Pearson BTEC

L3

Pearson BTEC
Level 3 Technical Occupational Entry for

Software Development Technicians (Certificate)

Specification

First teaching from August 2026

First certification from 2027

Issue 1

Qualification Number: 610/6199/2

Pearson
BTEC

Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate)

Specification

BTEC Technical Occupational Entry qualification
First registration 2026

About Pearson

We are the world's leading learning company operating in countries all around the world. We provide content, assessment and digital services to students, educational institutions, employers, governments and other partners globally. We are committed to helping equip students with the skills they need to enhance their employability prospects and to succeed in the changing world of work. We believe that wherever learning flourishes so do people.

References to third-party material made in this specification are made in good faith. Pearson does not endorse, approve or accept responsibility for the content of materials, which may be subject to change, or any opinions expressed therein. (Material may include textbooks, journals, magazines and other publications and websites.)

All information in this specification is correct at time of publication.

Publication code: VQ000382

All the material in this publication is copyright
© Pearson Education Limited 2025

Welcome

With a track record built over 30 years of student success, BTEC qualifications are widely recognised and respected. They provide progression to the workplace either directly or via study at higher levels. Recent data has shown that 1 in 5 adults of working age in the UK has a BTEC qualification.

Why choose BTEC Technical Qualifications?

BTEC Technical Qualifications enable students to develop a purposeful and coherent combination of knowledge, skills and behaviours to confidently enter or progress into employment at entry level in occupations that are recognised and demanded by employers.

The qualification, which is based on the occupational standards published by the Institute for Apprenticeships and Technical Education (IfATE), embodies a fundamentally student-centred approach to the curriculum, with a flexible, unit-based structure and an approach to learning and assessment that:

- provides students with meaningful and occupationally relevant learning experiences
- engages and motivates students to achieve as assessments can be focused on individual student needs and can be achieved as they progress through the qualifications
- promotes self-directed learning through the clarity and transparency of the standards to be achieved
- makes the qualifications accessible to a wider range of students, including part-time and adult students.

In developing these qualifications, we have collaborated with employers to ensure that the qualifications meet the current and emerging needs of industry. We have also worked with colleges and training centres to ensure that the qualifications meet their needs and those of their students.

We are providing a range of support to ensure that students and their tutors have the best possible experience during their course. Further information is provided on the qualification pages of our website.

A word to students

This qualification will require commitment and hard work. You will have to complete the learning for the required range of units, be organised and complete your assessments, which may include practical work-based activities, projects and vocational assignments. But you can feel proud to achieve a BTEC Technical Occupational Entry Qualification as you can be confident in your readiness to advance your career in your chosen occupation.

Good luck, and we hope you enjoy your course.

Contents

1 Introducing the qualification

What are Level 3 Technical Occupational Entry Qualifications?

Level 3 Technical Occupational Entry Qualifications are qualifications that are at Level 3 on the Regulated Qualifications Framework (RQF) and are designed to deliver the knowledge, skills and behaviours needed to enter the workplace. They can be delivered through a combination of classroom and work-based learning and assessment.

These qualifications are based on occupational standards designed by employers and published by the Institute for Apprenticeships and Technical Education (IfATE), which also approves the qualifications. IfATE has specified different categories under which Level 3 Technical Qualifications can be approved based on their scope and purpose. Detailed information about these categories can be found on IfATE's website.

Qualification purpose

The Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate) enables students to develop a purposeful combination of knowledge, skills and behaviours to enter employment as a software development technician at entry level. The qualification provides a strong foundation for them to achieve full occupational competence with further training and development in the workplace.

The qualification is designed to meet the needs of students (19+) and provides progression to employment in an occupation that is recognised and demanded by employers.

The qualification will:

- develop students' ability and confidence to apply the knowledge, skills and behaviours when carrying out the occupational duties and functions of a software development technician and to meet entry-level competence
- develop transferable skills and professional behaviours, such as managing own time to meet deadlines, managing stakeholder expectations, having a structured approach to prioritising tasks and reviewing own development needs, that are essential to personal effectiveness in a software development role
- develop knowledge and understanding of software development within industry, including the associated UK legislation and industry standards, software development and project life cycles, programming fundamentals, together with the skills needed to develop programmed solutions and the fundamentals of digital technologies, including threats to digital systems and emerging technologies
- provide opportunities for students to achieve a nationally recognised occupational qualification to support them in taking the next step in their career journey

- provide employers with reliable evidence of students' attainment against the Software Development Technician occupational standard and their readiness to enter employment in this sector.

The qualification can be taken on a part-time or full-time basis to meet the needs of older 19+ students.

Employer engagement and validation

In developing the Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate), we have worked closely with a dedicated panel of employers from a range of different types of organisations, which have:

- validated the demand for the qualification and confirmed that it is occupationally relevant and meets the current and emerging needs of industry
- confirmed that students will have an appropriate combination of knowledge, skills and behaviours relevant to the occupational standard that attests to their readiness to enter into employment in the related occupation at an entry level.

Progression opportunities

Students who achieve the Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate) will most likely progress into specific employment at entry level in roles such as business analyst, data analyst, DevOps engineer and software developer.

 © Pearson Education Limited 2025

2 Qualification summary and key information

Qualification title	Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate)
Qualification Number (QN)	610/6199/2
Regulation start date	11/08/25
Operational start date	01/08/2026
Approved age ranges	19+
Total qualification time (TQT)	290
Guided learning hours (GLH)	230
Assessment	Internal assessment demonstrating evidence of entry-level competence. Required methods of assessment and evidence will be described in the unit. Please see *Section 5 Assessment requirements*
Grading information	The qualification and units are graded Pass/Fail.
Entry requirements	No prior knowledge, understanding, skills or qualifications are required before students register for this qualification.
Funding	This qualification is eligible for funding as a Technical Occupational Entry qualification. Information about funding can be found on the Find a learning aim database.

3 Qualification structure

Pearson BTEC Level 3 Technical Occupational Entry for Software Development Technicians (Certificate)

The requirements outlined in the table below must be met for Pearson to award the qualification.

Minimum number of units that must be achieved	3

Unit number	Mandatory units title	Level	Guided learning hours
1	Software Development within Industry	3	60
2	Programming Solutions	3	135
3	Digital Technologies	3	35

4 Delivery

Occupational entry-level competence

This qualification is designed to be delivered in post-16 learning contexts. Delivery should focus on students' ability to use knowledge, skills and appropriate behaviours in the workplace. Links with the workplace should be encouraged throughout.

5 Assessment requirements

Language of assessment

Students must use English only during the assessment of this qualification.

A student taking the qualification(s) may be assessed in British Sign Language where it is permitted for the purpose of reasonable adjustment.

Further information on the use of language in qualifications is available in our *Use of languages in qualifications policy*, available on our website, qualifications.pearson.com.

Internal assessment

Internally assessed units are subject to standards verification. This means that centres set and mark the final summative assessment for each unit, drawing on mandatory evidence requirements and support that Pearson provides.

To pass each internally assessed unit, students must:

- achieve all the specified learning outcomes
- satisfy all the assessment criteria by providing sufficient and valid evidence for each criterion
- meet any prescribed evidence requirements for a unit, i.e. units may mandate practical demonstration of skills in a workplace or simulated environment
- prove that the evidence is their own.

Centres must ensure:

- assessment is carried out by tutors with relevant expertise in both the occupational area and assessment. For the occupational area, this can be evidenced by a relevant qualification or current (within three years) occupational experience that is at an equivalent level or higher than this qualification. Assessment expertise can be evidenced by qualification in teaching or assessing and/or internal quality assurance or current (within three years) experience of assessing or internal verification
- internal verification systems are in place to ensure the quality and authenticity of students' work, as well as the accuracy and consistency of assessment. These must include processes for detecting and reporting student malpractice such as plagiarism (including AI misuse), copying and collusion.

Students who do not successfully pass an assessment are allowed to resubmit evidence for the assessment.

Levels of control in internal assessment

Task setting

Centres are able to design tasks that address the assessment criteria within a unit. Restrictions on task setting such as mandatory forms of evidence requirement, or restrictions surrounding the context of assessment will be stated within the qualification unit and any accompanying authorised assignment brief(s). Although task setting is characterised as low control, Pearson applies quality assurance methodology to ensure that centre systems are in place to develop and assure high-quality assessments for students. The authorised assignment brief serves as a model for the expected presentation of a unit assessment. Further guidance and references are provided in *Section 9 Quality assurance of centres*.

Task taking

Centres must be able to authenticate the student response to the assessment. Supervision may not always be appropriate, if for example a student is gathering evidence for an assessment that is then prepared in a classroom environment. However, centres must be assured that students produce their own response to assessment criteria. This may require supervision of students in writing up outcomes to ensure they do not use text-generative AI software.

In the assessment for unit 2, students should make use of a generative AI model to create short snippets of code that address specific areas of functionality, i.e. a specific function/process that the student would integrate into a larger code.

The AI model should not be used to create a single overarching solution to the entire brief.

The chosen AI model should allow students to demonstrate how they have entered specific prompts to create an output that they can then review in terms of how far it meets the specific needs of the identified functionality.

Pearson does not specify which generative AI the centres should use; however, the nature of the task would suggest the use of a natural language processing model.

Task marking

Centre assessors and tutors will mark the student assessment response, using Pearson BTEC assessment/grading criteria and the guidance we provide in the specification and surrounding process, and training we provide supporting our quality assurance process. Pearson will quality assure the processes that centres use to ensure the standard of marking outcomes. We operate a risk-based quality assurance process ensuring that new centres, centres with large cohorts and centres with other risk factors get the support they need to ensure students achieve the outcome they have worked for.

Authorised Assignment Briefs (AABs)

Pearson has produced an Authorised Assignment Brief (AAB) for each unit to support centres in the assessment of this qualification. The AAB is published separately to the specification on the Pearson website both as a PDF and word document and sets out a recommended assessment approach. If students meet the requirements of the published AAB then they will meet the requirements set out in the assessment criteria.

It is strongly recommended that centres refer to the AAB for each unit.

Centres can use an AAB in three ways:

- As the assignment brief for students, without changing it.
- As a guide to the level of evidence that is required from students, while choosing to write their own assessment brief.

As a basis for their own assessments, taking the AAB and amending in line with a particular context or local need.

6 Centre recognition and approval

Centres must have approval prior to delivering any of the units in this qualification.

Centres that have not previously offered BTEC qualifications need to apply for, and be granted, centre recognition as part of the process for approval to offer individual qualifications.

Guidance on seeking approval to deliver BTEC qualifications is given on our website.

Approval agreement

All centres are required to enter into an approval agreement with Pearson, in which the head of centre or principal agrees to meet all the requirements of the qualification specification and to comply with the policies, procedures, codes of practice and regulations of Pearson and relevant regulatory bodies. If centres do not comply with the agreement, this could result in the suspension of certification or withdrawal of centre or qualification approval.

Centre resource requirements

As part of the approval process, centres must make sure that the resource requirements below are in place before offering the qualification:

- appropriate physical resources (for example IT, learning materials, teaching rooms, workshops, simulated workplaces and access to work experience where appropriate) to support the delivery and assessment of the qualification
- suitable staff for delivering and assessing the qualification (see *Section 5 Assessment requirements*)
- systems to ensure continuing professional development (CPD) for staff delivering and assessing the qualification
- health and safety policies that relate to the use of equipment by students
- internal verification systems and procedures (see *Section 5 Assessment requirements*)
- any unit-specific resources stated in individual units.

7 Access to qualifications

Access to qualifications for students with disabilities or specific needs

Equality and fairness are central to our work. Our *Equality, diversity and inclusion policy* requires all students to have equal opportunity to access our qualifications and assessments, and that our qualifications are awarded in a way that is fair to every student.

We are committed to making sure that:

- students with a protected characteristic (as defined by the Equality Act 2010) are not, when they are taking one of our qualifications, disadvantaged in comparison to students who do not share that characteristic
- all students achieve the recognition they deserve from their qualification and that this achievement can be compared fairly to the achievement of their peers.

For students with disabilities and specific needs, the assessment of their potential to achieve the qualification must identify, where appropriate, the support that will be made available to them during delivery and assessment of the qualification.

Centres must deliver the qualification in accordance with current equality legislation. For full details of the Equality Act 2010, please visit www.legislation.gov.uk.

Reasonable adjustments and special consideration

Centres are permitted to make adjustments to assessment to take account of the needs of individual students. Any reasonable adjustment must reflect the normal learning or working practice of a student in a centre or a student working in the occupational area.

Centres cannot apply their own special consideration – applications for special consideration must be made to Pearson and can be made on a case-by-case basis only.

Centres must follow the guidance in the Pearson document *Supplementary guidance for reasonable adjustments and special consideration in internal assessments.*

8 Recognising prior learning and achievement

Recognition of Prior Learning (RPL) considers whether a student can demonstrate that they can meet the assessment requirements for a unit through knowledge, understanding or skills they already possess and so do not need to develop through a course of learning.

Pearson encourages centres to recognise students' previous achievements and experiences in and outside the workplace, as well as in the classroom. RPL provides a route for the recognition of the achievements resulting from continuous learning.

RPL enables recognition of achievement from a range of activities using any valid assessment methodology. If the assessment requirements of a given unit or qualification have been met, the use of RPL is acceptable for accrediting a unit, units or a whole qualification. Evidence of learning must be valid, authentic, reliable, current and sufficient.

Further guidance is available in our policy document *Recognition of prior learning policy and process*, available on our website.

9 Quality assurance of centres

For the qualification in this specification, the Pearson quality assurance model will consist of the following processes.

Centres will receive at least one visit from our Standards Verifier, followed by ongoing support and development. This may result in more visits or remote support, as required to complete standards verification. The exact frequency and duration of Standards Verifier visits/remote sampling will reflect the level of risk associated with a programme, taking account of the:

- number of assessment sites
- number and throughput of students
- number and turnover of assessors
- number and turnover of internal verifiers
- amount of previous experience of delivery.

Following registration, centres will be given further quality assurance and sampling guidance.

For further details, please see the work-based learning quality assurance handbooks, available in the support section of our website:

- *Pearson Work-based Learning Centre Guide to Quality Assurance*
- *Pearson Work-based Learning Delivery Guidance & Quality Assurance Requirements.*
- Support is also available on our work based learning quality assurance webpages Quality Assurance – Work-based Learning (WBL) | Pearson qualifications

10 Units

This section of the specification contains the units that form the assessment for the qualification.

It is compulsory for students to meet all learning outcomes and the assessment criteria to achieve a grade. The assessment criteria determine the standard required. Content is compulsory unless it is provided as an example and is therefore marked 'e.g.,'. All compulsory content must be delivered, but assessments may not cover all content.

Where legislation is included in delivery and assessment, centres must ensure that it is current and up to date.

Unit 1: Software Development within Industry

Level: 3

Unit type: Mandatory

Assessment type: Internal

Guided learning hours: 60

Unit introduction

The primary role of a software developer is to design, create, test and implement code for a variety of platforms. A software developer helps organisations, in a range of industries, to meet their business needs by developing effective, efficient, robust and often innovative applications. Most industries are increasingly reliant on digital platforms, and understanding why, how and when code for these platforms should be created or updated is a key aspect of this role.

In this unit, students will gain an understanding of how software is developed and the stages that must be followed to develop software.

Students will research legislation and industry standards that apply to software development projects. They will examine common developmental processes used in the computing industry, including software development and project life cycles, and the importance of having effective software development teams.

Learning outcomes and assessment criteria

Learning outcomes	Assessment criteria
1. Explore current UK legislation and industry standards that apply to software development	1.1 Summarise UK legislation, its scope and its enforcement 1.2 Summarise the scope and importance of current industry standards related to software development 1.3 Explain how security controls are used and maintained to ensure confidentiality, integrity and availability when developing software
2. Investigate the software development life cycle	2.1 Describe the steps involved during each stage of the software development life cycle 2.2 Describe the key roles and responsibilities within the software development life cycle 2.3 Explain how user acceptance criteria and Key Performance Indicators (KPIs) are used within the software development life cycle 2.4 Explain the benefits of the software development life cycle for software developers 2.5 Describe the similarities and differences between different software development methodologies
3. Investigate the project life cycle	3.1 Describe the steps involved during each stage of the project life cycle 3.2 Describe the key roles and responsibilities within the project life cycle 3.3 Explain the benefits of the project life cycle for developing software
4. Understand the principles of effective teamworking when producing computer programs	4.1 Explain how effective teamwork contributes to the delivery of software development projects 4.2 Discuss software development collaboration tools that enable developers to work together effectively 4.3 Explain how different communication methods can be used to communicate with different audiences

Unit content

What needs to be learned

Learning outcome 1: Explore current UK legislation and industry standards that apply to software development

Students should learn current legislation and standards relating to software development. They should keep up to date with changes to legislation and industry standards, especially regarding the increased use of artificial intelligence and its use in software development.

1A UK legislation, its scope and enforcement, including the obligations of organisations and software development professionals

- The role of the Information Commissioner's Office (ICO) as an independent regulatory authority.
- Data Protection Act – General Data Protection Regulation (GDPR):
  - Definition of personal data
  - Legal sharing of data (e.g. obtaining consent, sharing only data that is necessary for the intended purpose, ensuring data is shared securely, requirements for sharing data outside of the European Economic Area)
  - Legal marketing consent (e.g. knowing when consent is required, limitation of consent for specific purposes, age of person giving consent, withdrawal of consent).
- Regulation of Investigatory Powers Act:
  - Amendments and expansions – Investigatory Powers Act, Data Retention and Acquisition Regulations
  - Investigatory Powers Commissioner's Office (IPCO).
- Human Rights Act:
  - Human Rights Act reform, Bill of Rights.
- Computer Misuse Act:
  - Amendments and expansions – Part 5 of the Police and Justice Act, Part 2 of the Serious Crime Act.
- Freedom of Information Act.
- Equality Act 2010:
  - Protected characteristics to protect individuals from discrimination (e.g. age, disability, gender identity, marital status, pregnancy and maternity, race, religion and sexual orientation)
  - Accessibility regulations.

What needs to be learned

- Intellectual Property Rights:
  - Patents, trademarks and copyright
  - Assets protected
  - Duration of rights
  - Penalties for non-compliance.

1B Industry standards, scope and compliance, including the obligations of organisations and software development professionals

- ISO 12207/15288 – Software life cycle procedures.
- ISO 29119 – Software testing.
- ISO 5055 – Internal structure of a software product.

1C SDLC security control guidelines to ensure confidentiality, integrity and availability

- Secure programming standards.
- Secure development environment.
- Software deployment to prevent security.
- Test environments security controls.

Learning outcome 2: Investigate the software development life cycle

2A Common stages involved during the software development life cycle

Students should develop an understanding of the main stages commonly found in software development life cycles. They should understand how each stage contributes to the overall software and how the stages interconnect within the larger context of software development.

- Planning and requirement analysis:
  - Application goals
  - Documentation:
    - Software configuration management plan
    - Software quality assurance plan
    - Project plan
    - Resource allocation
    - Cost estimation
  - Activities carried out:
    - Set application goals
    - Choose programming language based on organisational policy, expandability, availability of trained staff, costs, reliability.

What needs to be learned

- Defining requirements:
  - Documentation:
    - Requirements document (user acceptance criteria, Key Performance Indicators (KPIs))
    - Updated project plan
    - Requirements traceability matrix (RTM)
  - Activities carried out:
    - Questionnaires
    - Interviews
    - Focus groups
    - Observations
    - Studying documentation.
- Designing the product:
  - Documentation:
    - Interface design – wireframes
    - Security issues
    - Proof of concept
    - Design algorithms
    - Hardware compatibility
    - Operating systems compatibility
  - Updated project plan
  - Updated requirements traceability matrix (RTM)
    - Design elements formally associated with requirements.
- Developing the product:
  - Documentation:
    - Software code
    - Implementation map
    - Software help manual
    - Test plan
    - Updated project plan
    - Updated requirements traceability matrix (RTM) – development elements formally linked with requirements
  - Prototype process:
    - Determine basic requirements
    - Develop initial prototype
    - Review
    - Revise and enhance.

What needs to be learned

- Testing the product:
  - Key principles of testing for components:
    - Software
    - Hardware
    - Data
    - Interfaces
    - Resulting service
  - Documentation:
    - Test results
    - Updated project plan
    - Updated software help manual
    - Updated implementation map
    - Updated software code
    - Acceptance plan
  - Testing carried out:
    - Non-functional – availability testing, compatibility testing, configuration testing, load testing
    - Functional – unit testing, smoke testing, integration testing, system testing.
- Deploying/implementing the product:
  - Deployment locations:
    - On premises
    - Cloud
    - Hybrid
  - Deployment methods
    - Manual installation on each individual device
    - Scripted installation across multiple devices/systems
    - Replicating an image across multiple devices, virtual machines or cloud instances
    - Virtually
  - Documentation:
    - Acceptance testing
    - Client acceptance
  - Activities carried out:
    - Installation and configuration
    - Updates.

What needs to be learned

- Maintenance:
  - Documentation:
    - Software help manual
  - Activities carried out:
    - Software support
    - User support.

2B Similarities and differences between different software development methodologies

- Software development methodologies:
  - Waterfall
  - Agile.
- Factors to consider when comparing methodologies:
  - Software delivery time and speed
  - Document requirements
  - Structure and order of tasks (i.e. sequential or iterative)
  - Flexibility
  - Adaptability
  - Risk identification, management and mitigation
  - Size and complexity of the software suitable for
  - Ease of communication and collaboration between team members
  - Ease of client involvement and feedback opportunities.

2C Software development project roles and responsibilities

Students should develop an understanding of the different roles and responsibilities of different team members. They should understand how different individuals and teams work on different aspects of the project and how these are integrated to form a cohesive whole solution.

- Business analyst – responsible for translating business needs into requirements and ensures they are documented correctly.
- Product owner – represents the client or end users and has a clear vision of the end product.
- Software architect – defines the overall architecture systems and makes high-level design choices based on non-functional requirements.
- UX/UI designer – provides support to projects to analyse functional requirements intended for the end users.
- Software developer – in charge of writing the code and developing the software products.

What needs to be learned
• Software tester – responsible for ensuring that the software solution meets the requirements and complies with the quality standards. • Software project manager – responsible for delivering the software project on time and within budget. **2D Benefits of the software development life cycle** • Improved quality of software. • Greater likelihood of software meeting user needs. • Reduced costs of fixing problems after software delivery. • Each member of the project team focuses on one aspect of the software solution. **2E Methods of ensuring client data is held securely throughout the software development life cycle** • Holding client data under supervision. • Safeguarding client data against unauthorised access. • Not using personally identifiable information in test systems. • Adhering to Information Commissioner's Office (ICO) regulations.
Learning outcome 3: Investigate the project life cycle

3A Common stages involved during the project life cycle
• Software project planning principles:
  o Risk management
  o Dependencies
  o Integration
  o Prioritisation of tasks
  o Escalation of problems
  o Quality management.
• Conception and start-up:
  o Project mandate
  o Client requirements or project feasibility.
• Definition of the project:
  o Set up project team
  o Create the Project Initiation Document (PID).
• Planning:
  o Timescales
  o Costs
  o Risk management and controls.

What needs to be learned

- Launch and execution:
  - Carrying out the plan
  - Monitoring activity
  - Checking progress.
- Closure:
  - Handover of the product
  - User acceptance testing
  - Disbanding project team.
- Post-project evaluation:
  - Reviewing the project against success criteria.

3B Project life cycle roles and responsibilities

- Key stakeholder responsibilities:
  - Project manager – responsible for defining, planning, controlling and leadership
  - Technical teams – responsible for performing the project tasks
  - Team managers – responsible for following company policies and providing resources
  - Project sponsor – provides authority and guidance and maintains the priority of the project in the organisation
  - Client – provides the product requirements and project finance.
- Other stakeholders:
  - Suppliers – provide materials and equipment
  - Contractors – contribute specialist work
  - General public – may be affected by the project.
- Working within operational requirements:
  - Health and safety
  - Budgets
  - Brands
  - Normal business protocols.

3C Benefits of the project development life cycle

- Improves communication between team members.
- Project can easily continue if a project member leaves.
- Ensures a project is delivered on time.
- Resources can be allocated and planned when needed to ensure project is not delayed.
- Risk assessment reduces likelihood of project failing.

What needs to be learned

Learning outcome 4: Understand the principles of effective teamworking when producing computer programs

4A Software collaboration tools that can be used to enable developers to work together effectively

- Use of online code editors.
- Sharing workspaces.
- Single codebase.
- Branching code/modules between different developers.
- Merging code written by different developers.
- Version control.
- Track changes.
- Communication tools to allow developers to communicate (e.g. chat, discussion threads, video conferencing, document collaboration).

4B Importance of a collaborative mindset

- Reduced development time and testing.
- Developers are exposed to new techniques, tools and methods.
- Increased innovative approaches to problem solving.
- Diverse perspectives to increase code readability and reliability.
- Sharing of expertise, techniques and best practices between developers.

Students should show an awareness that collaboration goes beyond simply sharing ideas and problems. They should show an understanding of the importance of a collaborative mindset across different industry domains and sectors.

4C How effective project teamwork contributes to the delivery of software development projects

- Factors that affect team dynamics:
  - Attitudes:
    - Decision making
    - Interpersonal interactions
    - Performance, productivity
  - Loyalty to a specific person/group:
    - Recognising poor practice
    - Reporting concerns
  - Competition among colleagues:
    - Positive – innovation, motivation
    - Negative – creates tension, frustration

What needs to be learned

- Professional behaviour:
  - Respect for others
  - Commitment to quality
  - Responsibility
  - Accountability
  - Personal appearance
- Punctuality:
  - Managing own time
  - Working within time limits
- Cooperation:
  - Mutual benefit
- Conflict management:
  - Accommodating
  - Avoiding
  - Compromising.

- Collaborate, share good practice and solve problems together (e.g. decomposing software requirements together).
- Specialise in one area of expertise.
- Share ideas/solutions/designs/programming code from other projects completed to reduce development/testing time.

4D Communication methods and skills

- Communication methods, e.g.:
  - Discussion threads
  - Document collaboration
  - Instant messaging
  - Video conferencing.
- Communication skills, e.g.:
  - Non-verbal communication (body language, eye contact, hand gestures)
  - Active listening
  - Clarity
  - Concision
  - Confidence
  - Empathy.

What needs to be learned

- Adapting communication for different audiences:
  - Identify the purpose
  - Know the audience
  - Communicate through different channels
  - Use clear and concise language
  - Adjust style and tone
  - Seek and provide feedback.
- Importance of teamwork:
  - Achieving project objectives
  - Camaraderie
  - Motivates unity in the workplace
  - Offers different perspectives and feedback
  - Improves efficiency and productivity
  - Learning opportunities
  - Promotes workplace synergy.

Essential information for tutors and assessors

Essential resources

For this unit, students must have access to a range of programming languages, IDEs (integrated development environments) and diagramming tools to allow them to use a variety of tools and techniques (given in the unit content) to design and develop computer programs.

Assessment

This unit is internally assessed. To pass this unit, the evidence that students present for assessment must demonstrate that they have met the required standard specified in the learning outcomes and assessment criteria.

The assessment for this unit should be set in the context of the students showing how they have demonstrated and developed their skills drawing on learning from the unit. It must be designed in a way that enables students to meet all the assessment criteria.

The Authorised Assignment Brief (AAB) that includes this unit is a recommended assessment approach and sets out suitable sources of evidence for the learning outcomes. It also gives information about the standard and quality of evidence expected for students to achieve the learning outcome and pass each assignment. It is important that the information is used carefully alongside the assessment criteria.

Centres are free to amend the AAB or create their own assignment if they are confident it enables students to provide suitable and sufficient evidence to meet the stated standard of the assessment criteria and achieve the learning outcomes.

Unit 2: Programming Solutions

Level: 3

Unit type: Mandatory

Assessment type: Internal

Guided learning hours: 135

Unit introduction

This unit covers programming fundamentals and the knowledge and understanding needed to develop programmed solutions. This includes the knowledge, understanding and skills to break down problems, develop programmed solutions, identify and fix errors, and test those solutions.

Students will develop structured programmed solutions, including sequencing, selection and iteration. They will create decomposed solutions, incorporating separation of concerns, and develop solutions, including use of code blocking, library subroutines and user-defined subroutines. Students will develop an understanding of data structures and how to manipulate items in them, including arrays, lists, sets and hash maps. They will develop programmed solutions, using external data sources such as text files and relational databases.

Students will incorporate user interaction, ensuring that solutions are easy to use and accurate, and implement validation to ensure only valid data is processed.

Students will develop skills in finding and fixing errors in code, using the facilities of visual debuggers, and in testing programmed solutions, including functional (unit, integration) and non-functional testing (acceptance).

Students will communicate aspects of the solution and development of the solution to both technical and non-technical stakeholders.

No particular programming language paradigm or programming language is required. Students may use any paradigm of their choice as long as it can be used to demonstrate the unit content. Students are not required to produce a graphical user interface for any content or task in this unit.

Learning outcomes and assessment criteria

Learning outcomes	Assessment criteria
1. Be able to use fundamental concepts of computer programming	1.1 Implement techniques to control program structure and program flow 1.2 Apply techniques to decompose programs to create solutions to problems 1.3 Develop algorithms to show the rules that are followed to solve a problem
2. Be able to use data types, data structures and operations in solutions	2.1 Use data types in solutions 2.2 Apply mathematical, relational and logical/Boolean operators in solutions 2.3 Use data structures in solutions
3. Be able to use programming techniques to extend the functionality of programs	3.1 Use basic input and output, in relation to a user 3.2 Use functions to manipulate real numbers 3.3 Use string-handling functions to manipulate string data 3.4 Use external text files to import and export data
4. Be able to use validation techniques to improve the accuracy of data and reliability of programs	4.1 Apply validation techniques to ensure data can be processed 4.2 Demonstrate appropriate actions following data validation, such as skipping data or informing the user
5. Be able to implement meaningful user interfaces to programs	5.1 Develop a meaningful and accurate user interface for a given scenario 5.2 Handle user input, programmatically
6. Be able to use techniques to ensure programs are readable and maintainable	6.1 Use techniques to make programs readable, including program language-specific styles 6.2 Adopt source and version control to ensure all prior steps in development can be retrieved
7. Be able to apply techniques to ensure program solutions are robust and meet requirements	7.1 Use techniques to effectively manage source code 7.2 Use debugging techniques to identify and fix errors in programs 7.3 Select and use appropriate test data 7.4 Apply functional and non-functional testing techniques

Learning outcomes	Assessment criteria
8. Be able to interpret technical documentation and apply the basic principles of software design	8.1 Interpret customer-based requirements specifications 8.2 Interpret functional, non-functional and technical specifications 8.3 Implement separation of concerns
9. Be able to create the structure of a relational database and apply functions to access the data stored in it	9.1 Describe the difference between relational (SQL) and non-relational (NoSQL) databases 9.2 Design the structure of a relational database, including tables 9.3 Create primary keys to identify an entity in a relational database 9.4 Create relationships between tables by using foreign keys 9.5 Explain the concept of integrity 9.6 Design, implement and populate a relational database for a given scenario 9.7 Use an API to interact with a relational database 9.8 Discuss the communication, exchange and processing of data through different systems
10. Investigate issues related to the use of third-party code including artificial intelligence	10.1 Describe the issues related to the use of open-source code and APIs 10.2 Describe the issues related to the use of closed-source and proprietary APIs 10.3 Describe licensing issues related to third-party code 10.4 Explain how generative artificial intelligence can be used in software development and how this should be used responsibly
11. Be able to communicate to stakeholders	11.1 Communicate information about the functionality of a software solution to non-technical stakeholders 11.2 Communicate information about development of a software solution to technical stakeholders 11.3 Acknowledge sources of information

Unit content

What needs to be learned
Learning outcome 1: Be able to use fundamental concepts of computer programming

1A Program structure
• Program structure:
  o Statements
  o Blocks.
1B Flow control
• Sequence:
  o Most efficient and logical order of actions
  o Correct order of actions to ensure accurate outputs and avoid errors.
• Selection:
  o IF, ELSE and any combination, including nesting
  o SWITCH/CASE/MATCH.
• Iteration:
  o Count-controlled loops (FOR, FOR EACH)
  o Condition-controlled loops (WHILE REPEAT, DO UNTIL).
1C Spotting similarities within problems
• Pattern recognition:
  o Abstraction
  o Generalisation
  o Classification.
1D Program decomposition
• Subroutines:
  o Interfaces
  o Parameters
  o Arguments
  o Single entry, single exit
  o Procedures
  o Functions.
• Libraries:
  o Import libraries and specific subroutines from libraries
  o Calling library subroutines.

What needs to be learned

- User-defined subroutines:
  - Recognise when to use reusable components
  - Defining functions
  - Defining procedures
  - Calling user-defined subroutines.

1E Programming paradigms

Students should develop an awareness that programming code can be structured and organised in different ways and should learn a basic overview of different programming paradigms.

- Procedural (structures code into steps that are executed sequentially).
- Object-oriented (structures code around objects that have data and behaviours).
- Event driven (structures code by events that are triggered by user actions).

1F Algorithms

Students should understand fundamental principles of algorithms and develop algorithms to show the rules used to solve a problem, including:

- Logic
- Data structures
- Inputs and outputs
- Error handling
- Modularity.

Learning outcome 2: Be able to use data types, data structures and operations in solutions

2A Data types and mutability

- Data types:
  - Char, string, integer, real/float, Boolean.
- Variables:
  - Named place in memory
  - Declaration
  - Initialisation.
- Constants:
  - Concept of never changing
  - Declaration
  - Setting.

What needs to be learned

2B Operators

- Mathematical operators:
  - Addition, subtraction, division, multiplication, exponentiation, integer division, modulus.
- Relational operators:
  - Equivalence, less than, less than or equal to, greater than, greater than or equal to, not equal to.
- Logical/Boolean operators:
  - NOT, AND, OR, XOR.

2C Data structures

- Arrays:
  - Single data type
  - Fixed length.
- Lists:
  - Mixed data type
  - Dynamic length.
- Dimensionality:
  - One-dimensional
  - Two-dimensional.
- Operations (as allowed on structure by language):
  - Accessing an item (indexing)
  - Updating an item
  - Insert
  - Append
  - Remove.
- Collections:
  - Set:
    - Unordered
    - Without duplicate items
  - Hash map/dictionary:
    - Key value pairs
    - Ordered
    - Without duplicate keys.

What needs to be learned

Learning outcome 3: Be able to use programming techniques to extend the functionality of programs

3A Basic input and output

- Taking input from the keyboard.
- Outputting information to the display.

3B Numerical functions

- Rounding.
- Truncation.

3C String handling

- Concatenation.
- Splitting.
- Formatting, including tables.
- Length.
- Position of character/substring.
- Conversion:
  - Integer/float to string
  - String to integer/float.

3D Using data structures

- Ordering items.
- Locating item(s).
- Numerical and statistical operations (maximum, minimum, mean, mode, counting occurrences, total number of items).

3E Working with external text files

- Common delimiters (comma, whitespace).
- Reading data from a .txt file.
- Writing data to a .txt file.

Learning outcome 4: Be able to use validation techniques to improve the accuracy of data and reliability of programs

4A Validation checks

- Type check.
- Range check.
- Presence check.
- Format check.
- Length check.

What needs to be learned
4B Program actions following validation • Feedback to user. • Confirming validity and moving to next instruction. • Looping until validity is confirmed.
Learning outcome 5: Be able to implement meaningful user interfaces to programs

5A Provide meaningful and accurate user interaction
• Clarity and formatting of output messages/instructions.
• Formatting numeric data with consideration of:
  o Purpose
  o Accuracy
  o Ease of use.
5B Handle user input
• Simplification of user input (coding, menu options).
• Reformatting to allow processing (changing case, type casting).

Learning outcome 6: Be able to use techniques to ensure programs are readable and maintainable

6A Readability and maintainability
• Follow company, team and client approaches to:
  o Source and version control
  o Continuous code integration (i.e. to a shared repository)
  o Standard style (organisational, industry, language)
  o Layout guides (tabs, spacing, indents, line length)
  o Use of case (snake_case, camelCase, UPPER CASE, PascalCase)
  o Code annotations/comments/doc strings.

Learning outcome 7: Be able to apply techniques to ensure program solutions are robust and meet requirements
7A Source code management • Centralised code repositories. • Version control. • Check-out/check-in to allow developers to work on code locally. • Regular backups.

What needs to be learned

7B Debugging techniques

- Print statements.
- Visual debugger.
- Breakpoints.
- Step debugger.
- Memory inspection.

7C Selecting test data

- Typical/normal (valid data that should be accepted).
- Erroneous (invalid data that should be rejected).
- Boundary/extreme (either side of and on boundary).

7D Functional and non-functional testing

- Test-driven development.
- Unit testing (during development, components (a subroutine, a module)).
- Integration testing (during development, joining components).
- White-box testing (all paths through the code).
- Black-box testing (requirements driven).
- Stress testing/performance testing (load, volume).
- Alpha testing (final developmental).
- Beta testing (selected users).
- Acceptance testing (at the end, by the user, requirements driven).
- Test plan:
  - Test identifier
  - Description of the purpose of the test
  - Identification of test data to be used
  - Describing the expected outcomes
  - Logging the result of applying the test.
- Using the outcomes of testing to identify and fix issues.

7E Deployment of solutions

- Deployment of final product into a cloud-based environment.
- Testing of the product when being deployed.
- Stages of deployment.

What needs to be learned
Learning outcome 8: Be able to interpret technical documentation and apply the basic principles of software design

8A Requirements
• Customer-based.
• Use cases:
  o Ways a user interacts with a system, context.
• Storyboards.
• Performance.
• Accessibility.
8B Functional specifications
• What a solution is meant to do, input, output.
• User story (who, what, why).
8C Non-functional specifications
• How a system should perform not directly related to the functionalities:
  o System response times
  o System availability and accessibility
  o User interface responsiveness
  o Error handling.
8D Technical specifications
• Technical aspects required to develop, deploy and maintain the system:
  o Hardware requirements
  o Software requirements
  o Programming languages used.
8E Separation of concerns
• What it is and why it is important.
• Modularity (separate files).
• Subroutines.
• Libraries.

What needs to be learned

Learning outcome 9: Be able to create the structure of a relational database and apply functions to access the data stored in it

9A Databases

- Principles and uses of relational (SQL) databases.
- Principles and uses of non-relational (NoSQL) databases.

9B Relational database structures

- Schema.
- Entity.
- Attribute.
- Table.
- Record.
- Field.
- Relationship.

9C Keys

- Primary key.
- Composite key.
- Foreign key.

9D Relationships

- One-to-one.
- One-to-many.
- Many-to-many.

9E Integrity

- Entity integrity.
- Referential integrity.

9F Tools

- Data dictionary.
- Entity relationship diagram (ERD).

9G Using database functions

- Create the structure for a relational database.
- Updating, inserting, modifying and deletion.
- Retrieval of data for queries.
- Import and export.

What needs to be learned

9H Data interfaces and interactions

Students should understand the importance of having a holistic view of data. They should develop an awareness that the use of data goes beyond storing and extracting data in databases and understand the journey before, during and after being in a database. This includes communicating, exchanging and processing data through different:

- Hardware components
- Software components
- Interfaces
- APIs
- Protocols.

Learning outcome 10: Investigate issues related to the use of third-party code including artificial intelligence

10A Considerations for the use of third-party code

- What third-party code is.
- Open-source code and APIs.
- Closed-source code and APIs.
- Licensing issues.

10B Responsible use of generative artificial intelligence (AI)

Students should show an understanding of the different ways artificial intelligence can be used in software development, including:

- Code generation
- Code testing/correction
- Code optimisation
- Code debugging
- Code commenting/documentation
- Code analysis (e.g. analysing code efficiency, code security, etc.)
- Provide advice (e.g. offering explanations, suggesting improvements).

Students should show an awareness of how to use AI responsibly, including:

- Following organisational policy surrounding the use of AI (e.g. whether AI can be used, situations when AI can be used)
- Following licensing and terms of use of AI systems
- Checking the accuracy and reliability of the code produced
- Ensuring personal data handled by AI remains secure

What needs to be learned
• Awareness of the potential biases that AI systems might have inherited and how these may affect the behaviour of programs. Students should adopt a mindset of continual learning and adaptation when working with AI technologies. They should be aware of the reasons why it is important to stay up to date with new developments and with legal and ethical considerations.
Learning outcome 11: Be able to communicate to stakeholders
11A Communicate high-level information about the functionality of a software solution • Non-technical stakeholders. • Technical stakeholders. **11B Communicate information about the development of a software solution** • Technical stakeholder. • Technical requirements. • Structure. • Strengths. • Weaknesses. **11C Acknowledging sources, e.g. Harvard, ACM, IEEE** • In-text citations. • Reference list.

Essential information for tutors and assessors

Essential resources

For this unit, centres need:

- computer facilities
- access to the internet
- productivity software, e.g. word processor, spreadsheet, presentation, text editor, etc.
- a programming language, e.g. Java, Python, C++, C#
- a programming language translator for the selected language
- an integrated development environment (IDE), including an editor, with line numbers, syntax checker, step debugger, breakpoints and memory inspector
- a relational database with a programming interface (API), accessible from the programming language. This could be set up as a centre resource for all students to access, e.g. MySQL or similar. Alternatively, individual students could use SQLite or similar
- Access to AI generative models.

Assessment

This unit is internally assessed. To pass this unit, the evidence that students present for assessment must demonstrate that they have met the required standard specified in the learning outcomes and assessment criteria.

The assessment for this unit should be set in the context of the students showing how they have demonstrated and developed their skills drawing on learning from the unit. It must be designed in a way that enables students to meet all the assessment criteria.

The Authorised Assignment Brief (AAB) that includes this unit is a recommended assessment approach and sets out suitable sources of evidence for the learning outcomes. It also gives information about the standard and quality of evidence expected for students to achieve the learning outcome and pass each assignment. It is important that the information is used carefully alongside the assessment criteria.

Centres are free to amend the AAB or create their own assignment if they are confident it enables students to provide suitable and sufficient evidence to meet the stated standard of the assessment criteria and achieve the learning outcomes.

Task 2 is the coding task. Task 3 is the writing up of the coding task, showing the development journey. However, the two tasks must be done in parallel. The different code saved at completion of each user story is evidence of the coding task, but the development journey for that user story is written up in Task 3. The sections in Task 3 can be addressed whenever they come up in the development journey. For example, the different testing sections will be done at different stages of the development. Source control should be done at the completion of each functional user story. At any point, the state of the source code should be retrievable. In other words, the evidence for Task 2 will consist of many versions of the code, but each one will actually function.

Here is one way to approach Tasks 2 and 3:

Order	Action	Task
1	Retrieve a single backlog item. The order and choice are entirely up to you.	
2	Write a user story or stories to describe the backlog item. Remember, make them very small and focused.	3
3	Retrieve a copy of the most up-to-date source code version.	2
4	Break down the user story and design a way to implement it. You may wish to create an algorithm or use another design method of your choice. Consider how much, if any, existing code can be reused or has to be changed. This does not have to be written down, but you do need to think about it. It will be evidenced in your source code.	2
5	Thinking about your solution, are there any items in Task 3, e.g. the testing ones, that you can plan into this iteration of your code development? If so, then do the preparation for that, including any writing.	3
6	Now, develop the code to implement the user story. Remember, at any time, you may come across a situation that will allow you to address an item in Task 4, e.g. debugging using a visual debugger.	2, 3
7	Once the code for the user story is functional, then the source code should be stored as a new, most up-to-date version. Do not overwrite any existing versions.	2
8	In Task 3, where you wrote the user story, you need to tie the version of the source code to the user story, so the reader can find that version of the source code. This is a good place to also add a few words about how you know your code works. You can also provide screenshots, which is helpful and could reduce the number of words you need to write.	3
Back to 1	Repeat this process for all the backlog items. Be sure to leave time for acceptance testing and Task 4.	

Students should make use of a generative AI model to create short snippets of code that address specific areas of functionality, i.e. a specific function/process that the student would integrate into a larger code.

The AI model should not be used to create a single overarching solution to the entire brief.

The chosen AI model should allow students to demonstrate how they have entered specific prompts to create an output that they can then review in terms of how far it meets the specific needs of the identified functionality.

Pearson does not specify which generative AI the centres should use; however, the nature of the task would suggest the use of a natural language processing models.

Unit 3: Digital Technologies

Level: 3

Unit type: Mandatory

Assessment type: Internal

Guided learning hours: 35

Unit introduction

This unit covers digital technologies, including emerging technology trends and innovations. Students will research and examine the many different, new and evolving types of technology.

Students will examine the fundamental of computing systems, different types of hardware, how devices can be connected, how data can be secured and emerging technology trends.

 © Pearson Education Limited 2025

Learning outcomes and assessment criteria

Learning outcomes	Assessment criteria
1. Explore the fundamentals of digital technologies	1.1 Describe the purpose of common hardware components 1.2 Describe the purpose of common software components 1.3 Select suitable methods to connect digital devices 1.4 Design suitable human–computer interaction methods
2. Review the threats to digital systems and their impact on organisations	2.1 Categorise threats, vulnerabilities and risks to digital systems 2.2 Describe key methods of threat mitigation 2.3 Evaluate the impact that threats have on an organisation
3. Investigate the fundamentals of emerging digital technologies	3.1 Describe emerging technologies and how they are used in different situations

Unit content

What needs to be learned

Learning outcome 1: Explore the fundamentals of digital technologies

1A Purpose of common hardware components

- Input devices:
  - Keyboards
  - Mice
  - Touchscreens
  - Scanners
  - Near Field Technology/RFID (including card readers)
  - Sensors
  - Digital cameras/webcams (including their use in biometric systems)
  - Microphones (including their use in voice recognition and security applications).
- Output devices:
  - Screens/Touchscreens
  - Speakers
  - Printers
  - Motors/Actuators.
- Primary storage devices:
  - Random access memory
  - Read only memory
  - Flash memory.
- Secondary storage devices:
  - Magnetic hard disks
  - Solid state disks
  - Flash memory storage
  - Cloud storage.
- Internal hardware components:
  - CPU
  - Motherboards
  - Sound cards
  - Graphic cards
  - Ports.

What needs to be learned

- Factors that influence CPU performance:
  - Number of cores
  - Clock speed
  - Cache memory size.

1B Purpose of common software components

- Operating systems:
  - Real-time
  - Mobile
  - Batch
  - Multitasking
  - Network
  - Embedded
  - Distributed.

1C Select suitable methods to connect digital devices

- Network types:
  - Local area network (LAN)
  - Wide area network (WAN).
- Network topologies:
  - Bus
  - Ring
  - Star
  - Mesh.
- Network connection methods:
  - Wired
  - Wireless (Wi-Fi, Bluetooth, cellular (3g/4g/5g).

1D Design suitable human–computer interaction methods

- Design appropriate user interfaces. In all cases students should understand how the method works, typical input and output devices used, typical use, benefits and drawbacks of each method:
  - Command line interface
  - Menu-driven interface
  - Graphical user interface
  - Speech/Voice-driven interface
  - Motion-tracking interface.

What needs to be learned

Learning outcome 2: Review the threats to digital systems and their impact on organisations

2A Categorise threats, vulnerabilities and risks to digital systems

- Technical threats:
  - Malware (viruses, spyware, adware)
  - Denial-of-service (DoS) and distributed denial of service (DDoS)
  - Hacking/unauthorised access
  - Eavesdropping
  - ARP spoofing
  - Insecure API/3rd party libraries
  - Buffer overflow
  - Structured Query Language (SQL) injection.
- Physical threats (including natural disasters, theft, physical damage):
  - System location
  - System robustness.
- Human threats:
  - Human error (weak passwords, social engineering attacks, poor training)
  - Malicious employees.

2B Describe key methods of threat mitigation

- Anti-malware software.
- Air gapping.
- Minimising attack surface areas.
- Firewalls.
- Encryption (hashing, symmetric and asymmetric).
- Password security policies.
- Input sanitation.
- OS-provided runtime protection.
- Multifactor authentication (possession, knowledge and location based).
- Security testing (penetration testing).

2C Evaluate the impact that threats have on an organisation

- Loss of service.
- Financial loss.
- Reputation loss.
- Intellectual property loss.

What needs to be learned

3A Areas that emerging technologies will impact

- Internet of things.
- Wearable technologies.
- Smart homes.
- Autonomous vehicles.
- Smart cities.
- Physical computing:
  - What is physical computing?
  - Application of physical computing
  - Benefits and drawbacks of physical computing.

3B Applications, benefits and drawbacks of different emerging technologies

- Virtual technologies:
  - What is virtual computing?
  - Application of virtual computing
  - Benefits and drawbacks of virtual computing.
- Cloud technologies:
  - What is cloud computing?
  - Application of cloud computing
  - Benefits and drawbacks of cloud computing.
- Artificial intelligence (AI):
  - Narrow and general AI
  - Factors that have driven the development of AI
  - Machine learning (supervised and unsupervised)
  - Using AI to generate code
  - Evaluating AI generated-code
  - The impacts of AI on jobs and the skills required by programmers
  - Benefits and drawbacks of AI.
- Augmented reality (AR) and how it is used in different situations:
  - What is AR?
  - Hardware requirements
  - Processing requirements
  - Applications of AR
  - Benefits and drawbacks of AI.

Essential information for tutors and assessors

Essential resources

For this unit, students must have access to a range of programming languages, IDEs (integrated development environments) and diagramming tools to allow them to use a variety of tools and techniques (given in the unit content) to design and develop computer programs.

Assessment

This unit is internally assessed. To pass this unit, the evidence that students present for assessment must demonstrate that they have met the required standard specified in the learning outcomes and assessment criteria.

The assessment for this unit should be set in the context of the students showing how they have demonstrated and developed their skills drawing on learning from the unit. It must be designed in a way that enables students to meet all the assessment criteria.

The Authorised Assignment Brief (AAB) that includes this unit is a recommended assessment approach and sets out suitable sources of evidence for the learning outcomes. It also gives information about the standard and quality of evidence expected for students to achieve the learning outcome and pass each assignment. It is important that the information is used carefully alongside the assessment criteria.

Centres are free to amend the AAB or create their own assignment if they are confident it enables students to provide suitable and sufficient evidence to meet the stated standard of the assessment criteria and achieve the learning outcomes.

11 Appeals

Centres must have a policy for dealing with appeals from students. Appeals may relate to assessment decisions being incorrect or assessment not being conducted fairly. The first step in such a policy is a consideration of the evidence by a Lead Internal Verifier or other member of the programme team. The assessment plan should allow time for potential appeals after students have been given assessment decisions.

Centres must document all students' appeals and their resolutions. Further information on the appeals process can be found in the document *Internal assessment in vocational qualifications: Reviews and appeals policy*, available on our website.

12 Malpractice

Dealing with malpractice in assessment

Malpractice refers to acts that undermine the integrity and validity of assessment, the certification of qualifications and/or may damage the authority of those responsible for delivering the assessment and certification.

Pearson does not tolerate actual or attempted actions of malpractice by students, centre staff or centres in connection with Pearson qualifications. Pearson may impose penalties and/or sanctions on students, centre staff or centres where malpractice or attempted malpractice has been proven.

Malpractice may occur or be suspected in relation to any unit or type of assessment within a qualification. For further details on malpractice and advice on preventing malpractice by students, please see Pearson's *Centre Guidance: Dealing with Malpractice* available on our website.

Centres are required to take steps to prevent malpractice and to investigate instances of suspected malpractice. Students must be given information that explains what malpractice is for internal assessment and how suspected incidents will be dealt with by the centre. The *Centre Guidance: Dealing with Malpractice* document gives full information on the actions we expect you to take.

Pearson may conduct investigations if we believe a centre is failing to conduct internal assessment according to our policies. The above document gives further information and examples. It details the penalties and sanctions that may be imposed.

In the interests of students and centre staff, centres need to respond effectively and openly to all requests relating to an investigation into an incident of suspected malpractice.

Student malpractice

The head of centre is required to report incidents of suspected student malpractice that occur during Pearson qualifications. We ask centres to complete *JCQ Form M1* (www.jcq.org.uk/malpractice) and email it with any accompanying documents (signed statements from the student, invigilator, copies of evidence, etc) to the Investigations Processing team at candidatemalpractice@pearson.com. The responsibility for determining appropriate sanctions or penalties to be imposed on students lies with Pearson.

Students must be informed at the earliest opportunity of the specific allegation and the centre's malpractice policy, including the right of appeal. Students found guilty of malpractice may be disqualified from the qualification for which they have been entered with Pearson.

Failure to report malpractice constitutes staff or centre malpractice.

Teacher/centre malpractice

The head of centre is required to inform Pearson's Investigations team of any incident of suspected malpractice (which includes maladministration) by centre staff before any investigation is undertaken. The head of centre is requested to inform the Investigations team by submitting a *JCQ M2* Form (downloadable from www.jcq.org.uk/malpractice) with supporting documentation to pqsmalpractice@pearson.com. Where Pearson receives allegations of malpractice from other sources (for example Pearson staff, anonymous informants), the Investigations team will conduct the investigation directly or may ask the head of centre to assist.

Pearson reserves the right in cases of suspected malpractice to withhold the issuing of results/certificates while an investigation is in progress. Depending on the outcome of the investigation, results and/or certificates may not be released, or they may be withheld.

You should be aware that Pearson may need to suspend certification when undertaking investigations, audits and quality assurances processes. You will be notified within a reasonable period of time if this occurs.

Sanctions and appeals

Where malpractice is proven, we may impose sanctions or penalties, such as:

- mark reduction for affected external assessments.
- disqualification from the qualification
- debarment from registration for Pearson qualifications for a period of time.

If we are concerned about your centre's quality procedures, we may impose sanctions such as:

- working with centres to create an improvement action plan.
- requiring staff members to receive further training.
- placing temporary suspensions on certification of students
- placing temporary suspensions on registration of students
- debarring staff members or the centre from delivering Pearson qualifications
- suspending or withdrawing centre approval status.

The centre will be notified if any of these apply.

Pearson has established procedures for considering appeals against penalties and sanctions arising from malpractice. Appeals against a decision made by Pearson will normally be accepted only from the head of centre (on behalf of students and/or members or staff) and from individual members (in respect of a decision taken against them personally). Further information on appeals can be found in the *JCQ Appeals booklet* (www.jcq.org.uk/exams-office/appeals).

13 Further information and publications

- Edexcel, BTEC and Pearson Work Based Learning contact details: https://qualifications.pearson.com/en/contact-us.html.
- Books, software and online resources for UK schools and colleges: www.pearsonschoolsandfecolleges.co.uk.
- Our publications catalogue lists all the material available to support our qualifications. To access the catalogue and order publications, please visit our website.

Further documents that support the information in this specification:

- *Access arrangements and reasonable adjustments* (JCQ)
- *A guide to the special consideration process* (JCQ)
- *Collaborative and consortium arrangements for the delivery of vocational qualifications policy* (Pearson)
- *UK information manual* (updated annually and available in hard copy) **or** *Entries and information manual* (available online) (Pearson).
- *Distance learning and assessment policy* (Pearson)

14 Glossary

General terminology used in specification

Term	Description
Level	Units and qualifications have a level assigned to them. The level assigned is informed by the level descriptors defined by Ofqual, the qualifications regulator.
Guided learning hours (GLH)	This indicates the number of hours of activities that directly or immediately involve tutors and assessors in teaching, supervising, and invigilating students, for example lectures, tutorials, online instruction and supervised study. Units may vary in size.
Total qualification time (TQT)	This indicates the total number of hours that a typical student will take to complete the qualification. This is in terms of both guided learning hours but also unguided learning, for example private study, time spent in the workplace to master skills.
Learning outcomes	The learning outcomes of a unit set out what a student knows, understands or is able to do as the result of a process of learning.
Assessment criteria	The assessment criteria specify the standard the student is required to meet to achieve a learning outcome.
Unit content	This section sets out the required teaching content of the unit and specifies the knowledge, skills and understanding required for achievement of the unit. It enables centres to design and deliver a programme of learning that will enable students to achieve each learning outcome and to meet the standard determined by the assessment criteria.
Summative assessment	Assessment that takes place after the programme of learning has taken place.
Valid assessment	The assessment assesses the skills or knowledge/understanding in the most sensible, direct way to measure what it is intended to measure.
Reliable assessment	The assessment is consistent, and the agreed approach delivers the correct results on different days for the same students and different cohorts of students.

SEB 050825 VQ000382 L3 TOEQ SOFTWARE TECH ISS1 050825.DOCX.1-60/0

For information about Pearson Qualifications, including Pearson Edexcel, and BTEC visit qualifications.pearson.com

Edexcel and BTEC are registered trademarks of Pearson Education Limited

Pearson Education Limited. Registered in England and Wales No. 872828
Registered Office: 80 Strand, London WC2R 0RL.

VAT Reg No GB 278 537121

Cover image © Gorodenkoff / Shutterstock

Pearson
BTEC

Publication code:
VQ000383