can contribute and promote a holistic delivery programme that encourages the application of prior or concurrent learning. In addition it can be used as a vehicle to provide amplification and aggregation for the grading criteria of internally assessed units into a coherent and technically relevant industry-based task with its sector-specific nature encouraging, developing and assessing learners' use of transferable skills in realistic and demanding contexts. Extended guidance on synoptic assessment is provided later in this unit documentation.

Aim and purpose

To enable learners to participate in a technical project that requires them to work as part of a team. This will help to develop their knowledge, skills and understanding with respect to the project planning and implementation process and the importance of working effectively as part of a team. Learners will undertake a project that reflects the pathway they are following within the overall qualification.

[7] Please visit the specification homepage to access the transferable skills standards and associated guidance and recording documentation.

Unit introduction

The success of a project will depend on how efficient the project team are in executing the project goals and objectives during the project management lifecycle. This will be impacted, in part, by how well work roles are matched to individual competencies, how well members work within a group and whether project milestones are prioritised effectively.

This unit provides an opportunity to evidence achievement of the transferable skill of teamwork.

Teamwork is important because it creates **human synergy**. It amplifies by producing end results that are greater than the sum of the individual contributions.

The unit is a focal end point, providing opportunities for real synergy across the qualification. It is mandatory as it has been requested by industry. Learners will learn about the project life cycle, resources, issues and methodology. They will be able to produce a project specification and plan as a team for an identified project. Roles and responsibilities will be agreed between the team so that project expectations are clearly understood by all interested stakeholders.

Learners will also learn about the importance of good time management and communication skills in order to facilitate a successful project outcome.

This unit will provide the opportunity for synoptic assessment as learners may undertake projects that are linked to one or more units across the qualification spectrum.

Unit content

The project planning process	
Life cycle	• Analysis phase (eg defining goals, feasibility study, preparing project proposal). • Design phase (eg identifying possible solutions, systems). • Implementation phase (eg creating solution, testing, training). • Evaluation phase (eg project review, feedback from client/user).
Project methodologies	(eg) • Prince 2. • Rapid Application Development (RAD). • Waterfall model. • Structured systems analysis and design methodology (SSADM). • Critical path method (CPM). • as required by Client.
Project problems	• Poor communication (eg between project team members, between stakeholders). • External factors (eg change of project team members, change in finance, timescales). • Lack of project management and leadership. • Conflict between members and/or clients. • Poor tracking of project progress. • Adhering to legislative and organisational requirements. • Unrealistic timescales. • Lack of testing. • Poor quality of product.

The project planning process	
Project planning software and its functionality	(eg) • Project Libre. • Microsoft Project. • Freedcamp.
Benefits of project planning software	• (eg) • File sharing. • Speed of response. • Communication. • Equal access to timescales and tasks.

Planning a project with others to meet a specified outcome	
Stakeholders	(eg) • Project sponsor (funds the entire project). • Business experts (define their requirements for the end product/result). • Project manager (controls the project plan). • Project team (builds the project outcome based on accepted roles and responsibilities). • End user (may be the client or the employees of the client).
Stakeholder expectations	• Level and method of communication used to communication with them. • Kept up-to-date with respect to progress. • How will they measure success? • Outcome of project. • Timescales. • Budgets. • Resources.

Defining project goals	• Review context and requirements of project.
	• Identify key attributes of project:
	  • types of products/services to be delivered
	  • critical constraints eg project duration, costs etc
	  • technologies, tools and techniques to be used
	  • quality requirements to be met
	  • benefits to be achieved.
	• How success will be measured.
	• Must be well defined and measurable.

Project deliverables	• Product or service given to a client.
	• Have due dates (deadline for delivery).
	• Deliverables are:
	  • tangible
	  • measurable
	  • specific.
	• For internal and/or external clients, satisfying a milestone as part of the project plan.

Planning a project with others to meet a specified outcome	

Project schedule	• Required activities and tasks.
	• Resources for each task/activity.
	• Length of time to complete each task/activity.
	• Resource constraints.
	• Develop critical path (tasks dependent on other tasks).
	• Develop schedule – calendar schedule of all tasks with:
	  • estimate of time period
	  • resource required for each task
	  • project team member(s) involved
	  • time on each task
	  • begin and end dates of individual task.

Allocating roles and responsibilities	• Identify roles within project team. • Identify tasks/activities to be carried out. • Define roles based on relevant skills, qualities and knowledge.

Following a project plan as part of a team to meet a specified project outcome	
Time management skills	• (eg) • Set clear goals. • Break down personal goals into discreet steps. • Review personal progress towards identified goals. • Prioritising – order of importance/urgency. • Organisation of work schedule. • List making (eg to do list). • Perseverance (eg take a positive attitude). • Avoid procrastination (eg avoid distractions from noise, emails).
Measuring progress against milestones	• Measure quantity of work output. • Determine whether timelines within plan needs to be modified. • Review their progress with others.
Effective communication	• Interpersonal skills (eg verbal communication skills). • Cues in verbal communication (eg body language, tone of voice). • Questioning techniques (eg open/closed probing). • Written communication (eg note taking, reports, emails, letters). • Understanding the audience (eg who are they, what their knowledge base is). • Clarification of requirements. • Verbal and non-verbal techniques.

Following a project plan as part of a team to meet a specified project outcome	
Effective team meetings	• Agenda available prior to meetings. • Start and end promptly. • Check attendees. • Establish and review ground rules. • Assign administrative roles. • Summarise decisions and assign action items. • Debrief, evaluate and plan for improvement. • Distribute meeting minutes promptly.
Giving constructive feedback	• Provide feedback promptly. • Descriptive. • Not judgemental. • Supportive. • Fair and reasonable. • Positive as well as negative (feedback sandwich).
Receiving feedback	• Pay attention. • Do not become defensive. • Do not launch a counterattack. • Avoid flippancy. • Convey understanding. • Indicate a willingness to work together towards a solution/improvement. • Accept praise graciously.

Reviewing collaborative working as part of a project team	
Review process	• Success of project. • Meeting of objectives. • Factors influencing the outcome (eg work process, external changes, interpersonal aspects).

Performance outcomes

On successful completion of this unit, learners will be able to:

Performance outcome 1:	Understand the project planning process.
Performance outcome 2:	Plan a project with others to meet a specified outcome.
Performance outcome 3:	Follow a project plan as part of a team to meet a specified project outcome.
Performance outcome 4:	Review collaborative working as part of a project team.

Grading criteria

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO1: Understand the project planning process	**P1** Describe the **four** stages of the project life cycle.		
	P2 Compare the characteristics of **three** project methodologies.		
	P3 Identify **six** of the problems that can occur within a project.	**M1** Explain how the problems identified could be overcome.	
	P4 Compare the functionality of **two** examples of project planning software.	**M2** Explain the benefits of using project planning software in managing a project.	
PO2: Plan a project with others to meet a specified outcome	**P5** Identify a minimum of **two** stakeholders for a specified project.	**M3** Explain the expectations of the stakeholders identified for a specified project.	
	P6 Create a project plan for a specified project.	**M4** Describe in detail **three or more** goals required to complete a specified project.	**D1** Discuss the project deliverables required to meet the goals for a specified project.
	P7 Check the plan with stakeholders to confirm accuracy.		
	P8 Outline the project schedule for a specified project setting clear timelines and milestones.		
	P9 Allocate the roles and responsibilities required for a specified project.		

Performance outcomes	Pass	Merit	Distinction
	To achieve a pass the learner must evidence that they can:	**In addition to the pass criteria, to achieve a merit the evidence must show the learner can:**	**In addition to fulfilling the pass and merit criteria, to achieve a distinction the evidence must show that the learner can:**
PO3: Follow a project plan as part of a team to meet a specified project outcome	**P10** Effectively plan and manage own time as part of a project team.		
	P11 Carry out agreed roles and responsibilities, negotiating and resolving any conflicts or shortcomings during the project.	**M5** Measure progress against identified milestones.	**D2** Justify any changes to resource and/or milestones.
	P12 Participate in team meetings to discuss project progress.	**M6** Communicate effectively through verbal and non-verbal methods with project team members.	**D3** Explain the importance of effective non-verbal communication.
	P13 Consider feedback received from project team members and make adjustments as appropriate.		
PO4: Review collaborative working as part of a project team	**P14** Reflect on the effectiveness of collaborative working for a specified project.	**M7** Suggest a range of improvements to support collaborative working for future projects.	
	P15 Evaluate your personal contribution to the project.		

Assessment amplification

This section provides amplification of what is specifically required or exemplification of the responses learners are expected to provide.

In completing this unit, the process of teamworking will be continually developed and reflected upon.

This unit is best assessed through a single project and would benefit from employer involvement.

Minutes of meetings, development logs, emails, videos or recordings of discussions, written documentation, diagrams and charts, PowerPoint slides as well as the outcome of the project could provide evidence for this unit.

Note: There will be an opportunity to bring in learning from all other units studied and learners should be encouraged to draw on their experience as well as what they have learned.

To address this particular learning outcome, learners must be provided with a scenario that has a specified project outcome. They will be expected to work in teams to produce the project plan, but the individual learner evidence must show how the learner met the assessment criteria for themselves.

For **D3** learners must demonstrate an understanding of non-verbal communication and the problems that inexplicit written communication can cause.

Employer engagement guidance

If learners are in the workplace then the centre could ask the employer whether there are any suitable projects that the learners could work on as part of the team. It would be helpful for the employer to be made aware of the sort of skills that the learners have to practice.

Delivery guidance

This unit could effectively be used alongside other practical units, which allows learners to work in a team environment.

Performance outcome 1

It is important that learners understand the project planning process and that it is about collaborative working from the outset. Learners need to understand that when working as a team on any project, they have the opportunity to make suggestions, volunteer for specific tasks/activities etc.

Through a class discussion, learners could be given an overview of the project life cycle with emphasis on how they could contribute as a team member. Learners should also have an awareness of the different project methodologies and that different organisations/situations may use different ones.

Problems do occur when working on projects. Therefore it is important that learners have the opportunity to discuss the various issues/problems that could occur, the effect these could have on the project and how they can be overcome (if at all).

Learners could be given project scenarios and in small groups give presentations explaining the problems that may arise, whether they can be overcome and how.

It is also useful at this stage for learners to be introduced to project planning software such as Project Libre and Microsoft Project.

Performance outcome 2

Learners need to understand who the stakeholders are within a project and the fact that the project team is also classified as a stakeholder. Learners could be asked to research different stakeholder types and through a group discussion explain what they believe the individual stakeholder expectations to be. Learners could be given a scenario where they interview different people taking on the roles of various stakeholders in order to establish their expectations.

Part of the project planning process requires discussions with and between stakeholders. Learners could work in small groups or a larger group and discuss the requirements for a specific project. Based on the information they are provided with, they could identify the stakeholders, their expectations, the project goals and deliverables and the project schedule. The groups could give presentations of their plans to the other groups.

Allocating roles and responsibilities is never an easy task when working on projects; learners should be made aware on what basis choices may be made eg experience, knowledge, skills etc. They also need to be aware that they could nominate people for different roles as long as they can justify their choices. Learners could be given different project scenarios to consider (eg setting up a new network) and then: a) identify the roles and responsibilities that would be required in a project team, and b) agree who would carry out the different roles.

Performance outcome 3

In order to carry out their own individual roles and responsibilities effectively, whether working in isolation or as part of a team, it is important that learners develop their time management skills. They could consider the roles and responsibilities that they have been allocated based on the previous scenarios and consider how they will address their time management. They need to be aware of the importance of documenting their time management plans where timescales are involved eg prioritising tasks.

Measuring progress against milestones is important for individual team members as well as the team as a whole. They need to have an understanding of how they measure progress and what they measure it against. In their groups they could discuss how they would measure the team's progress for the project scenario as well as how the individual team members would check their personal progress.

Communication can make or break many a project and it is important that learners understand what effect poor and good communication could have on their project. They could be asked to work in small groups in order to come up with examples of poor communication and its effects on the project outcome. They could also identify examples of good communication and how it can overcome problems/issues as well as other positive effects on the project outcome.

Team meetings also play a big part in projects: they are used to review progress, discuss issues and problems etc. Learners should be taught how to plan, participate in and record team meetings. They could work as large or small project groups and practice planning team meetings, taking an active part in discussions and formally documenting outcomes.

Providing and receiving constructive feedback can be a challenge for many people. Learners need to understand that, especially when working on projects, they may have to provide feedback – for instance telling another team member that they are delaying the project because they have not finished a task, or telling someone that the results of their particular task are not satisfactory. It can also be difficult to give positive feedback without sounding patronising. Learners have to consider their reactions to positive and negative feedback. They should be provided with guidance on how they can provide constructive feedback as well as receive constructive feedback in a positive way. They could practice giving and receiving feedback in pairs based on fictitious scenarios. They could then be provided with feedback on how well they provided and/or received the feedback.

It is suggested that learners are provided with a mini project to work on as a team so that they can apply the knowledge, skills and understanding that they have learnt.

Performance outcome 4

Learners need to understand that when reviewing the success of the project, they should also consider how well the team worked together and how things could be improved for future projects. They need to consider how well the team met the project outcome, and whether there were internal and/or external factors that influenced that outcome. Could these factors have been handled more effectively by the team, or did they play a part in the problems that the team had? Did everyone play their part as agreed, or was the team let down by others?

As a group, they could discuss the outcome of their mini project from PO3 and use their newfound skills in providing and receiving constructive feedback on how they carried out their individual roles and responsibilities, and how they felt other team members performed.

They could be asked to consider how they could improve their collaborative working as a team for future projects to prevent similar issues arising.

Synoptic assessment guidance

Synoptic assessment is a mandatory requirement of all AQA Tech-levels and this qualification has been designed with synoptic learning and assessment at its heart. Units link to each other, providing development on concepts and topics, reinforcing learning and skill development which enables learners to bring knowledge and skills from other units to contribute to the assessment of units as shown. Being able to work synoptically is the cornerstone of work-based problem-solving as learners make judgements on assessed prior learning in the context of new situations.

The mapping provided below shows where opportunities to undertake synoptic assessment can be found across the units of this qualification. Centres must ensure that these opportunities are built into their programmes of learning and assessment activities.

IT: Programming project #1

A local baker has successfully run a high street pastry and cake shop for a number of years. Due to the boom in artisan bakery, business is rising and they are considering expanding into a number of premises and offering their goods online through an e-commerce solution. A delivery service is also being considered with drivers being linked to the shops via specialist applications running on mobile devices, providing them with customer and order delivery information in real-time.

As part of this expansion, the baker will understandably require new computer systems which are securely connected to process customer sales, online orders and their accounts with suitable external links to permit internet access (world wide web and email etc). Effective management of the business will also rely on using the internet to survey competitors and promote the business, eg through a dynamic and interactive website which customers can use to check latest offers, make orders and book deliveries.

Administration users will need suitable client PCs, secured from threats, which will provide the typical office-style business software suites, sector-specific applications and safe access to the internet.

One of the key attractions of the business is being able to track customer buying trends and offer loyalty discounts to promote products which need a sales boost. This typically involves calculation of average sales orders, probabilities of different types of discounts (eg free delivery, vouchers etc) affecting shop sales.

Having worked successfully with your company before, the baker has asked you to draw up suitable plans for the installation of the hardware and bespoke software needed and the network infrastructure that is also required to meet their needs. Full documentation is required before a presentational handover occurs and it is assumed that further consultation may also be needed to refine details and answer further queries should they arise.

Although management and employees of this company have basic IT skills, documentation should be as inclusive as possible and use jargon sparingly whilst explaining technical details clearly and concisely.

You are part of the team assigned to complete this task and must complete it within the desired timeframe. How you manage your time, resources and priorities to finish this project is your decision.

Unit 1 – Fundamental principles of computing

This unit provides underpinning knowledge that supports learners' investigation into the client's needs, principally the hardware and software that is required to fulfil the project. For hardware this would include not only the computer systems required but the communication methods and external hardware (input, output and external backing storage devices).

In addition the learner will benefit from an appreciation of data, information and the data-processing cycle as these will assist their planning and problem-solving when considering how the client will use the computer systems to support their business needs (eg data entry, processing, storage and required outputs).

Key synoptic links:

Unit 1 – AO1: Identify the different types of computer
Unit 1 – AO2: Understand and evaluate the hardware requirements of a computer system
Unit 1 – AO3: Understand the software requirements of a computer system
Unit 1 – AO4: Understand how data is converted to information
Unit 1 – AO5: Demonstrate how computers process user requirements

IT: Programming project #1

Unit 2 – Computer programming

This unit supports learners to complete this project by introducing them to the different aspects of computer programming that may be used to meet the client's needs. Principally this will involve the use of different design tools, procedural algorithms, user interfaces and appropriate programming languages to collect customer, product and order data, process it in a secure and efficient manner and produce meaningful information.

Key synoptic links:

Unit 2 – AO1: Understand the different types of computer programming languages and their common uses
Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Unit 2 – AO4: Demonstrate the principles of good program practice and user interface design

Unit 3 – Website technologies

A key aspect of this project will involve the creation (or tailoring) of an e-commerce system designed to sell the baker's products through an online shopping experience. Many of the skills and techniques learners develop throughout the qualification contribute to this key project component, including an appreciation of the website technologies required and practical skills in markup languages, relational databases, and client-side and server-side scripting languages. Learners may also benefit from examining website vulnerabilities in order to counter them and ensure that their solution is robust and secure for customer use.

Key synoptic links:

Unit 3 – PO1: Understand the key features of website technologies
Unit 3 – PO2: Demonstrate the key features and functions of a markup language
Unit 3 – PO3: Demonstrate the key features of a style-sheet language
Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language
Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Unit 3 – PO6: Recognise vulnerabilities and counter threats to website technologies

Unit 4 – Mobile applications programming

The client's requirement for a specialist application that runs on a mobile device neatly incorporates the learner's understanding of programming mobile applications. It is likely that the majority of the learner's skills in this discipline will be exercised by this task including key features and functions, the correct design tools and techniques, practical use of mobile development tools and the ability to create and deploy the solution to the correct platform.

Key synoptic links:

Unit 4 – PO1: Understand the key features of mobile application development
Unit 4 – PO2: Apply the key features and functions of mobile application programming languages
Unit 4 – PO3: Demonstrate the ability to design mobile applications
Unit 4 – PO4: Demonstrate the proficient use of mobile development tools
Unit 4 – PO5: Create and deploy a working mobile application using cross-platform development

IT: Programming project #1

Unit 5 – Mathematics for computing

Maths features prominently throughout the project, from the basic arithmetic needed to calculate project costs and timeframes to the statistical analysis required to determine customer buying trends and calculate appropriate discounts.

Learners could also benefit from skills developed when working with different number systems as many hardware error codes are displayed in hexadecimal format. Programming logic invariably relies on Boolean logic when creating conditions for controlling selections and iterations. Techniques used to gather and interpret data should also provide vital support for learners when testing their project solution and supporting the success (or otherwise) of its outcome.

Key synoptic links:

Unit 5 – AO1: Understand and manipulate data in common number systems
Unit 5 – AO2: Understand and apply the foundations of computer logic
Unit 5 – AO3: Understand and interpret information using sets, sequences, series, probability and recursion
Unit 5 – AO4: Apply arithmetic expressions to abstract real world ideas
Unit 5 – AO5: Apply matrix methods to solve problems

Unit 6 – Event driven programming

Building software for the website, shop or driver's mobile application could leverage elements of event driven programming. If this is the case the learner should be able to demonstrate many of the concepts and techniques they have assimilated in this unit to solve the problem appropriately.

Key synoptic links:

Unit 6 – PO1: Understand the key features of event driven programming languages
Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 – PO3: Demonstrate the ability to design event driven applications
Unit 6 – PO4: Implement event driven applications to a professional standard

Unit 7 – Object oriented programming

Building software for the website, shop or driver's mobile application could leverage elements of object oriented programming. If this is the case the learner should be able to demonstrate many of the concepts and techniques they have assimilated in this unit to solve the problem appropriately.

Key synoptic links:

Unit 7 – PO1: Understand object oriented programming
Unit 7 – PO2: Design software solutions using an object oriented approach
Unit 7 – PO3: Implement object oriented applications to a professional standard
Unit 7 – PO4: Understand how to test and maintain programs
Unit 7 – PO5: Understand how to produce documentation

IT: Programming project #2

A local bowling centre is refitting its site as a result of progressive new ownership.

They will require new computer systems which are securely connected as a network to process customer bookings, provide computerised scoring and automated features on each lane, with suitable external links to permit internet access (world wide web and email etc) and data connectivity to other sites in the business to promote centre-based competitions and league tables.

Effective management of the business will also rely on using the internet to survey competitors and promote the business, eg through a dynamic and interactive website which customers can use to check latest offers, book sessions and track their playing statistics etc.

Administration users will also need suitable client PCs, secured from threats, which will provide the typical office-style business software suites, sector-specific applications and safe access to the internet. Game Marshals, who patrol the lanes, will need constant mobile communication and the ability to check game scores, reset lanes and clocks in real-time.

One of the key attractions of the business is the tracking software being used to monitor individual customer and team performances when bowling. This typically involves calculation of average performance levels, winning probabilities and playing trends.

Having worked successfully with your company before, the new owners have asked you to draw up suitable plans for the installation of the hardware and software needed and the network infrastructure that is also required to meet their needs. Full documentation is required before a presentational handover occurs and it is assumed that further consultation may also be needed to refine details and answer further queries should they arise.

Although management and employees of this company have basic IT skills, documentation should be as inclusive as possible and use jargon sparingly whilst explaining technical details clearly and concisely.

You are part of the team assigned to complete this task and must complete it within the desired timeframe. How you manage your time, resources and priorities to finish this project is your decision.

Unit 1 – Fundamental principles of computing

This unit provides underpinning knowledge that supports learners' investigation into the client's needs, principally the hardware and software that is required to fulfil the project. For hardware this would include not only the computer systems required but the communication methods and external hardware (input, output and external backing storage devices).

In addition the learner will benefit from an appreciation of data, information and the data-processing cycle as these will assist their planning and problem-solving when considering how the client will use the computer systems to support their business needs (eg data entry, processing, storage and required outputs).

Key synoptic links:

Unit 1 – AO1: Identify the different types of computer
Unit 1 – AO2: Understand and evaluate the hardware requirements of a computer system
Unit 1 – AO3: Understand the software requirements of a computer system
Unit 1 – AO4: Understand how data is converted to information
Unit 1 – AO5: Demonstrate how computers process user requirements

IT: Programming project #2

Unit 2 – Computer programming

This unit supports learners to complete this project by introducing them to the different aspects of computer programming that may be used to meet the client's needs. Principally this will involve the use of different design tools, procedural algorithms, user interfaces and appropriate programming languages to collect customer and game data, process it in a secure and efficient manner and produce meaningful information.

Key synoptic links:

Unit 2 – AO1: Understand the different types of computer programming languages and their common uses
Unit 2 – AO2: Analyse the tools and techniques for planning, design and development
Unit 2 – AO3: Evaluate the key features and techniques used in computer programming
Unit 2 – AO4: Demonstrate the principles of good program practice and user interface design

Unit 3 – Website technologies

A key aspect of this project will involve the creation (or tailoring) of an e-commerce system designed to book bowling sessions and lanes through an online service. Many of the skills and techniques learners develop throughout this qualification contribute to the creation of this key project component, including an appreciation of the website technologies required and practical skills in markup languages, relational databases, and client-side and server-side scripting languages. Learners may also benefit from examining website vulnerabilities in order to counter them and ensure that their solution is robust and secure for customer use.

Key synoptic links:

Unit 3 – PO1: Understand the key features of website technologies
Unit 3 – PO2: Demonstrate the key features and functions of a markup language
Unit 3 – PO3: Demonstrate the key features of a style-sheet language
Unit 3 – PO4: Demonstrate the key features and functions of a client-side scripting language
Unit 3 – PO5: Demonstrate the key features and functions of a server-side scripting language
Unit 3 – PO6: Recognise vulnerabilities and counter threats to website technologies

Unit 4 – Mobile applications programming

The client's requirement for a specialist application that runs on a mobile device neatly incorporates the learner's understanding of programming mobile applications. It is likely that the majority of the learner's skills in this discipline will be exercised by this task including key features and functions, design tools and techniques, practical use of mobile development tools and the ability to create and deploy the solution to the correct platform.

Key synoptic links:

Unit 4 – PO1: Understand the key features of mobile application development
Unit 4 – PO2: Apply the key features and functions of mobile application programming languages
Unit 4 – PO3: Demonstrate the ability to design mobile applications
Unit 4 – PO4: Demonstrate the proficient use of mobile development tools
Unit 4 – PO5: Create and deploy a working mobile application using cross-platform development

IT: Programming project #2

Unit 5 – Mathematics for computing

Maths features prominently throughout the project, from the basic arithmetic needed to calculate project costs and timeframes to the statistical analysis required to determine customer booking trends and calculate appropriate discounts.

Learners could also benefit from skills developed when working with different number systems, as many hardware error codes are displayed in hexadecimal format. Programming logic invariably relies on Boolean logic when creating conditions for controlling selections and iterations. Techniques used to gather and interpret data should also provide vital support for learners when testing their project solution and supporting the success (or otherwise) of its outcome.

Key synoptic links:

Unit 5 – AO1: Understand and manipulate data in common number systems
Unit 5 – AO2: Understand and apply the foundations of computer logic
Unit 5 – AO3: Understand and interpret information using sets, sequences, series, probability and recursion
Unit 5 – AO4: Apply arithmetic expressions to abstract real world ideas
Unit 5 – AO5: Apply matrix methods to solve problems

Unit 6 – Event driven programming

Building software for the website, shop or marshalls' mobile application could leverage elements of event driven programming. If this is the case the learner should be able to demonstrate many of the concepts and techniques they have assimilated in this unit to solve the problem appropriately.

Key synoptic links:

Unit 6 – PO1: Understand the key features of event driven programming languages
Unit 6 – PO2: Demonstrate the use of event driven language features and functions
Unit 6 – PO3: Demonstrate the ability to design event driven applications
Unit 6 – PO4: Implement event driven applications to a professional standard

Unit 7 – Object oriented programming

Building software for the website, shop or marshalls' mobile application could leverage elements of object oriented programming. If this is the case the learner should be able to demonstrate many of the concepts and techniques they have assimilated in this unit to solve the problem appropriately.

Key synoptic links:

Unit 7 – PO1: Understand object oriented programming
Unit 7 – PO2: Design software solutions using an object oriented approach
Unit 7 – PO3: Implement object oriented applications to a professional standard
Unit 7 – PO4: Understand how to test and maintain programs
Unit 7 – PO5: Understand how to produce document

Useful links and resources

Resources will very much depend on what the scenarios are that the centre wishes to use as this unit could be used for any of the pathways.

Book

- Major I, Greenwood A and Goodman M, *The definitive guide to project management*, ISBN-10 0273663976, Financial Times/Prentice Hall (2003).

Websites

- Freedcamp (free project-management software): freedcamp.com
- Project Libre (free project-planning software): projectlibre.org
- Project management lifecycle: mpmm.com/project-management-methodology.php

13 Externally set and marked examinations

13.1 Introduction

The Foundation Technical Level (360 GLH) in IT: Scripting and App Programming (TVQ01015) qualification units Y/507/6424 (Fundamental principles of computing) and F/507/6465 (Computer programming) are assessed via an externally set and marked AQA examination.

The Technical Level (720 GLH) in IT: Programming (TVQ01013) qualification unit Y/507/6469 (Mathematics for programmers) is assessed via an externally set and marked AQA examination.

External examinations are set by AQA (sometimes in collaboration with an employer or a professional body) and are sat by learners in a controlled examination environment, at a preset time and date and marked by AQA.

Examinations are available for externally assessed units in January and June and entries must be made in accordance with AQA's procedures.

Further information on how to make entries for examinations can be found in the AQA *Centre Administration Guide for Technical and Vocational Qualifications*.

13.2 Examination format and structure

Unit title	Fundamental principles of computing
Exam sessions	January and June
Duration	2 hours
Type of exam	Written exam A mixture of multiple choice, short answer and case study type questions.
Number of marks	80
Weighting of unit	25% of IT: Scripting and app programming (TVQ01015) 12.5% of IT: Programming (TVQ01013)

Unit title	Computer programming
Exam sessions	January and June
Duration	2 hours
Type of exam	Written exam A mixture of multiple choice, short answer and case study type questions.
Number of marks	80
Weighting of unit	25% of IT: Scripting and app programming (TVQ01015) 12.5% of IT: Programming (TVQ01013)

Unit title	Mathematics for programmers
Exam sessions	January and June
Duration	2 hours
Type of exam	Written exam A mixture of multiple choice, short answer and case study type questions.
Number of marks	80
Weighting of unit	12.5% of IT: Programming (TVQ01013)

13.3 Reasonable adjustments and special considerations

Information on the reasonable adjustments allowed for the external examinations within this qualification can be found in the AQA *Centre Administration Guide for Technical and Vocational Qualifications*.

13.4 Availability of past examination papers

Sample and past examination papers for this qualification are available from AQA.

14 External quality assurance

14.1 Overview

AQA's approach to quality assurance for this qualification is described within each unit specification.

External quality assurance for Tech-levels takes the form of verification and is concerned with maintaining the quality of assessment and checking that the assessment process has been undertaken appropriately by centre staff. It focuses on auditing the whole process and enables the head of centre, and all individuals involved in the assessment process, to understand what is required by them.

14.2 Quality assurance visits

When a learner is registered or a centre wants to submit work, this triggers a verification visit from an AQA external quality assurer (EQA).

Once a centre has registered learners, these visits will occur, as a minimum, every six months and will be face-to-face at a centre.

Our EQAs offer advice and guidance on any aspect of quality assurance in between formal visits, via telephone or email, and additional visits can be arranged.

These meetings will involve verifying that:

- all of the staff, resources, processes and procedures are still in place
- the centre is continuing to meet the approved centre criteria (those signed off during the initial centre approval visit)
- there is evidence of meaningful employer involvement in delivery.

A major part of the verification process is to check that the centre's policies and procedures (including internal standardisation minutes, record keeping, IQA/assessor records and materials) meet AQA's requirements and ensure valid and reliable assessment.

The EQA will look at a representative sample of learner work to verify that the results awarded by the centre are valid, as well as reviewing evidence of the activities that have been undertaken to standardise assessments.

These samples will be taken from different sites if the centre operates at more than one location, from different centre assessors or IQAs and at different stages of delivery – all samples will be selected by the EQA.

As part of the sample, the EQA will request examples of learner work at Pass, Merit and Distinction. This will also support the centre in their internal standardisation.

If centre assessment decisions are found to be inconsistent, adjustments can be made (at a learner and cohort level) or in more severe cases (where a fundamental inconsistency or non-compliance is identified), sanctions (from a Level 1: Action plan through to Level 4: Suspension of delivery) can be put in place.

14.3 Sanctions

Sanctions are used to help process improvement and are a way of protecting the validity of assessments or assessment decisions. We will only ever impose sanctions on a centre that are proportionate to the extent of the risk identified during the quality assurance process.

Sanctions can be applied at a learner, centre or centre staff level – and they can be at qualification or centre level and take the following form:

Level 1: Action point in EQA report

Level 2: Suspension of direct claims status (where applicable)

Level 3: Suspension of learner registration and/or certification

Level 4: Withdrawal of centre approval for a specific qualification.

Further information on levels and application of sanctions can be found in the AQA *Centre Administration Guide for Technical and Vocational Qualifications*.

15 Internal assessment and quality assurance

15.1 Overview

The Foundation Technical Level (360 GLH) in IT: Scripting and App Programming (TVQ01015) qualification units M/507/6476 (Website technologies) and Y/507/6486 (Mobile applications programming) are internally assessed by the centre.

For the Technical Level (720 GLH) in IT: Programming (TVQ01013) qualification units D/507/6487 (Event driven programming), K/507/6489 (Object oriented programming), and A/507/6464 (Industrial project) are internally assessed by the centre.

All assessment decisions that are made internally within a centre are externally quality assured by AQA.

AQA has worked with employers and professional bodies to produce guidance on what is the most appropriate form of assessment or evidence gathering for all internal centre assessment.

The most appropriate method of assessment (or evidence gathering) is detailed against each Unit. Should a centre wish to use an alternative method of assessment to that detailed, then justification must be provided during AQA quality assurance visits to the centre.

This justification needs to lay out why the centre feels their approach to assessment is more appropriate, efficient or relevant to the learner and/or subject and should be provided in writing to the AQA external quality assurer.

Centres should tailor the assessment to suit the needs of the learner, and internal assessments can take place at a time to suit the centre or learner.

Centres should take a best practice approach with learners being assessed through real life or work-based activity to generate the required evidence (see Section 8.1 Meaningful employer involvement).

15.2 Role of the assessor

The role of the assessor is to:

- carry out initial assessments of learners to identify their current level of skills, knowledge and understanding and any training or development needs
- review the evidence presented against the requirements of the qualification, to make a judgement on the overall competence of learners
- provide feedback to learners on their performance and progress. This feedback needs to give learners a clear idea of the quality of the work produced, where further evidence is required and how best to obtain this.

15.3 Assessor qualifications and experience

In order to assess learners working towards this qualification, assessors must:

- have appropriate knowledge, understanding and skills relevant to the units within this qualification
- have experience as a practitioner and/or teaching and training with significant experience of creating programmes of study in relevant subject areas
- undertake activities which contribute to their continuing professional development (CPD).

15.4 Applying portfolio assessment criteria

When assessing learner's work, the centre should consider the level of attainment in four broad areas:

1 the level of independence and originality
2 the depth and breadth of understanding
3 the level of evaluation and analysis
4 the level of knowledge, skills or competency demonstrated.

15.5 Authentication of learner work

The centre must be confident that a learner's work is their own. You must inform your learners that to present material copied directly from books or other sources such as the internet, without acknowledgement, will be regarded as deliberate deception. This also includes original ideas, as well as the actual words or artefacts produced by someone else.

Learners' work for assessment must be undertaken under conditions that allow the centre to authenticate the work. If some work is done unsupervised, then the centre must be confident that the learners' work can be authenticated with confidence – eg being sufficiently aware of an individual learner's standard and level of work to appreciate if the evidence submitted is beyond the level of the learner.

The learner is required to sign a declaration that the work submitted for assessment is their own. The centre will also countersign this declaration that the work was carried out under any specified conditions – recording details of any additional assistance. This must be provided with the learner's work for external quality assurance purposes.

Any assistance given to an individual learner beyond that given to the group as a whole, even if within the parameters of the specification, must also be recorded.

If some work is done as a part of a team, the centre must be confident that the learner's contribution to that team activity can be clearly identified and authenticated.

15.6 Tutor assistance and feedback

Whilst learners are undertaking assignment tasks, tutors must ensure that any assistance given, or offered as a result of a learner's question and/or request for help, does not compromise the learner's ability to independently perform the task in hand.

During assessment, tutors can give general feedback and support to learners, most notably, on the following:

- development of the required knowledge and skills underpinning the assignment at hand
- confirmation of the assessment criteria being assessed
- clarification of the requirements of the *Assignment brief*
- identification of assignment deadlines.

Tutors, however, must **not** assist learners directly and specifically with assignment tasks.

Tutors are not permitted to provide 'formative' feedback on learner's work, ie feedback, prior to submission for marking, on an assignment/task that will enable the learner to amend the assignment/task to improve it.

Once learner work has been submitted for marking, then tutors must give clear and constructive feedback on the criteria successfully achieved by the learner. Tutors should also provide justification and explanation of their assessment decisions. Where a learner has not achieved the performance criteria targeted by an assignment, then feedback should not provide explicit instructions on how the learner can improve their work to achieve the outstanding criteria. This is to ensure that the learner is not assisted in the event that their work is considered for resubmission.

15.7 Research and references

Where learners are required to undertake research towards the completion of a task, they should reference their research results in a way that is informative, clear and consistent throughout their work. We do not prescribe a specific way to organise references, but we expect tutors to discuss this with learners and identify a 'house style' that learners are then expected to use. Learners may include a bibliography of relevant sources on larger assignments where there has been significant research and there is value in recording all sources fully.

15.8 Role of the internal quality assurer

An internal quality assurer (IQA) must be appointed to ensure the quality and consistency of assessments within the centre. Each assessor's work must be checked and confirmed by an internal quality assurer.

The IQA must observe assessors carrying out assessments, review assessment decisions from the evidence provided and hold standardisation meetings with the assessment team to ensure consistency in the use of documentation and interpretation of the qualification requirements.

All assessment decisions made within a centre must be standardised to ensure that all learners' work has been assessed to the same standard and is fair, valid and reliable.

Evidence of all standardisation activity should be retained by the centre and could take the form of, for example, records of training or feedback provided to assessors, minutes of meetings or notes of discussions.

Our external quality assurers (EQAs) will always be happy to provide guidance and assistance on best practice.

Internal standardisation activity may involve:

- all assessors marking trial pieces of work and identifying differences in marking standards
- discussing any differences in marking at a training meeting for all assessors
- cross-moderation of work between assessors.

15.9 Internal quality assurer qualifications and experience

In order to internally quality assure the assessment of learners working towards this qualification, IQAs must:

- have appropriate knowledge, understanding and skills relevant to the units within these qualifications
- have experience as a practitioner and/or teaching and training with significant experience of creating programmes of study in relevant subject areas
- undertake activities which contribute to their continuing professional development (CPD).

15.10 Record keeping

The centre must be able to produce records that show:

- the assessor and IQA allocated to each learner
- the evidence assessed
- the dates of assessment and IQA
- details of internal standardisation activities of the assessor – (what, when and by whom)
- the grade awarded and rationale for this.

16 Resits, resubmissions and retakes

16.1 Note on terminology

Resits refer to learners taking further attempts at an examined/externally assessed unit.

Resubmissions refer to learners undertaking a second attempt at an internally assessed unit task/assignment prior to external quality assurance.

Retakes refer to learners undertaking a second attempt at an internally assessed unit after external quality assurance.

16.2 Rules on resits, resubmissions and retakes

Resits and retakes are permitted where a learner has either failed the requirements of the unit, or where they wish to improve on a grade awarded.

For certification purposes, AQA will recognise the best achievement by the learner and not the most recent.

Resitting an exam or external assessment
The learner is permitted **three** attempts (one initial and two resits) in relation to each examined/externally assessed unit of the specification.

Learners who have been awarded the Foundation qualification and have progressed to the full Technical Level are allowed to use the resit opportunities to go back and improve the grade achieved in the external assessment. Any improvement cannot be used to upgrade and reclaim the previously awarded Foundation qualification.

Resubmitting internal assessments
The learner is permitted **one** resubmission in relation to each internally assessed unit of the qualification, but only when the tutor believes the learner can achieve the outstanding criteria without further guidance. Any resubmission of work must be undertaken prior to external moderation.

Retaking internal assessments
The learner is permitted **one** retake in relation to each internally assessed unit of the qualification. This could mean the learner doing the entire unit work again, or simply correcting a task/assignment before the unit is again submitted for external moderation by AQA. With a retake, learners are not allowed a resubmission opportunity.

Any retake and/or resubmission of work must be completed within a defined and reasonable period of time following learner feedback of the initial assessment. Any work provided as evidence must be authenticated by the learner as their own.

17 Grading

17.1 Overview

Performance in all units is graded at Pass, Merit or Distinction. These unit grades are then converted into points and added together to determine the overall grade for the qualification.

The overall Foundation Technical Level in IT: Scripting and App Programming (TVQ01015) qualification is graded as P, M, D, D*.

The overall Technical Level in IT: Programming (TVQ01013) qualification is graded as PP, MP, MM, DM, DD, D*D, D*D*.

17.2 Internally assessed units

Centres must ensure that all assessment criteria in the unit are covered during the teaching and learning process so that learners can meet the requirements. Work should be assessed against the grading criteria provided within each unit.

- To achieve a Pass, a learner must have satisfied all Pass criteria.
- To achieve a Merit, a learner must achieve all of the Pass and all of the Merit criteria.
- To achieve a Distinction, a learner must achieve all of the Pass, Merit and Distinction criteria.

17.3 Externally assessed (examined) units

These units are assessed by AQA using a marks-based scheme. After the assessment has taken place and been marked, the grade boundaries are set by AQA. These grade boundaries are based on the level of demand of the assessment and learners' performance – all learners that took the assessment, not just those in your centre.

When the assessment results are shared with the centre, AQA will report on the grade boundaries.

Note: These grade boundaries may change for each assessment window according to the demand of the assessment – this is important to maintain standards across each window.

Learners' grades are converted into points.

17.4 Points per grade – unit level

Table 1 shows the points for each grade at a unit level.

Table 1: Points per grade

Grade	Internally/centre assessed unit
Pass	36
Merit	54
Distinction	72

17.5 Final grade for overall qualification

The total possible number of points that can be achieved for the Foundation Technical Level (360 GLH) in IT: Scripting and App Programming is 288.

The total possible number of points that can be achieved for the Technical Level (720 GLH) in IT: Programming is 576.

Points for overall qualification grade

Table 2: Foundation Technical Level (360 GLH) in IT: Scripting and App Programming (TVQ01015)

Grade	Points boundary
P	144
M	198
D	252
D*	270

Table 3: Technical Level (720 GLH) in IT: Programming (TVQ01013)

Grade	Points boundary
PP	288
MP	360
MM	396
DM	468
DD	504
D*D	522
D*D*	540

17.6 The 'Near Pass' rule

A near pass will be applied to an **externally assessed unit** for those learners who may fall just short of a pass grade. The unit grade will still be reported as a grade U, since the learner will not have performed to the minimum standard required for a Pass grade, but will qualify as a near pass for the purposes of determining the overall qualification grade.

The actual mark required to achieve the 'near pass' grade on an examined unit will change from year to year, depending on the grade boundaries that have been set. A learner will receive 27 points if they achieve a Near Pass.

A learner is allowed one Near Pass in an externally assessed unit in a Foundation Technical Level or up to two Near Pass results (six or eight unit Technical Level) or up to three Near Pass results (12 unit Technical Level).

All other eligibility requirements for achieving the qualification will remain the same:

- the total points score is above the Pass threshold; **and**
- all other units are passed

18 Administration arrangements

Full details of all of the administration arrangements relating to AQA Tech-levels can be found in the AQA *Centre Administration Guide for Technical and Vocational Qualifications*, including:

- how to apply for centre approval
- registration of learners
- dealing with recognition of prior learning (RPL)
- how to make examination entries
- dealing with missed examination dates
- examination invigilation arrangements
- how to make claims for certificates
- how to appeal against an assessment, IQA or EQA decision
- retention of learner work and assessment/IQA records
- dealing with potential malpractice or maladministration.

Details of all AQA fees can be found on the AQA website at aqa.org.uk

19 Appendix A: Transferable skills standards and guidance

19.1 Transferable skills – communication standards (oral)

Evidence must clearly show that the learner can:

CO1	Prepare a suitable presentation.	1.1 Research suitable topics for the presentation. 1.2 Research the most appropriate format for the presentation. 1.3 Plan the structure of the presentation. 1.4 Make use of any appropriate supporting materials and prepare any other resources needed for the presentation.
CO2	Use language, vocabulary, tone and style suited to the complexity of the topic and the context.	2.1 Use appropriate language and vocabulary. 2.2 Structure what is presented to help the audience follow the sequence of the main points and ideas. 2.3 Use tone and style of presentation appropriate to the audience and environment.
CO3	Use a variety of methods to engage the audience.	3.1 Provide examples to illustrate complex points. 3.2 Use relevant images from appropriate sources to illustrate key points. 3.3 Use at least one additional method to engage the audience.

Required evidence[8]

- Learner preparation evidence (planning notes, research).
- Learner presentation including all support materials.
- Assessor observation record*.

Learner guidance

The learner should consider the purpose, topic and audience as follows:

- the presentation should be eight minutes long to allow the learner to demonstrate the appropriate skills
- the presentation must always be contextualised within the technical subject content, and should not be simulated
- an audience of at least two or three people which may or may not include peers.

[8] For evidence marked with an asterix (*) recording documents are available for centres to use – please see aqa.org.uk/tech-levels/transferable-skills

CO1

There should be evidence showing that the learner has:

- researched the technical subject content of a complex matter
- selected information relevant to the purpose of the presentation
- planned how to structure the presentation
- planned to use a relevant image or images to illustrate key points of the presentation – that adds value to the overall presentation
- included one additional method to engage audience for example questioning, completion of handout, discussion etc.

CO2

Learners should:

- give a well-structured delivery and must clearly highlight the main points of their presentation using tone, gesture or expression
- use appropriate vocabulary suited to the audience and environment.

CO3

Learners must:

- give examples to explain ideas
- make effective use of an image or images and other support materials to engage the audience and to illustrate key points, for example through use of video clips, explanatory notes or other technically related activities.

Tutor guidance

- Tutors should use an observation record to support their assessment.
- Tutors should ensure that those observing are familiar with the observation record content and purpose.
- The presentation may be delivered through spoken communication or using sign language.
- Tutors should look for fitness of purpose and styles of presentation. Brief notes may be used as a prompt, but learners should not rely on them entirely.

19.2 Transferable skills – communication standards (written)

Evidence must clearly show that the learner can:

CW1	Select appropriate formats for presenting information as a report.	1.1 Decide on the most appropriate format for the technical report. 1.2 Plan the structure of the technical report. 1.3 Make use of any appropriate supporting materials and prepare any other resources needed for the technical report.
CW2	Select and use an appropriate style and tone to suit their audience.	2.1 Use appropriate language and vocabulary. 2.2 Structure the technical report to help the audience follow the sequence of the main points and ideas. 2.3 Use tone and style appropriate to the intended recipient(s).
CW3	Organise material coherently, to suit the length, complexity and purpose of their technical report, proofread and where necessary, re-draft documents.	3.1 Spell, punctuate and use grammar accurately. 3.2 Make their meaning clear. 3.3 Use relevant images from appropriate sources to illustrate key points. 3.4 Proofread their technical report. 3.5 Obtain feedback and amend technical report accordingly.

Evidence required[9]

- A learner technical report of at least 1,000 words excluding support materials.
- An assessor recording form*.

Learner guidance

The learner should:

- produce a technical report about a complex subject which must be at least 1,000 words long
- include subject matter, which may well have a number of strands that is challenging to the individual learner in terms of the ideas it presents.

[9] For evidence marked with an asterix (*) recording documents are available for centres to use – please see aqa.org.uk/tech-levels/transferable-skills

CW1

It is essential that learners know how to:

- organise their technical report
- link paragraphs in various ways
- use features, such as indentation and highlighting, to suit different types of documents.

CW2

Learners should know how to:

- produce a technical report that takes account of the vocabulary, tone and techniques normally used when producing documents for particular purposes and different recipients
- write with confidence and with the appropriate degree of formality.

CW3

In supporting key points:

- images that could be used include: graph, sketch, picture or material taken from a presentation
- learners should know how to check their work to ensure that spelling, punctuation and grammar are accurate
- learners should know how to write grammatically correct sentences, including correct use of a variety of verb tense, form and person (for example passive voice); spell accurately, complex, irregular and technical words and use punctuation effectively for example bullet points, semicolon, colon, apostrophes) to ensure their meaning is clear.

Tutor guidance

For the technical report produced, assessors should look for evidence that the learner has:

- selected an appropriate format for report
- organised relevant information using a clear and coherent structure
- used technical vocabulary when appropriate
- ensured that text is legible with accurate use of spelling, grammar and punctuation.

The learner should not be penalised for one or two errors providing meaning is still clear.

19.3 Transferable skills – problem-solving standards

Evidence must clearly show that the learner can:

PS1	Identify a problem and the tools and techniques that could be used to explore the problem.	1.1 Identify, analyse and describe the problem. 1.2 Identify a variety of tools and techniques which could be used to explore the problem. 1.3 Plan how you will investigate the problem highlighting which tools and techniques will be used.
PS2	Implement both the plan to investigate the problem and the plan to solve the problem.	2.1 Implement the plan for investigating the problem and seek support and feedback from others as necessary. 2.2 Record and analyse the results of the investigation. 2.3 Identify the solution(s) to solve the problem. 2.4 Plan the steps to be taken in order to solve the problem, identifying any risks, and implement the solution.
PS3	Check if the problem has been resolved and review the approach to tackling problems.	3.1 Check whether the problem has been resolved/ solved. 3.2 Analyse the results and draw conclusions on the success of the problem-solving process. 3.3 Review the approach to tackling/solving the problem, including whether other approaches might have proved more effective.

Evidence required[10]

- Explore/plan* – to be completed by the learner.
- Do* – to be completed by the assessor.
- Review* – to be completed by the assessor.

Learner guidance

The learner must demonstrate:

- a systematic approach to tackling problems, including identifying which is the most appropriate method, then developing a plan and implementing it
- how they went about the problem-solving process.

Evidence should be on individual performance. A group approach to problem-solving does not allow learners to achieve specific elements of the standards.

Activities must always be in relation to the core subject content and should not be simulated.

Effective definition of the problem will help the learner tackle it systematically and produce valid evidence. Tutors may discuss with learners the most appropriate definition of the problem and what sort of results might be expected so the learner is clear on what would show that the problem had been solved.

[10] For evidence marked with an asterix (*) recording documents are available for centres to use – please see aqa.org.uk/tech-levels/transferable-skills

PS1

Learners should:

- recognise, identify and describe the main features of the problem
- identify how they will explore the problem and the tools and techniques they will use
- use a variety of methods for exploring the problem.

PS2

Learners should:

- obtain approval to implement their plan from an appropriate person, which could be the tutor or supervisor
- make effective judgements, based on feedback and support available, when putting their plan into action
- check their plan regularly for progress and revise it accordingly.

PS3

Learners should:

- use an appropriate method for checking if the problem has been solved. For example if a learner designed a procedure or process for improving a system that records information, they would need to test this out and report back on their findings
- know how to describe the results in detail and draw conclusions on the success of their problem-solving skills
- reflect back on the process considering areas such as:
  - did they spend enough time considering the features of the problem?
  - were they effective in planning action points to tackle the problem?
  - did they take a logical approach to checking if the problem had been solved/resolved?

In some circumstances, achievement of the standard may be possible even if the problem has not been solved or resolved, especially if factors were outside of their control, and the learner was able to demonstrate the process of tackling the problem.

Tutor guidance

- Tutors should check problem-solving implementation planning.
- Tutors may be required to provide a witness statement in support of evidencing the processes.

19.4 Transferable skills – research standards

Evidence must clearly show that the learner can:

R1	Design a research study.	1.1 Identify possible topics for research. 1.2 Choose one topic, identifying appropriate objectives for detailed research, and plan how to carry out the research. 1.3 Select a variety of resources to gather relevant information and identify appropriate methods and techniques to carry out the research.
R2	Conduct data collection and analysis.	2.1 Collect data using the appropriate methods to test the hypotheses/theories. 2.2 Carry out an appropriate analysis of the data. 2.3 Draw appropriate conclusions that are supported by the evidence from the data analysis.
R3	Present findings of the research and evaluate the research activities.	3.1 Prepare and present results of research. 3.2 Present the information in a clear and appropriate format adapted to the needs of the audience. 3.3 Seek feedback and use it to support own evaluation of research skills.

Required evidence[11]

- Plan* – to be completed by the learner.
- Do* – to be completed by the assessor.
- Review* – to be completed by the assessor.
- Results of research.

Learner guidance

The learner should demonstrate they can:

- identify clear and appropriate objectives for the research study
- plan and carry out research activities with the particular objectives in mind
- design their research study in a systematic way
- present their findings as well as evaluating their research skills and activities
- be clear about the objectives of the research study, for example to assess the positive and negative impact of digital photography on sports journalism to predict future trends
- identify sources, methods and strategies they plan to use to investigate the topic
- carry out the research within a clearly defined structure, with a measure of complexity that should be reflected in the breadth and nature of the research objectives
- undertake the analysis required to make the best use of information/data and the requirement to give a clear justification for their conclusions
- make different research methodologies.

Activities must always be contextualised within the core subject content, and should not be simulated.

[11] For evidence marked with an asterix (*) recording documents are available for centres to use – please see aqa.org.uk/tech-levels/transferable-skills

RS1

The learner should explore:

- a variety of possible topics to research and should spend time deciding on clear and measurable objectives when designing their research study
- objectives and discuss and agree them with a tutor or supervisor
- a wide variety of sources when gathering their information
- the use of at least three different types of resource
- one source that is primary (gathered by the learner), for example, interview, questionnaire, survey, rather than from secondary for example encyclopaedia, interpretations of original material.

The learner should produce a plan detailing how they will carry out the research.

RS2

The learner should:

- keep a record of the sources used
- independently collect information including data
- analyse information collected and identify information and data most relevant to their research objectives.

RS3

When presenting their findings, learners should:

- use a format that is most appropriate to the content in terms of audiences, subject matter and research objectives
- communicate research findings clearly
- seek feedback from appropriate people
- show how they have used this feedback to help evaluate their research skills
- evaluate their research activities addressing all aspects including identifying the research objectives, collecting and analysing data and/or information, and recording, presenting and explaining findings.

Tutor guidance

- Tutors should agree research objectives with learner.
- Tutors should check that different types of resource have been used.

19.5 Transferable skills – teamwork standards

Evidence must demonstrate the learner can:

TW1	Plan the work with others.	1.1 Agree realistic objectives for working together and what needs to be done in order to achieve them. 1.2 Share relevant information to help agree team roles and responsibilities. 1.3 Agree suitable working arrangements with other team members.
TW2	Develop and maintain cooperative ways of working towards agreed objectives checking progress on the way.	2.1 Organise and complete own tasks efficiently to meet responsibilities. 2.2 Seek effective ways to develop cooperation such as ways to resolve conflict and maintain open communication. 2.3 Share accurate information on progress and agree changes where necessary to achieve objectives.
TW3	Review working with others and agree ways of improving collaborative work in the future.	3.1 Agree the extent to which working with others has been successful and objectives have been met. 3.2 Identify factors, including their own role, in influencing the outcome. 3.3 Provide details of how they could improve working with others in the future, including interpersonal skills.

A group/team is defined as **three or more** people (eg peer, co-worker) who are working towards shared objectives. It is not acceptable for tutors/assessors to be part of the team. The nature of the teamworking should reflect the sector in which the qualification sits, eg engineering, business or IT.

Required evidence[12]

- Plan*.
- Do*.
- Review*.
- Minutes of meetings.
- Witness statement.
- Peer statements.

Learner guidance

Meeting the standard will confirm that the learner has:

- demonstrated the ability to work cooperatively with others
- be clear about the objectives the team or group is working towards and their own responsibilities
- planned and carried out the work supporting others, reviewing outcomes and suggesting ways of improving work with others.

Activities must always be contextualised within the core subject content, and should not be simulated.

[12] For evidence marked with an asterix (*) recording documents are available for centres to use – please see aqa.org.uk/tech-levels/transferable-skills

TW1

As part of the initial team planning meeting the learner should:

- offer suggestions and listen to others to agree realistic objectives, prioritise tasks and identify resources and timescales
- be clear about their own responsibilities and the areas of work for which they are answerable to others
- produce a plan showing what needs to be done by the team clarifying own responsibilities and arrangements for working with others in the team.

TW2

Learners should take responsibility for:

- organising their own work to meet the agreed deadlines
- the use of correct and appropriate techniques and approaches when carrying out tasks
- actively looking for ways to develop and support cooperative working, including helping to deal with conflict and taking a lead role in anticipating the needs of others
- considering the rights and feeling of others
- ensuring at least one team progress meeting should be held before the final review meeting.

TW3

During the team review meeting learners should:

- provide information about their own contribution to the work of the team ie what did they do and how did they interact with other members of the group
- explain how improved inter-personal skills could contribute to more effective collaboration in the future (for example 'I should listen more carefully when negotiating activities/tasks')
- identify improvements they could make in managing tasks (for example 'I could have been better organised with notes at team meetings').

Tutor guidance

Tutors are encouraged to support the evidence process by completing a witness statement.

Realising potential

Get help and support

Visit our website for information, guidance, support and resources at aqa.org.uk/tech-levels

E: techlevels@aqa.org.uk

T: 0800 085 0391

aqa.org.uk

G02013h