T-LEVELS | IfATE Institute for Apprenticeships and Technical Education

T LEVEL

Technical Qualification in Digital Software Development

Specification

First teaching from September 2025

Version 1.0 – May 2025

Pearson

T-LEVELS | IfATE Institute for Apprenticeships and Technical Education

T Level Technical Qualification in Digital Software Development (Level 3)

Specification

First teaching September 2025

Version 1.0 May 2025

Pearson

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 learners, educational institutions, employers, governments and other partners globally. We are committed to helping equip learners 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.

This specification is Version 1.0

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 VQ000475

Contents

1 Introducing the qualification

T Level programme

T Levels are two-year, Level 3 study programmes that follow the study of GCSEs and Technical Awards and offer an alternative to A Levels and Apprenticeships.

T Levels combine classroom theory, practical learning and a minimum 315 hours of industry placement with an employer. The work placement ensures students have real experience of the workplace.

T Level programmes are developed in collaboration with employers so that the content meets the needs of industry and prepares students for work. T Levels provide the knowledge and experience needed to progress to highly skilled employment, an Apprenticeship or higher-level study, including university.

Understanding the Specification and Administrative Guide

This specification should be read in conjunction with the Administrative Guide for Delivery and Assessment. The specification contains all the information you need to teach the Technical Qualification, including content and assessment details. The Admin Guide contains the information and references you need to register as a Provider, register students and administer their results. It also contains grading information and information on resources.

What is the Technical Qualification (TQ)?

The *T Level Technical Qualification in Digital Software Development* is the main classroom-based element of the T Level. Students will learn using a curriculum that has been shaped by industry experts.

During the two-year programme, students will acquire the core knowledge that underpins each industry. They will develop occupationally specific skills that will allow them to enter skilled employment within a specific occupation.

Technical Qualification and Outline Content

The Outline Content for the *T Level Technical Qualification in Digital Software Development* has been produced by T Level panels of employers, professional bodies and Providers. It is based on the Apprenticeship Standards.

Pearson has used the Outline Content to form the basis of the Technical Qualification specification. This includes:

- elaboration of the Outline Content to produce a specification that gives Providers an accurate interpretation of what needs to be taught and assessed
- enabling students to achieve threshold competence in relation to the Occupational Specialism component
- the integration of English, maths and digital competencies.

English, maths and digital competencies

English, maths and digital competencies are signposted against Occupational Specialism content. This is because these competencies are best enhanced when students are developing the knowledge and skills they need for threshold competence. The signposting suggests the content areas where the competencies can best be developed by students in the course of their learning. The competencies are indicated by abbreviations (e.g. E1) and the explanation of the abbreviations is contained in Appendix 1.

Employer and Provider panels

Pearson engaged with employer and Provider panels throughout the development of the Technical Qualification. This ensured:

- the content gives students quality preparation to help them progress
- assessments are realistic and assess the knowledge and skills that are important to employers
- the Technical Qualification meets the needs of Providers.

Pearson is grateful to all university and further education lecturers, teachers, employers, professional body representatives and other individuals who have generously shared their time and expertise to help us develop these new qualifications.

Qualification purpose

This Technical Qualification is for T Level students who are undertaking the *T Level in Digital Software Development*. It is intended for students who want to progress to a career in the *digital* sector.

The purpose of the *T Level Technical Qualification in Digital Software Development* (Level 3) is to ensure students have the knowledge and skills needed to progress into highly skilled employment, an Apprenticeship or higher-level study, including university, within the specialist area of digital software development.

At the end of the Technical Qualification, students are expected to demonstrate threshold competence, meaning that they have gained the core knowledge and skills related to digital software development and are well placed to develop full occupational competence with additional development and support once in employment in the digital sector.

Student profile and progression

Students undertaking this Technical Qualification will be 16–19 years old and in full-time education.

The typical student has:

- a clear idea about the industry sector in which they wish to pursue a career
- an idea of the type of job role they would like to explore as a career.

This Technical Qualification aligns to the Level 3 Apprenticeship for Software Development Technicians. The qualification therefore supports progression to entry-level job opportunities in software development.

Job roles could include:

- Software Development Technician
- Junior Developer
- Junior Web Developer
- Junior Application Developer
- Junior Mobile App Developer
- Junior Games Developer
- Junior Software Developer
- Junior Application Support Analyst
- Junior Programmer
- Assistant Programmer
- Automated Test Developer.

Alternatively, students could progress to Level 3 Apprenticeships such as those mentioned above to develop and gain certification of full occupational competence, or they could progress to higher-level Apprenticeships such as the Level 4 Software Developer depending on their skills and/or experience.

Where students may not have access to an Apprenticeship or would prefer a more academic route, they could progress to relevant Higher National Certificate (HNC) or Higher National Diploma (HND) programmes or degree programmes.

Students must check the entry requirements for each degree programme with the relevant higher education provider.

2 Qualification summary and structure

Summary

Qualification title	T Level Technical Qualification in Digital Software Development (Level 3)
Qualification number (QN)	610/5801/4
First teaching	September 2025
This qualification replaces	603/5832/4 T Level Technical Qualification in Digital Production, Design and Development
Total Guided Learning Hours (GLH)	1260 hours (630 hours core)
Total Qualification Time (TQT)	1740 hours (840 hours core)
Occupational Specialism(s)	Digital Software Development (630 GLH, 900 TQT)
Components and weighting	Core Paper 1 = 30% of core (15% of total) Core Paper 2 = 30% of core (15% of total) Core ESP = 40% of core (20% of total) Core Component = 50% of total Occupational Specialism = 50% of total
Recommended age range	16–19
Grading information	Core and Employer Set Project (ESP) components are graded A*–E or Unclassified. The Occupational Specialism (OS) component is graded Pass, Merit, Distinction or Unclassified. The overall grading is on a scale of Pass, Merit, Distinction, Distinction* or Unclassified. The overall grade is awarded by the Institute for Apprenticeships and Technical Education (IfATE).

Qualification title	T Level Technical Qualification in Digital Software Development (Level 3)
Entry requirements	There are no formal prior learning requirements. It is the Provider's responsibility to ensure students recruited have a reasonable expectation of success. Students are most likely to succeed if they have qualifications at Level 2 (for example, 5 GCSEs at grade 4 and above including English and maths or a vocational Tech Award pass at Level 2). Students may demonstrate the ability to succeed in various ways. For example, they may have relevant work experience or may have shown specific aptitude through diagnostic tests or other non-educational experience.
Assessment	• The core and ESP components are externally set and marked by Pearson. • The OS components are set by Pearson. These are externally marked by Pearson.

Assessment Structure

The *T Level Technical Qualification in Digital Software Development* has two mandatory components.

1. Core component

This component covers the underpinning knowledge, concepts and skills that support threshold competence in the digital industry.

The content for the Core component is provided in *Section 3*.

Assessment component	Assessment method	Duration	Marks	Weighting	Timetable	Availability
Core Paper 1	Written examination	2 hours 15 minutes	90	30%	Set date/time	June/ November
Core Paper 2	Written examination	2 hours 15 minutes	90	30%	Set date/time	June/ November
Employer Set Project	Externally set project	14 hours 30 minutes	100	40%	Set date/time	May/ November

2. Occupational Specialism component

There is one Occupational Specialism component in this Technical Qualification.

These components cover the Occupational Specialism knowledge and skills required to demonstrate threshold competence for the specialism. The Occupational Specialism is assessed by a skills-related project that synoptically assesses the Performance Outcome skills and associated underpinning knowledge.

The content for the Occupational Specialism component is provided in *Section 4*.

Assessment component	Assessment method	Duration	Marks	Weighting	Timetable	Availability
Digital software development	Externally set project	50 hours 30 minutes	144	100%	Windowed	March to May

What does the qualification cover?

The Technical Qualification content has been designed from the Outline Content created by the Institute for Apprenticeships and Technical Education and the Digital T Level panel.

We have used the Outline Content to create the Technical Qualification specification and assessment, which have been validated by our own panel of employers and Providers to ensure they are appropriate for the progression routes identified.

Students learn about the following topics:

- Problem solving
- Introduction to programming
- Emerging issues
- Legislation and regulatory requirements
- Business context
- Data
- Digital environments
- Security.

3 Core Component

The content of the Core component has the core skills mapped to where there are opportunities to develop them. The competencies and skills are not expected to be developed at every point where they are mapped, but using this guidance teachers will embed them into teaching to prepare students for the assessments in the Core component.

The core skills are assessed through the Employer Set Project. The core skills for this Core component are as follows:

1. Be able to reflectively evaluate
2. Communicate information clearly to a technical and non-technical audience
3. Work with others in a collaborative manner to allow for/encourage faster, better and more efficient achievement of goals
4. Develop software/Create an artefact
5. Apply a logical approach to solving problems to:
    - identify and fix defects
    - propose software solutions
6. Ensure software development activity mitigates risks to security.

Content

Core paper 1

Content area 1: Problem solving	
Students will solve digital software development problems that form a complete solution or a sub-part of a solution. Students will use problem-solving skills to analyse problems and to identify solutions that can be represented as systems, processes, relationships, organisations of data or code. Where the term 'code' is used, this refers to Python 3.10 or later.	
1.1 Computational thinking	
1.1.1	Know the definition and understand the purpose of computational thinking.
1.1.2	Know when to use computational thinking.
1.1.3	Know and understand the benefits and drawbacks of using computational thinking.
1.1.4	Know the components of computational thinking: • decomposition • pattern recognition • abstraction • algorithmic design.
1.1.5	Know and understand the benefits and drawbacks of using the components of computational thinking.
1.1.6	Know and understand the purpose of decomposition.
1.1.7	Know the tasks of decomposition: • identify the main features of a problem • characterise each identified feature • break problems down into smaller, more manageable parts • break solutions down into smaller, more manageable parts.
1.1.8	Be able to use decomposition for problem solving.
1.1.9	Know and understand methods to represent decomposition: • block diagrams • information flow diagrams • flowcharts • code • written descriptions.
1.1.10	Be able to use the methods to represent decomposition.

1.1.11	Know and understand the purpose of pattern recognition.
1.1.12	Be able to use pattern recognition for problem solving: • find and interpret trends and similarities within and between problems and processes • find and interpret common features between a given problem and existing solutions • make predictions and assumptions based on identified patterns.
1.1.13	Know and understand the purpose of abstraction.
1.1.14	Know and understand the tasks of abstraction: • identify information that is needed • filter out unnecessary details • hide details of internal workings.
1.1.15	Be able to use abstraction: • what inputs are needed • what the expected outputs and outcomes are • things that will vary • things that will remain constant • key actions the solution must perform • repeated processes the solution will perform.
1.1.16	Be able to use abstraction in problem solving.
1.1.17	Understand the interrelationships between components of computational thinking and make judgements about the suitability of using the components in digital software development.
1.2 Algorithmic design	
1.2.1	Know the definition and understand the characteristics and purpose of algorithms.

1.2.2	Know and understand that algorithms can be expressed in:
	• flowcharts:
	  ○ terminators
	  ○ processes
	  ○ sub-processes
	  ○ decisions
	  ○ inputs/outputs
	  ○ arrows
	  ○ labels
	• written descriptions using hierarchical markers to indicate sequence
	• code (commands in *Appendix 2*).

1.2.3	Know and understand the benefits and drawbacks of the ways of expressing algorithms in flowcharts.
1.2.4	Know and understand the benefits and drawbacks of the ways of expressing algorithms in written descriptions.

1.2.5	Know and understand the benefits and drawbacks of the ways of expressing algorithms in code.
1.2.6	Know and understand actions to control ordering of steps in algorithms: • sequence • selection • iteration.
1.2.7	Be able to determine the purpose of an algorithm and how it works.
1.2.8	Be able to determine the output of an algorithm given an input.
1.2.9	Be able to identify errors in an algorithm.
1.2.10	Be able to correct errors in an algorithm.
1.2.11	Be able to translate between the different notations for algorithms.
1.2.12	Be able to design algorithms and solutions that use actions.
1.3 Strategies	
1.3.1	Know the different approaches to solving problems and understand their purpose and when they are used: • top-down • bottom-up • modularisation.
1.3.2	Know the benefits and drawbacks of using the different approaches to solving problems.
1.3.3	Understand the purpose of root cause analysis and when it is used.

1.3.4	Know and understand approaches to root cause analysis:
	• five whys
	• failure mode and effects analysis (FMEA)
	• event tree analysis (ETA)
	• actions to take after using root cause analysis:
	  ○ log
	  ○ close
	  ○ escalate to an appropriate manager, specialist or external third party.

1.3.5	Know and understand the process of the high-level problem-solving strategy: • define the problem • gather information • analyse the information • make a plan of action • implement a solution • review the solution.

1.3.6	Understand the interrelationships between problems and problem-solving strategies and make judgements about the suitability of strategies for solving the problems in digital software development.

Content area 2: Introduction to programming	
Students will analyse digital software development problems that may involve software, people, processes and data. Students will use a variety of tools and techniques when developing a complete solution or a sub-part of a solution. Where the term 'code' is used, this refers to Python 3.10 or later.	
2.1 Standard data types	
2.1.1	Know the definition of common data types and understand their purpose and when each is used: • integer • float • string • Boolean.
2.2 Variables and constants	
2.2.1	Know the definition of variables, understand their purpose and when they are used.
2.2.2	Know the definition of constants, understand their purpose and when they are used.
2.2.3	Understand the purpose of data type conversion functions and why they are used. See *Appendix 2* for functions.
2.2.4	Know a definition of scope and understand the role of scope and when it is used.
2.2.5	Know and understand how variables are managed by scope: • global variables • local variables.
2.2.6	Be able to use scope.
2.2.7	Be able to declare variables and constants using standard data types.
2.2.8	Be able to use variables and constants.
2.2.9	Be able to use data type conversion functions.
2.3 Data structures	
2.3.1	Know the standard data structures and understand their purpose and when each is used: • list • array • dictionary.
2.3.2	Be able to interpret code using data structures.

2.3.3	Be able to develop code using data structures.
2.3.4	Be able to debug code using data structures.
2.4 Operators	
2.4.1	Understand the purpose of arithmetic operators: • add • subtract • divide • multiply • exponentiation • integer division • modulus. See *Appendix 2* for operators.
2.4.2	Understand the purpose of relational operators: • equivalence • less than • greater than • not equal • less than or equal to • greater than or equal to. See *Appendix 2* for operators.
2.4.3	Understand the purpose of Boolean operators: • not • and • or. See *Appendix 2* for operators.
2.4.4	Be able to use operators: • arithmetic • relational • Boolean.
2.4.5	Be able to interpret code using operators.
2.4.6	Be able to create code using operators.
2.4.7	Be able to debug code using operators.
2.5 Input and output	
2.5.1	Understand how to implement input and output: • keyboard • screen • text file.

2.5.2	Know and understand the use of text files for input and output: • open a file for reading • open a file for writing • write lines to the file • close the file.
2.5.3	Be able to interpret code using input and output.
2.5.4	Be able to create code using input and output.
2.5.5	Be able to debug code using input and output.
2.6 Actions	
2.6.1	Know and understand sequence, its purpose and when it is used.
2.6.2	Know and understand selection, its purpose and when it is used: • if • else if • else • match/case. See *Appendix 2* for key words.
2.6.3	Know and understand loops, their purpose and when they are used: • count-controlled loop • condition-controlled loop.
2.6.4	Know and understand iteration: • count-controlled ‘for’ loops • condition-controlled ‘while do’ loops.
2.6.5	Know and understand the benefits and drawbacks of loops.
2.6.6	Be able to interpret code using actions: • sequence • selection • iteration.
2.6.7	Be able to develop code using actions: • sequence • selection • iteration.
2.6.8	Be able to debug code using actions: • sequence • selection • iteration.

2.7 Functions and procedures	
2.7.1	Know characteristics of functions, understand their purpose and when they are used: • may or may not take parameters • must return a result.
2.7.2	Know characteristics of procedures, understand their purpose and when they are used: • may or may not take parameters • must not return a result.
2.7.3	Know and understand where functions and procedures come from: • user-written • pre-written and built-in as part of the programming language • pre-written and supplied in libraries as part of the programming language • pre-written and supplied in libraries from third parties.
2.7.4	Know and understand the benefits and drawbacks of using pre-written code.
2.7.5	Be able to interpret code using user-written and pre-written code: • functions • procedures.
2.7.6	Be able to develop code using user-written and pre-written code: • functions • procedures.
2.7.7	Be able to debug code using user-written and pre-written code: • functions • procedures.
2.8 Validation	
2.8.1	Know the definition of validation and understand the purpose and when validation checks are used: • presence check • length check • range check • type check • format check • check digit.
2.8.2	Be able to interpret code using validation.
2.8.3	Be able to develop code using validation.
2.8.4	Be able to debug code using validation.

2.9 Design considerations and programming practices	
2.9.1	Determine the logical order for actions within processes.
2.9.2	Determine the order of operations in calculations and processes: • to ensure outputs are accurate • to ensure errors are avoided.
2.9.3	Determine the selection of data structures: • to ensure efficiency of execution time • to ensure efficient use of memory.
2.9.4	Determine the order of actions: • to ensure efficiency of execution time • to ensure efficient use of memory.
2.9.5	Know and understand naming conventions and their purpose: • meaningful names • camelCase • snake_case.
2.9.6	Understand the impact on readability of code style conventions: • naming conventions • use of white space • maximum line length.
2.9.7	Make judgements about the suitability of an algorithm in meeting requirements, efficiency in use of storage and execution time, appropriateness in the choice of data structures, data types, variables, constants, presentation and maintainability.
2.10 Robust code	
2.10.1	Know and understand the characteristics of robust code: • handles unexpected inputs • handles unexpected terminations • produces specific and meaningful error messages.
2.10.2	Know and understand the process of debugging and when it is used: • locating errors in code • correcting errors in code.
2.10.3	Understand the role of debugging in producing solutions that are robust.
2.10.4	Be able to locate errors in code.
2.10.5	Be able to correct errors in code.
2.11 Common algorithms	
2.11.1	Know and understand the algorithms for searching, how they work and when they are used: • linear search • binary search.

2.11.2	Know and understand the algorithms for sorting, how they work and when they are used: • bubble sort • insertion sort • merge sort.
2.11.3	Know and understand the benefits and drawbacks of using: • linear search • binary search • bubble sort • insertion sort • merge sort.
2.11.4	Know and understand metrics to compare algorithms: • use of memory space • execution time • number of comparisons.
2.11.5	Know and understand best case, worst case and average case for common algorithms, using logical reasoning (Big O not required).
2.11.6	Make judgements about the suitability of using different algorithms for searching.
2.11.7	Make judgements about the suitability of using different algorithms for sorting.
2.12 Testing	
2.12.1	**Testing components**
2.12.1.1	Understand the reasons for testing individual components of a solution before putting them together in the final solution: • software • hardware • data • interfaces • resulting service (final product).
2.12.2	**Testing methods**
2.12.2.1	Know a definition of testing methods and understand their purpose, benefits and drawbacks, and when they are used: • concept • unit • boundary • integration • performance • system • acceptance • usability

	• regression • load/stress • closed box • open box.
2.12.2.2	Be able to use testing methods.
2.12.3	**Automation**
2.12.3.1	Know and understand the purpose of automation methods and when they are used: • macros • scripts • functional testing tools.
2.12.4	**Test data and test plan**
2.12.4.1	Know a definition for types of test data and understand the purpose of test data and when it is used: • valid • invalid • boundary • erroneous.
2.12.4.2	Be able to create test data.
2.12.4.3	Know and understand the steps and structure of a test plan and when it is used: • identifying tests to be carried out • describing the purpose of the identified test • identifying test data to be used • describing the expected results • recording actual results.

Content area 3: Emerging issues	
3.1 Impact of digital technologies	

3.1.1	Understand how the increased reliance on digital systems impacts:
	• organisational culture:
	  ○ changes in communication methods (face to face, email, video calls)
	  ○ increased productivity and availability expectations
	  ○ increase in staff monitoring
	  ○ new working practices (remote/hybrid/in-office working)
	  ○ automation of services including the use of artificial intelligence (AI)
	• society:
	  ○ loss of jobs
	  ○ shift in skill requirements
	  ○ reduction in human decision making and loss of empathy

	○ privacy (digital footprint, surveillance) ○ changing behaviours (loss of social skills, digital identity) ○ access to wider social networks (personal and professional) ○ access to online services (government, commercial and entertainment) ○ potential isolation (lack of skill, equipment, connectivity, resistance to change) ○ improved access to information (professional and personal) ○ increased use of AI including generative AI (textual, graphical, video and audio) ○ globalisation: – access to global media sources.
3.1.2	Understand the importance of digital inclusion: • ensuring fair access to digital services: ○ suitable technologies (hardware and software) ○ connectivity ○ checking for bias within datasets ○ conforming to codes of best practice ○ public sector bodies' website and mobile applications accessibility regulations: – key features and purpose.
3.1.3	Understand how end user characteristics affect the use of and inclusivity of digital systems: • age • skills: ○ digital ○ literacy • internal/external audience • cultural issues, including bias in digital systems • additional needs: ○ accessibility issues.
3.1.4	Know and understand the benefits of professional development: • increased industry and sector competence • increased employability potential and employment security • achieving access to knowledge of and adherence to industry standards.
3.2 Emerging technologies	
3.2.1	Understand how developments in technologies impact organisations, individuals and society: • storage media: ○ increased demand for storage • processing technologies: ○ quantum computing

	• Internet of Things (IoT):
	  ○ edge computing
	  ○ use within different contexts (industrial, smart city, domestic)
	• artificial intelligence:
	  ○ generative AI
	  ○ machine learning
	• extended reality:
	  ○ augmented reality
	  ○ virtual reality
	• open source software
	• blockchain
	• environmental:
	  ○ consumption of rare metals
	  ○ energy to produce electronic systems
	  ○ environmental impact of disposal
	• autonomous machines:
	  ○ self-driving cars
	  ○ robotic assembly lines.

3.2.2	Understand the interrelationships between digital and emerging technologies and make judgements about their impacts on organisations, society and individuals in digital software development.

Content area 4: Legislation and regulatory requirements	
4.1 Legislation	

4.1.1	Understand the key points and implications to employers of the relevant health and safety legislation:
	• Health and Safety at Work Act
	  ○ key points:
	  – provide a safe working environment
	  – ensure staff are properly trained
	  – adequate welfare provision
	  – provide relevant information, instruction and supervision
	• display screen equipment.
	  ○ implications for employers:
	  – conduct a display screen equipment workstation assessment
	  – reduce risks including making sure workers take breaks from display screen equipment work
	  – provide an eye test if an employee asks for one
	  – provide training and information for employees.

4.1.2	Understand the health and safety risks and preventative measures of working with digital systems:
	• possible risks:
	  ○ using display screen equipment
	  ○ health and safety requirements
	• methods of mitigating risk
	  ○ adequate training
	  ○ safe working environment
	  ○ safe working practices.

4.1.3	Understand data security and protection legislation including the effect on organisations and individuals:
	• Data Protection Act/General Data Protection Regulation:
	  ○ purpose of legislation
	  ○ eight principles.

4.1.4	Understand computer misuse legislation: • the principles of the Computer Misuse Act (CMA) 1990 • consequences for company and employee • employee awareness • types of crime covered by legislation.

4.1.5	Understand equality legislation:
	• the nine protected characteristics
	• types of discrimination:
	  ○ direct
	  ○ indirect
	  ○ harassment
	  ○ victimisation
	• where individuals are protected
	• when to take action against discrimination:
	  ○ time limits for claims.

4.1.6	Understand intellectual property legislation: • unregistered designs • registered designs • patents.
4.1.7	Understand the interrelationships between digital software development and digital legislation, and make judgements about the impact on organisations, society and individuals.
4.1.8	Know that international law applies to some offences: • international law in cyberspace • international law and surveillance.

4.2 Guidelines	

4.2.1	Know the sources of codes of conduct:
	• organisational
	• professional:
	  ○ British Computer Society (BCS)
	  ○ The Institution of Analysts and Programmers (IAP)
	  ○ Chartered Institute of Information Security (CIISec)
	• governmental.

4.2.2	Understand how guidelines in codes of conduct influence professional behaviour:
	• ensuring individuals follow policies, procedures and legislation
	• ensuring quality of work:
	  ○ minimising risk to the public
	  ○ acting with competence and integrity
	• meeting deadlines
	• effective communication
	• maintaining confidentiality and trust.

4.2.3	Know the sources of digital industry standards: • International Organization for Standardization (ISO) • Web Content Accessibility Guidelines (WCAG) • World Wide Web Consortium (W3C®) • Internet Engineering Task Force (IETF) • British Standard (BS) • Institute of Electrical and Electronics Engineers (IEEE) • Payment Card Industry Security Standards Council (PCI SSC).

4.2.4	Understand the purpose of acceptable use policies (AUP):
	• purpose of AUP
	• typical content:
	  ○ permitted activities
	  ○ prohibited activities
	  ○ working practices including confidentiality
	  ○ communication etiquette including projecting correct organisation image
	  ○ sanctions/penalties.

4.2.5	Understand the importance of whistleblowing procedures.
4.2.6	Understand the interrelationships between digital software development and guidelines, and make judgements about the impact on organisations, society and individuals.

Core paper 2

Content area 5: Business context	
5.1 Business environment	

5.1.1	Know the purpose and sectors of different types of organisations:
	• purpose of the organisation:
	  ○ providing a service
	  ○ providing a product
	• private sector:
	  ○ small or medium-sized enterprise (SME)
	  ○ large enterprise
	  ○ non-governmental organisation (NGO)
	• public sector
	• voluntary/charity:
	  ○ not for profit.

5.1.2	Know the names and definitions of different business models: • Business to Customer (B2C) • Business to Business (B2B) • Business to Many (B2M).

5.1.3	Know the different types of stakeholders:
	• internal stakeholders:
	  ○ owners
	  ○ directors
	  ○ employees
	• external stakeholders:
	  ○ customers/clients
	  ○ suppliers
	  ○ shareholders
	  ○ outsourced services
	  ○ investors/funders
	  ○ government.

5.2 Digital value to organisations	

5.2.1	Understand how digital systems are used to support key organisation areas:
	• sales and Marketing:
	  o better market research
	  o better brand promotion, including social media
	  o online selling
	  o contextualising customer behaviour to personalise services offered
	  o better customer retention
	  o brand differentiation and values
	  o use of analytic tools, including search and social media analytics
	• research, design and development:
	  o provision of unique products and services
	• Human Resources:
	  o staff records
	  o performance management
	  o training records
	• operations:
	  o enhanced internal communication
	  o automation of internal processes
	  o automated manufacturing
	  o remote working
	  o intranet/shared workspace
	  o document sharing and online shared storage
	• management:
	  o real-time monitoring of key performance indicators:
	  – sales
	  – customers served
	  – units manufactured
	  – real-time location of assets
	• logistics:
	  o automated stock control
	• finance:
	  o reduced costs
	  o increased revenue
	  o better financial reporting via up-to-date information.

5.2.2	Understand how digital systems are used to meet user needs and ensure quality of product/service:
	• appropriate and effective functionality:
	  ○ allows users to do all required tasks
	• reduction of pain points:
	  ○ response time (communication of expected response time, notification of change in response time)
	  ○ complexity of task
	• appropriate accessibility provision
	• compatibility:
	  ○ with internal legacy systems
	  ○ with proposed future systems
	  ○ with external services
	• availability of service:
	  ○ minimise downtime
	  ○ future proofing for upgrades
	• effective end user support:
	  ○ provision of digital support
	• ease of installation:
	  ○ provision of installation package.

5.3 Risk to organisations of using digital systems	

5.3.1	Understand the potential risks to organisations when using digital systems:
	• security breaches:
	  ○ compromised confidentiality
	  ○ loss of integrity
	  ○ reduced availability
	• privacy breaches:
	  ○ personal information
	  ○ business information
	• regulatory and legal non-compliance
	• audience exclusion:
	  ○ biases
	  ○ poor user experience
	• emerging rival technologies
	• technical issues:
	  ○ reliance and system failure
	  ○ system not fit for purpose.

5.3.2	Understand the potential impact of risks to organisations when using digital systems: • legal action • fines • reputational damage • withdrawal of licence to practise • loss of business.
5.4 Technical change management	

5.4.1	Understand the internal factors that trigger change in organisations:
	• internal factors:
	  ○ organisational restructuring
	  ○ expansion
	  ○ downsizing
	  ○ new strategic objectives:
	  – diversification
	  – rebranding
	  – additional features or services.

5.4.2	Understand the external factors that trigger change in organisations:
	• political:
	  ○ change in government
	  ○ conflict
	  ○ shift in government priorities
	• economic:
	  ○ provision of new services
	  ○ recession
	  ○ inflation
	  ○ interest rates
	  ○ consumer trends
	  ○ new competitors
	  ○ entering new markets
	• social:
	  ○ changes in demographics
	  ○ market/social trends
	  ○ adapting to remote working
	  ○ cultural expectations
	• technological:
	  ○ emergence of new technologies
	  ○ retirement of obsolete technologies
	  ○ system failure
	  ○ zero-day vulnerabilities

	• legal:
	  ○ new legislation
	  ○ changes to legislation
	• environmental:
	  ○ sustainability issues
	  ○ pandemics
	  ○ natural disasters.

5.4.3	Understand how organisations can respond to change:
	• new or amended policies
	• new or amended business processes:
	  ○ change in staffing numbers
	  ○ change in delivery schedules
	  ○ change in opening hours
	• new or amended products or services:
	  ○ completely new products or services
	  ○ next generation products or services
	  ○ minor updates to existing products or services
	• new or improved digital systems:
	  ○ back-end systems
	  ○ customer-facing systems
	• improved training
	• restructuring:
	  ○ change in management structure
	  ○ redrawing of boundaries.

5.4.4	Understand the processes, benefits and drawbacks of the change management process:
	• identifying type of change:
	  ○ new system
	  ○ amendment to existing system
	• role of change advisory board (CAB):
	  ○ prioritise change requests
	  ○ review change requests
	  ○ stages of approval
	  ○ monitor change process:
	  – collate and analyse change data
	  – check change implementation
	  – take action to accelerate change
	  ○ provide feedback
	• identifying the changes to be made:
	  ○ using SMARTER objectives (specific, measurable, achievable, realistic, time-bound, evaluated, reviewed)

	• identifying impact of change:
	  ○ measure/forecast positive and negative impact
	  ○ analysis of positive and negative impact
	• allocation of resources:
	  ○ budget
	  ○ time
	  ○ staffing
	  ○ hardware and software
	• identifying and communicating potential risks and desired impact(s) to stakeholders:
	  ○ gain acceptance
	  ○ ensure compliance
	• configuration of the new system or process:
	  ○ integration with legacy systems
	  ○ maintaining service during change
	• importance of fully testing new systems:
	  ○ reproducibility of results
	  ○ test environment, including hardware and software
	• method of implementing change:
	  ○ parallel
	  ○ phased
	  ○ direct
	  ○ pilot
	• documenting the change process:
	  ○ ensuring requirement traceability, including responsibility and accountability
	  ○ maintaining up-to-date information
	  ○ recording of all decisions
	  ○ retaining change documentation
	  ○ user training manuals
	• importance of rollback planning:
	  ○ backup methodology
	  ○ backup location
	  ○ recover plan
	• identify training needs:
	  ○ new training requirements
	  ○ refresher courses
	• identify methods of monitoring progress:
	  ○ post-progress review
	• version control software.

5.4.5	Understand the factors that determine the feasibility of a digital project:
	• benefits and drawbacks:
	  ○ financial savings
	  ○ cost of implementing change
	  ○ impact on processes, including productivity gains, improved communication and security
	  ○ provision of new products
	  ○ impact on company reputation
	• risks:
	  ○ resistance to change from workforce
	  ○ misuse of new systems
	  ○ inadequate support for new system
	  ○ inadequate knowledge of new system
	  ○ disruption caused by implementation of new systems
	• constraints:
	  ○ budget
	  ○ time
	  ○ human resources and technological resources.

Content area 6: Data	
Students will develop fundamental knowledge and understanding of data relevant to digital software development in order to communicate with other professionals. Students will understand how to store, access, quality assure, manipulate, analyse and process data.	
6.1 Data, information and knowledge	
6.1.1	Know and understand the differences and relationships between: • data • information • knowledge.
6.1.2	Know and understand sources for generating data: • humans: surveys, forms • artificial intelligence (AI)/machine learning: dangers of feedback loop • sensors: temperature, accelerometer, vibration, sound, light, pressure • Internet of Things (IoT): smart objects (thermostats, lights, security camera, trackers • transactions: customer data, membership, timing, basket.
6.1.3	Know and understand ethical data practices and the metrics to determine the value of data: • quantity • timeframe • source • veracity.

6.1.4	Understand how organisations use data and information: • analysis to identify patterns • system performance analysis: load, outage, throughput, status • user monitoring: login/logout, resources accessed • targeted marketing: discounts, upselling • threat/opportunity assessment: competitors, security, compliance.
6.1.5	Understand the interrelationships between data, information and the way it is generated, and make judgements about the suitability of data, information and the way it is generated in digital software development.
6.2 Methods of transforming data	
6.2.1	Know and understand methods of transforming data: • manipulating • analysing • processing.
6.3 Data taxonomy	
6.3.1	Know the definition of each category, understand its purpose, and understand that data is categorised as: • quantitative • qualitative.
6.3.2	Know the definition for structured data, understand its purpose and understand that quantitative data is structured.
6.3.3	Know the definition for unstructured data, understand its purpose and understand that qualitative data is unstructured.
6.3.4	Know the definition for each representation and understand the representation of quantitative data: • discrete values • continuous values • categorical values.
6.3.5	Know and understand the properties of qualitative data: • stored and retrieved only as a single object • codified into structured data.
6.3.6	Understand the interrelationships between data categories, data structure and transformation, and make judgements about the suitability of data categories, data structure and transformation in digital software development.

6.4 Data types	
6.4.1	Know the definition of common data types and understand their purpose and when each is used: • integer • real • character • string • Boolean • date • Blob.
6.4.2	Understand the interrelationships between structured data, unstructured data and data type.
6.4.3	Understand the interrelationships between data type and data transformation.
6.4.4	Be able to make judgements about the suitability of using structured data, unstructured data, data types and data transformations in digital software development.
6.5 Data formats	
6.5.1	Know the definition of common data formats and understand their purpose and when each is used: • JSON • Text file • CSV • UTF-8 • ASCII • XML.
6.5.2	Understand the interrelationships between data format and data transformation and make judgements about the suitability of using data formats in digital software development.
6.6 Structures for storing data	
6.6.1	Understand the role of metadata in providing descriptions and contexts for data.
6.6.2	Know the definition of file-based and directory-based structures and understand their purposes and when they are used.
6.6.3	Know the definition of hierarchy-based structure and understand its purpose and when it is used.
6.6.4	Understand the interrelationships between storage structures and data transformation.

6.7 Data dimensions and maintenance	
6.7.1	Know the definitions of the six Vs (dimensions) and understand the six Vs (dimensions) of Big Data and their impact on gathering, storing, maintaining and processing: • volume • variety • variability • velocity • veracity • value.
6.7.2	Know the definition of Big Data and understand that it has multiple dimensions.
6.7.3	Understand the impact of each dimension on how data is gathered and maintained.
6.7.4	Know the definitions of data quality assurance methods and understand their purpose and when each is used: • validation • verification • reliability • consistency • integrity • redundancy.
6.7.5	Know and understand factors that affect how data is maintained: • time • skills • cost.
6.7.6	Understand the interrelationships between the dimensions of data, data quality assurance methods and factors that impact how data is maintained, and make judgements about the suitability of maintaining, transforming and quality assuring data in digital software development.
6.8 Data systems	
6.8.1	Know the definition of data wrangling and understand its purpose and when it is used.
6.8.2	Know and understand the purpose of each step of data wrangling: • structure • clean • validate • enrich • output.

6.8.3	Know and understand the purpose of each core function of a data system: • input • search • save • integrate • organise (index) • output • feedback loop.
6.8.4	Know the types of data entry errors and understand how and why they occur: • transcription errors • transposition errors.
6.8.5	Know and understand methods to avoid data entry errors: • validation of user input • verification of user input by double entry • drop-down menus • pre-filled data entry boxes.
6.8.6	Know and understand the factors that impact data entry: • time needed to create the screens • expertise needed to create screens • time needed to enter the data.
6.8.7	Understand the relationship between factors that impact data entry and data quality and make judgements about the suitability of methods to reduce data entry errors in digital software development.
6.8.8	Understand the relationship between factors that impact implementation of data entry and make judgements about the suitability of implementing data entry in digital software development.
6.9 Data visualisation	
6.9.1	Know and understand data visualisation formats and when they are used: • graphs • charts • tables • reports • dashboards • infographics.

6.9.2	Know and understand the benefits and drawbacks of data visualisation formats based on: • type of data • intended audience • brief.
6.10 Data models	
6.10.1	Know the types of data models and understand how they organise data into structures: • hierarchical • network • relational.
6.10.2	Know and understand the factors that impact the selection of data models for organising data: • efficiency of accessing individual items of data • efficiency of data storage • level of complexity in implementation.
6.10.3	Understand the benefits and drawbacks of different data models and make judgements about the suitability of data models based on efficiency and complexity.
6.10.4	Be able to draw and represent data models: • hierarchical models with blocks, arrows and labels • network models with blocks, arrows and labels • relational models with tables, rows, columns and labels.
6.11 Data access across platforms	
6.11.1	Understand the features, purposes, benefits and drawbacks of accessing data across platforms: ○ permissions: ○ authorisation ○ privileges ○ access rights ○ rules • access mechanisms: ○ role-based access (RBAC) ○ rule-based access control (RuBAC) ○ Application Programming Interfaces (API).
6.11.2	Know and understand the benefits and drawbacks of methods to access data across platforms.
6.11.3	Understand the interrelationships between data access requirements and data access methods and make judgements about the suitability of accessing data in digital software development.

6.12 Data analysis tools	

6.12.1	Know data analysis tools and understand their purpose and when they are used:
	• storing Big Data for analysis:
	  o data warehouse
	  o data lake
	  o data mart
	• analysis of data:
	  o data mining
	  o reporting
	• use of business intelligence gained through analysis:
	  o financial planning and analysis
	  o customer relationship management (CRM):
	  – customer data analytics
	  – communications.

6.12.2	Understand the interrelationships between data analysis tools and the scale of data.

Content area 7: Digital environments	
7.1 Hardware	
7.1.1	Understand the features and use of different types of physical computers: • personal computers • mobile devices (smartphones and tablets) • servers • embedded devices.

7.1.2	Understand the features and use of different types of hardware devices:
	• input devices
	• output devices
	• processors:
	  ○ number of cores
	  ○ clock speed
	  ○ cache size
	  ○ mobile processors
	• main memory:
	  ○ RAM (Random Access Memory)
	  ○ ROM (Read-only Memory)
	• secondary storage:
	  ○ magnetic
	  ○ solid state
	  ○ optical
	• motherboard

	• graphics processing units
	• network interface devices:
	  ○ PCI (Peripheral Component Interconnect)
	  ○ USB (Universal Serial Bus)
	• cooling:
	  ○ air cooling
	  ○ liquid cooling
	• sensors.

7.2 Software	

7.2.1	Understand the features and use of operating systems:
	• batch:
	  ○ non-interactive applications
	  ○ high volume
	  ○ scheduling
	• multitasking:
	  ○ concurrent execution of multiple tasks
	  ○ time-slicing
	  ○ interrupts
	• real-time operating system:
	  ○ monitoring and control applications
	  ○ transaction processing
	• network operating system:
	  ○ resource sharing
	  ○ user management
	  ○ communication
	• mobile operating system:
	  ○ smartphones and tablets
	  ○ lower processing requirements
	  ○ increased battery life.

7.2.2	Understand the features and use of common utilities: • file management • defragmenters • file compression • package managers • protection software • backup software.

7.2.3	Understand the features and use of common code development tools:
	• integrated development environments:
	  ○ code editing tools
	  ○ debugging tools
	  ○ screen design tools
	• compilers
	• interpreters.

7.2.4	Understand the features and use of common application software: • word processors • spreadsheets • databases • email • project management software.
7.3 Networks	
7.3.1	Understand the benefits and drawbacks of connecting devices to form networks.

7.3.2	Understand the features of different types of networks:
	• number of users
	• connection media
	• coverage media
	• network types:
	  ○ personal area network (PAN)
	  ○ local area network (LAN)
	  ○ metropolitan area network (MAN)
	  ○ wide area network (WAN)
	  ○ virtual private network (VPN).

7.3.3	Understand the features, characteristics, benefits and drawbacks of connectivity methods:
	• wired:
	  ○ copper/ethernet
	  ○ fibre-optic
	• wireless:
	  ○ wireless access points.

7.3.4	Understand the features, benefits and drawbacks of the common network topologies: • star • mesh • tree • logical versus physical.

7.3.5	Understand the features, benefits and drawbacks of different network models: • client-server • thin client • peer-to-peer.
7.3.6	Understand the role of common components of a network: • server • client • router • network switch • internet connection/internet backbone.
7.3.7	Understand the seven-layer OSI (open systems interconnection) model, including the function and related protocols of each layer: • application layer • presentation layer • session layer • transport layer • network layer • data link layer • physical layer.
7.3.8	Understand the four-layer TCP/IP (transmission control protocol/ internet protocol) model, including the function and related protocols of each layer: • application layer • transport layer • internet layer • network layer.

7.3.9	Understand the role of data packets in transmitting over a network, including:
	• contents and structure of a data packet
	• role of the components of a data packet
	• packet switching:
	  ○ causes of packet loss
	• error handling:
	  ○ cyclic redundancy check (CRC).

7.3.10	Understand the role of common network protocols:
	• web protocols:
	  ○ HTTP
	  ○ HTTPS
	• mail protocols:
	  ○ SMTP
	  ○ POP
	  ○ IMAP
	• routing protocols:
	  ○ RIP
	  ○ OSPF
	• application protocols:
	  ○ FTP
	  ○ SFTP
	  ○ DHCP
	  ○ DNS.

7.3.11	Understand the concepts of bandwidth and latency, and their effect on the performance of networks and connected systems.
7.4 Virtual environments	

7.4.1	Understand the role and characteristics of common virtual environment components:
	• virtual machines:
	  ○ clients (virtual PC, virtual switch, virtual router)
	  ○ servers
	• hypervisors:
	  ○ type 1
	  ○ type 2.

7.4.2	Understand the key features of virtual environments: • increased security • managed execution • sharing • aggregation • emulation • isolation • portability.

7.4.3	Understand the benefits of the use of virtual environments: • cost effectiveness for large environments • easy management • resilience • potentially lower carbon footprint • improved disaster recovery options • better testing environments • provision of education and training options.
7.4.4	Understand the drawbacks of the use of virtual environments: • extra hardware load • slower execution time • potential for false representation of performance.
7.5 Cloud environments	
7.5.1	Understand different types of cloud: • private • public.
7.5.2	Understand the benefits of the use of cloud: • portability • elasticity • fewer storage limitations • cost effectiveness.
7.5.3	Understand common cloud delivery models, their advantages and disadvantages, and the way in which responsibility and ownership of resources are distributed between the client and the cloud provider: • Infrastructure as a Service (IaaS): o client manages application software, system software (middleware and operating system), runtime, data and user accounts o cloud provider manages virtualisation and hardware (servers, network and storage) • Platform as a Service (PaaS): o client manages application software, data and user accounts o cloud provider manages virtualisation, hardware (servers, network and storage) and systems software (middleware and operating system) and runtime • Software as a Service (SaaS): o client manages user accounts and data o cloud provider manages virtualisation, hardware (servers, network and storage) systems software (middleware and operating system), runtime and application software.

7.6 Resilient digital environments	

7.6.1	Understand the benefits of resilient environments and the impact on organisations and clients:
	• increased security:
	  ○ data security (storage and transfer)
	  ○ reduced vulnerabilities
	• increased reputation:
	  ○ protect brand/image
	  ○ retain customer confidence
	• reduction in downtime.

7.6.2	Understand methods used to improve the resilience of digital environments:
	• software updates/upgrades:
	  ○ planned updates/upgrades
	  ○ patches in response to new vulnerabilities
	• hardware replacement:
	  ○ rolling replacement plans
	  ○ secure disposal
	• data and system redundancy
	• device hardening:
	  ○ removal of unneeded ports, applications, permissions and access
	• backup systems and recovery procedures:
	  ○ onsite
	  ○ remote/offsite
	  ○ cloud
	• hot, cold and warm sites
	• standard operating procedures:
	  ○ effective staff training
	  ○ induction
	  ○ new digital systems
	  ○ new or updated policies.

Content area 8: Security	
8.1 Security risks	

8.1.1	Know the type of confidential information held by organisations:
	• Human Resources:
	  ○ salaries and benefits
	  ○ staff personal details

	• commercially sensitive information:
	  ○ client details
	  ○ stakeholder details
	  ○ intellectual property
	  ○ sales numbers
	  ○ contracts
	• access information:
	  ○ usernames
	  ○ passwords
	  ○ multi-factor authentication (MFA) details
	  ○ personal identification number (PIN)
	  ○ access codes
	  ○ passphrases
	  ○ biometric data.

8.1.2	Understand why information must be kept confidential by organisations:
	• salaries and benefits:
	  ○ prevent competitors from offering higher wages to attract staff
	  ○ prevent employees from comparing salaries/demanding comparable pay
	• staff details:
	  ○ protect privacy
	  ○ prevent competitors from directly contacting them
	• intellectual property:
	  ○ prevent competitors from copying designs
	• client details:
	  ○ prevent competitors from contacting clients
	  ○ protect client privacy
	• sales numbers
	• access information:
	  ○ prevent unauthorised access.

8.1.3	Understand the potential impact to an organisation of failing to maintain privacy and confidentiality:
	• non-compliance with regulations:
	  ○ loss of licence to practise
	• loss of trust
	• damage to organisation’s image
	• financial loss:
	  ○ fines
	  ○ refunds
	  ○ loss of earnings/termination of contracts
	• legal action
	• reduced security.

8.2 Types of threats and vulnerabilities	

8.2.1	Understand potential technical threats and their impacts on organisations and individuals including prevention and mitigation methods:
	• botnets
	• denial of service (DoS)/distributed denial of service (DDoS)
	• malicious hacking:
	  ○ hacktivists/nation states/organised crime/individuals
	  ○ password cracking/brute force
	  ○ cross-site scripting
	  ○ SQL injection
	  ○ buffer overflow
	• malware:
	  ○ viruses
	  ○ worms
	  ○ key loggers
	  ○ ransomware
	  ○ spyware
	  ○ remote access trojans
	• social engineering:
	  ○ phishing
	  ○ spear phishing
	  ○ smishing
	  ○ vishing
	  ○ pharming
	  ○ watering hole attacks
	  ○ USB baiting
	• domain name server attack/redirection of traffic
	• insecure application programming interfaces (APIs)
	• man-in-the-middle attacks
	• open/unsecured Wi-Fi networks.

8.2.2	Understand potential technical vulnerabilities to systems and data:
	• inadequate security processes:
	  ○ weak encryption
	  ○ inadequate password policy
	  ○ failure to use multi-factor authentication
	• out-of-date components:
	  ○ hardware
	  ○ software (lack of support/compatibility with legacy systems, zero-day bugs)
	  ○ firmware.

8.2.3	Understand potential human threats, including prevention and mitigation methods, to systems and data:
	• human error:
	  ○ file properties
	  ○ confirmation boxes
	  ○ staff training
	• malicious employee:
	  ○ immediate removal from premises
	  ○ suspend user accounts immediately
	• disguised criminal:
	  ○ accompany all visitors
	  ○ check identification of visitors
	• poor cyber hygiene:
	  ○ locking all unattended machines
	  ○ not writing down passwords
	  ○ poor password management.

8.2.4	Understand potential physical vulnerabilities, including prevention and mitigation methods, to systems, data and information, including:
	• lack of access control:
	  ○ entry control systems
	• poor access control:
	  ○ do not allow tailgating
	  ○ use complex access codes
	  ○ change codes regularly
	  ○ monitor access areas
	  ○ audit of staff access to secure areas
	• nature of location:
	  ○ protect against shoulder surfing
	  ○ protect against the environment
	  ○ protect against vandalism
	• poor system robustness:
	  ○ rugged machines
	• natural disasters.

8.2.5	Understand the potential impact to an organisation of threats and vulnerabilities: • loss/leaking of sensitive data • unauthorised access to digital systems • data corruption • disruption of service • unauthorised access to restricted physical areas.

8.3 Threat mitigation	

8.3.1	Understand the purposes, processes, benefits and drawbacks of common threat mitigation techniques:
	• security settings:
	  ○ hardware
	  ○ software
	• anti-malware software:
	  ○ function
	  ○ actions
	• intrusion detection
	• encryption:
	  ○ hashing
	  ○ symmetric
	  ○ asymmetric
	• user access policies
	• staff vetting
	• staff training
	• software-based access control
	• device hardening
	• backups:
	  ○ type (full, incremental, differential)
	  ○ safe storage
	• software updates
	• firmware/driver updates
	• air gaps
	• certification of APIs (application programme interfaces)
	• VPNs (virtual private networks)
	• multi-factor authentication
	• password managers
	• port scanning
	• penetration testing:
	  ○ ethical hacking
	  ○ unethical hacking.

8.3.2	Understand the processes and procedures that assure internet security and the reasons why they are used:
	• firewall configuration:
	  ○ rules for traffic (inbound and outbound)
	  ○ traffic type rules
	  ○ application rules
	  ○ IP address rules
	• network segregation:
	  ○ virtual
	  ○ physical
	  ○ offline network
	• network monitoring
	• port scanning.

8.4 Interrelationship of components required for effective security	

8.4.1	Understand how the relationships in the CIA triad interrelate:
	• confidentiality:
	  ○ ensuring that data is kept private by controlling who has access to the data
	• integrity:
	  ○ ensuring that the data has not been tampered with; this can be done by maintaining confidentiality
	• availability:
	  ○ ensuring that data is available and useful; this can be done by ensuring integrity.

8.4.2	Understand the elements of the Identification Authentication Authorisation Accountability (IAAA) model, including the techniques used and their benefits and drawbacks:
	• identification:
	  ○ recognising the individual within a digital system
	  ○ knowledge-based identification, including username
	  ○ possession-based identification methods
	  ○ biometric-based ID methods
	• authentication:
	  ○ verifying the identity claimed during the identification phase
	  ○ multi-factor authentication methods
	  ○ passwords and passphrases
	  ○ biometric authentication
	• authorisation:
	  ○ ensuring that authenticated users can only access resources and perform actions that they are permitted to
	  ○ role-based, using the role of the user within the digital system
	  ○ access control lists

	• accountability:
	  ○ ensuring that any actions within a system can be traced back to the responsible user
	  ○ audit logs
	  ○ user activity.

Employer Set Project

Pre-task – Familiarisation with the industry context E1 E4 E6 M1 M2 M7 D1 D3 D5

The purpose of the pre-release task is to allow students the opportunity to familiarise themselves with the context of the main brief, i.e. the use of digital solutions in the financial sector.

Students are encouraged to carry out independent investigation and share findings with others, work in groups and communicate with others, and take part in and lead discussions.

While the pre-release task is not directly assessed and not taken under controlled conditions, students will be able to use your research to inform responses to the assessed tasks.

Task 1 – Planning a project

Project planning tools

Be able to use project planning tools to apply understanding of project planning in response to a scenario.

Make use of given template (provided by Pearson) to produce a project plan containing a Gantt chart and a resource and cost plan.

Gantt chart E4 E5 M1 M2 M3 M5 M8 M9 M10 D1 D2 D4

- Assess the strengths and skills of people and assign appropriate tasks to them.
- Make scheduling decisions in response to a defined deadline.
- Prioritise activities or tasks based on analysis of requirements.
- Demonstrate how to correctly and appropriately assign resources to project tasks.
- Use an appropriate software development lifecycle to efficiently organise project tasks.

Resource and cost plan E1 E4 M1 M2 M4 M8 D1 D2 D4

- Identify and calculate costs of a project, including:
  - materials
  - physical resources
  - personnel.
- Select and allocate resources to the resource list and correctly attribute costs to provide an accurate estimate of the total project cost.
- Determine the affordability and viability of implementing a project and its impact on a company over time.

Rationale E2 E3 E4 E5

- Consider the factors that are most relevant when planning projects.
- Justify notable project planning decisions made (particularly those that will have significant impact on the outcomes of the project), with consideration given to:
  - order and timing of tasks
  - allocation of team members
  - potential benefits and risks
  - impact of decisions on timings and costs.

Task 2 – Identifying and fixing defects in existing code

Use of testing to identify defects

- Assess the given code against requirements.
- Carry out testing to identify issues in given code.
- Perform any remedial actions required, justifying any decision made when fixing the defect.

Documenting the testing process E1 E4 M4 M8 D1 D2 D4

Provide annotated evidence of testing, including:

- identifying tests to be carried out
- describing the purpose of the identified test
- identifying test data to be used (valid, valid extreme, invalid, invalid extreme, erroneous)
- describing the expected results
- describing the actual results of the tests performed
- comparing the actual results of testing with the expected results
- describing any further actions that are required
- refining the system as required.

The solution M4 M5 M7 D2 D4 D6

- Correct errors in code and add or remove code to ensure that the given code is functional and meets the given requirements.
- Follow appropriate programming conventions when fixing code to ensure that it makes use of precise logic and programming structures throughout, so that the program produces consistently correct outcomes.

Task 3 – Designing a solution

Decomposition of the problem E1 E2 E3 E4 E5 M1 M5 M6 D1 D2 D4

- Break down the problem into smaller parts suitable for computational solutions, justifying any decisions made. Make effective use of detailed abstraction and refinement.
- Use elements of reusable components.
- Use good decomposition to show all the necessary sub-systems that make up the main solution.
- Use appropriate tools for communicating algorithms (flowcharts, code).

Application of logical thinking M1 D2

- Describe the parts of the solution using algorithms.
- Clearly define the steps.
- Uniquely define each step. They should depend on the input and the result of the preceding steps.
- Ensure the algorithm makes use of key constructs (e.g. sequence, selection and iteration).

Use of conventions E3 M4

- Demonstrate correct use of structure and convention for the chosen method of communication (flowcharts, code), such as correct use of symbols for flowcharts and key words used in code.
- Select and make consistent use of appropriate naming conventions throughout.

Communication of the design E4 M1 D4

- Ensure design documents are of sufficient detail to:
  - effectively communicate the intended solution
  - allow the client to make informed decisions
  - allow a third party to use design documents to create the proposed solution.
- Communicate intended solution effectively and clearly, with use of:
  - appropriate combination of written and diagrammatical presentation
  - appropriate use of technical vocabulary
  - consideration of audience
  - explanations of structures and process in the design.

Task 4a – Developing a solution

The solution E1 E5 M2 M4 M5 M6 M7 D1 D4 D6

Apply an undertaking of programming to develop a solution that meets the requirements of a brief, including:

- refining the system as required
- demonstrating an appropriate level of technical skill and understanding of programming techniques and problem solving
- use of pre-written and user-written modules with appropriate interface.

Code organisation M4

Ensure code produced for the solution is appropriate to meet the demands of the brief, including:

- trying to avoid multiple pages of nested if-clauses and for-loops with a lot of copy-pasted procedural code
- clear meaningful indentation
- precise use of logic functions, classes or objects, with proper structure
- comments whenever possible to help explain the logic
- good use of local variables and minimal use of global variables
- use of constants
- well-designed interface

- consistent style throughout
- defensive programming and handling data securely
- good exception handling.

User experience E4

Meet user needs, with consideration of:

- consistency of the product
- simplification of user input (e.g. coding)
- meaningful output messages (e.g. explanation of what to input, outputting results in a meaningful way, error messages)
- visualisation of data (e.g. use of Matplotlib to produce graphs of results, show trends and patterns).

Note:

- Formal documentation of testing is not required in this activity.
- Testing by the student should still be carried out but its use should be implicit through:
  - a working product
  - a product that meets the requirements detailed in the given task brief.

Task 4b – Reflective evaluation

Programming outcomes E1 E2 E3 E4 D3

- Be able to apply reflection and evaluation techniques.
- Provide evidence that the product meets brief requirements:
  - include measures against success criteria
  - provide evidence that the product meets user needs from testing
  - discuss how it could be improved if the problem was revisited and given detailed consideration.

Scheme of Assessment – Core Component

There are three assessments in the Core component of the *T Level Technical Qualification in Digital Software Development*:

- Core Examination Paper 1
- Core Examination Paper 2
- Employer Set Project.

The mapping, timings, scheduling and preparation for the assessments shown below are for the current specimen assessment material. The actual live assessments will have the same overarching number of tasks and overall focus. However, the order of tasks and the details within the tasks may change each series.

Core examination

Paper 1
Written examination: 2 hours 15 minutes **30% of the core assessments** **90 marks**
Content overview 1. Problem solving 2. Introduction to programming 3. Emerging issues 4. Legislation and regulatory requirements
Assessment overview A written examination comprising two sections, A and B. Students answer all questions in each section. Each section of the examination will get more challenging as the student progresses by ramping up demand and difficulty in a manner broadly similar to the other sections. Each section will be assessed through a combination of: • short open response items • medium open response items • extended open response questions. The examination is: • set and marked by Pearson • timetabled at a time and on a date specified by Pearson.
Administration This paper must be assessed under examination conditions following JCQ's Instructions for Conducting Examinations (ICE).

Paper 2
Written examination: 2 hours 15 minutes **30% of the core assessments** **90 marks**
Content overview 5. Business context 6. Data 7. Digital environments 8. Security
Assessment overview A written examination comprising two sections, A and B. Students answer all questions in each section. Each section of the examination will get more challenging as the student progresses by ramping up demand and difficulty in a manner broadly similar to the other sections. Each section will be assessed through a combination of: • short open response items • medium open response items • extended open response questions. The examination is: • set and marked by Pearson • timetabled at a time and on a date specified by Pearson.
Administration This paper must be assessed under examination conditions following JCQ's Instructions for Conducting Examinations (ICE).

Core Examination Assessment Objectives

Assessment Objective		Paper 1 (Marks/%)	Paper 2 (Marks/%)
AO1a	Demonstrate knowledge and understanding of the content (knowledge)	3 (3.3%)	10 (11.1%)
AO1b	Demonstrate knowledge and understanding of the content (understanding)	27 (30%)	21 (23.3%)
AO2	Apply knowledge and understanding of the content to different situations and contexts	39 (43.3%)	38 (42.2%)
AO3a	Analyse information and issues related to the content	12 (13.3%)	12 (13.3%)
AO3b	Evaluate information and issues related to the content	9 (10%)	9 (10%)

Paper 1	AO1a	AO1b	AO2	AO3a	AO3b
Section A	3	21	3	3	0
Section B	0	6	36	9	9
Total 90 marks	30		39	21	

Paper 2	AO1a	AO1b	AO2	AO3a	AO3b
Section A	6	18	3	3	0
Section B	4	3	35	9	9
Total 90 marks	31		38	21	

Employer Set Project

Employer Set Project
Externally assessed project: 14 hours 30 minutes **40% of the core assessments** **100 marks**
Content overview When responding to the Employer Set Project, students will need to draw upon knowledge and understanding from across the core content in a synoptic manner in order to effectively respond to a brief within a vocational context.
Assessment overview There is a pre-release task that is not assessed. There are five parts to the assessment: • Task 1: Planning a project • Task 2: Identifying and fixing defects in an existing code • Task 3: Designing a solution • Task 4a: Developing a solution • Task 4b: Reflective evaluation. Students will undertake the assessed elements of the project tasks under supervised conditions. Internet access is not permitted. Students may not use AI or any other tool designed to prepare a response. The assessment will take place over multiple sessions up to a combined duration of 14 hours 30 minutes. The project outcomes will consist of a portfolio of evidence submitted electronically. Students will undertake a project in response to a realistic contextual challenge. The project is validated by an employer panel, taking into account the client's requirements and the user experience. The project will consist of planning documentation, an annotated digital portfolio, prototype digital product, testing evidence and evaluation. The project will be set and marked by Pearson.
Administration Providers must follow the guidance in the following: • General Administrative Support Guide • Administration Support Guide for the specific Technical Qualification Employer Set Project (if applicable). These are located on the Training and Admin Support webpage.

Employer Set Project Assessment Objectives

Assessment Objective			Proportion
AO1	**Planning**	Plan an approach to developing solutionsto solve problems in response to a brief.	12%
AO2	**Application**	Apply knowledge and skills to develop software, create an artefact, fix defects and mitigate risks to security.	41%
AO3	**Selecting relevant techniques and resources**	Select relevant tools, techniques and resources to respond to a brief and work in a collaborative environment.	9%
AO4	**a. English skills**	Use appropriate English skills to communicate technical information to both technical and non-technical audiences.	3%
	b. Maths skills	Use appropriate maths skills to realise a project outcome in response to a brief.	
	c. Digital skills	Use appropriate digital skills to realise a project outcome in response to a brief and communicate technical information to both technical and non-technical audiences.	
AO5	**a. Project outcome**	Realise a project outcome by producing software and artefacts in response to a brief.	26%
	b. Review	Review how well digital solutions meet a brief, using reflective evaluation.	9%

Resources for the delivery of the Core component content

The following resources are required for the delivery of the Core component for this Technical Qualification:

- diagramming/flowcharting software
- Python 3 (version 3.10 or later) and an appropriate IDE
- Python libraries, in addition to standard libraries or other libraries that provide the same/comparable functionality/capabilities:
  - Pandas
  - Matplotlib
- spreadsheet software (for example, MS Excel, Google Sheets) for:
  - the handling and manipulation of datasets
  - producing a dashboard
  - producing project plans (Gantt chart and costings).

General:

- computer
- internet access
- audio/visual recording equipment
- software:
  - word processing (for example, MS Word, Google Docs)
  - presentation (for example, MS PowerPoint, Google Slides)
  - project management (for example, MS Excel, MS Project)
  - basic image-editing software (for example, Photoshop, GIMP)
  - programming software (for example, TextEdit)
  - database software (for example, MS SQL, phpMyAdmin)
  - web browsers (for example, Chrome, Firefox, Edge)
- data sources (for example, online, social media, analytical)
- research resources (for example, online, books, journals)
- a web server.

4 Occupational Specialisms

Digital Software Development

Content area 1: Be able to analyse a problem to define requirements and acceptance criteria aligned to user needs

What underpinning knowledge do students need?	

1.1	**Understand the stages of the software development lifecycle and be able to apply them to digital projects**
	• Lifecycle stage – research and familiarisation:
	  ○ explore and understanding the initial client request/project brief
	  ○ carry out research relating to the specific context and market environment to inform the planning and design of a digital project, including:
	  – common problems and risks
	  – current uses of hardware and software within the identified context
	  – newly emerging technologies
	  – existing or potential solutions and how these meet different user needs
	  – industry/situational-specific guidelines and regulations
	  – identify shortfalls in own skills and knowledge and plan learning opportunities to make up these shortfalls.
	• Lifecycle stage – planning and requirement analysis:
	  ○ explore the requirements of a given brief and define the scope of an intended solution in the form of a project proposal, including:
	  – identify business requirements such as new software/amending/increasing security
	  – assess the measurable value of the proposed solution in relation to:
	  ▪ the user
	  ▪ the client/business
	  – apply computational thinking (decomposition, pattern recognition and abstraction) to split the problem into discrete objects
	  – define the functional and non-functional requirements of the solution
	  – define the key performance indicators (KPIs) of the solution
	  – identify the performance constraints in digital projects
	  – creation of user acceptance criteria
	  – schedule projects (tasks, subtasks, milestones)
	  – allocate appropriate resources to digital projects
	  – estimate costs of digital projects

	– choose programming language(s) for digital projects based on key criteria – suitability to the proposed task – organisational policy – scalability – security – availability of trained staff – costs – reliability – identify risks and explore ways to mitigate these risks. • Lifecycle stage – performing a user analysis: o select and use business analysis models to aid problem solving: – user stories – activity diagrams – mind maps – product road maps – process diagrams – entity relationship diagrams. • Lifecycle stage – designing the product: o create interface designs o plan how to solve key problems (design algorithms). o create data requirements designs o use generative AI tools to produce sections of code for inclusion in a larger solution o review the output of generative AI and refine as required to meet the identified requirements. • Lifecycle stage – developing and testing the product: o create a prototype o plan and implement appropriate testing, including: – module testing – system integration testing – automated testing – user testing and feedback. • Lifecycle stage – deploying/implementing the product: o install and configure the product o update the product o document key changes/iterations o ensure suitable version control • Lifecycle stage – maintenance: o provide system support o provide user support o carry out bug fixing o arrange/carry out user training o release updates to enhance the product.

	• Understand the value to the organisation of the digital product and the role the software development lifecycle plays. (E1, E2, E3, E4, E5, M2, M6, M7, M8, M9, M10, D1, D2, D3, D4)
1.2	**Understand the roles and responsibilities of the digital team within the software development lifecycle** • Product owner/client – sets and communicates the requirements of the product. • Scrum master – facilitates the team to maintain team cohesion, ensuring the team has access to resources they require. • Technical lead – provides the technical guidance and support for the project team. • Project manager – responsible for planning, organising, managing (budget, scope, schedule, risk and quality) all phases of a project. • Software development team: o systems analyst – analyses the current system and provides the requirements for the new system – uses the requirements to design the new software solution o UX/UI designer – interviews users, researches market data and gathers findings to design the user interface o software developer/engineer – uses the designs to create and maintain a working solution that is usable, secure and stable o operations engineer – ensures the stability of the product o security engineer – ensures the security of the product. • Software testers – responsible for the quality assurance of the software and development. (E4, E5)
1.3	**Understand the key features of, and be able to select, appropriate project methodologies when developing a software solution** • Key features of Agile: o incremental delivery models: – sprint – epic – story – spikes o emphasis on producing high-quality roducts with initially limited functionality o each increment delivers additional functionality o requirements can be continually altered throughout the project o can lack formal documentation o users/clients see working products at each iteration o cost-effective method to get an initial product to market o cancelled/partially completed projects will still have resulted in some useable code/product(s).

	• Key features of Scaled Agile:
	  ○ expanded incremental delivery models:
	  – sprint
	  – epic
	  – story
	  – spikes
	  – product increments
	  ○ emphasis on producing high-quality products with initially limited functionality
	  ○ each increment delivers additional functionality
	  ○ requirements can be continually altered throughout the project
	  ○ provides cohesive reporting
	  ○ users/clients see working products at each iteration
	  ○ cost-effective method to get an initial product to market
	  ○ cancelled/partially completed projects will have still resulted in some useable code/product(s)
	  ○ final iteration focuses on stability.
	• Key features of Waterfall:
	  ○ rigidly structured with systematic steps
	  ○ progress measured through the number of artefacts completed
	  ○ high initial costs with slower return on investment
	  ○ products cancelled during early stages may result in no useable code/product(s)
	  ○ limited user/client interaction
	  ○ high importance placed on documentation.
	• Key features of Rapid Application Development (RAD):
	  ○ focus on quick creation of prototypes which are systematically improved
	  ○ emphasis on developing all features first and quality second
	  ○ users/clients may see only partially working products or mock ups during early iterations
	  ○ relies heavily on reuse of previously developed code
	  ○ suitable for small- to medium-scale projects.
	• Key features of the Lean methodology:
	  ○ emphasis on reducing ‘waste’ in software (and projects) which may include:
	  – unnecessary or incorrect features
	  – repeated tasks
	  – unnecessarily complex solutions
	  – ineffective communication
	  – unnecessary changes
	  ○ decisions are made last minute to reduce chance of waste
	  ○ short iteration cycles with fast delivery of working versions
	  ○ emphasis placed on producing iterations that are fit for use and not specifically on delivering all features or requirements

	○ suitable for projects with small teams and when resources are limited ○ understand features and approaches of User Centred Design (UCD) and how it is used within a software development lifecycle: – considerations for UCD: ▪ who is the user? ▪ what does the user want to achieve by using the product? ▪ how does the user interact with the product? ▪ when does the user interact with the product? ▪ why is the product being used? ▪ what is the user experience? – characteristics of UCD (empathetic, iterative, interdisciplinary). – stages of the UCD iterative approach: ▪ understand the use context ▪ specify the user requirements ▪ design the solution ▪ assess against requirements. (E5, D1, D3, D6)
1.4	**Understand and define the functional and non-functional requirements of a software solution** • An understanding of the concept of 'secure by design' and how this influences the decisions made regarding functional and non-functional requirements. • An understanding of and ability to define functional requirements of a software solution: ○ the inputs required ○ the data needed ○ the data processing that must take place ○ the logic of the system ○ the deployment and usage platforms for the software. • An understanding of and ability to define non-functional requirements of a software solution in terms of: ○ security considerations ○ required accessibility features ○ scalability requirements ○ key performance indicators and metrics in relation to: – responsiveness – load handling – reliability ○ user acceptance criteria. • Understand the use of 'spike testing' as an early product-testing method to establish requirements and determine the extent of the problem and the required scope of a software solution. (E1, E5, M2, M6, M7, M8, D2, D3, D6)

1.5	**Investigate the current and potential uses of emerging technologies and how they impact industries**
	• Investigate impact on software development of emerging technologies such as:
	  ○ operational technology (OT)
	  ○ artificial intelligence (AI) and virtual intelligence (VI)
	  ○ subsets of AI and VI including:
	  – conversational and generative AI
	  – machine learning
	  – object recognition
	  – computer vision
	  ○ the Internet of Things (IoT)
	  ○ biometrics
	  ○ robotics and automation
	  ○ cloud services and platforms
	  ○ data lakes and data warehousing
	  ○ drones
	  ○ 3D printing
	  ○ developments in communications networks (e.g. 5G, increased availability of broadband networks).
	(E1, E2, E3, E4)

1.6	**Understand how to identify and be able to address personal training needs that can boost job performance during the software development process**
	• Considerations to ensure the software developer is able to complete the required solution:
	  ○ identify what further knowledge is needed
	  ○ identify what new skills are needed
	  ○ determine whether the software developer has the ability to complete the required solution.
	• Use different methods to address personal training needs required to complete the project:
	  ○ undertake coaching from a professional or peer
	  ○ learn new skills on the job
	  ○ carry out self-study
	  ○ use online professional forums
	  ○ sign up to internet workshops for additional training.
	(E5, E6, D1, D3, D6)

Content area 2: Apply ethical principles and manage risks in line with legal and regulatory requirements when developing software

What underpinning knowledge do students need?	

2.1	**Investigate the legal and regulatory requirements that apply to developing software**
	• Investigate and apply legal and regulatory considerations appropriate to the context and market environment in which they are developing software:
	  ○ intellectual property rights and licences
	  ○ consumer protection
	  ○ age ratings and classifications
	  ○ advertising laws
	  ○ data protection and privacy
	  ○ copyright and patent
	  ○ gambling legislation
	  ○ responsibilities concerning staff and employment practices
	  ○ territorial restrictions
	  ○ system security
	  ○ equality and diversity
	  ○ investigate and apply standards for software development
	  ○ ISO/IEC/IEEE 90003:2018
	  ○ W3C.
	• Investigate and consider ethical implications that apply to software development:
	  ○ code of conduct
	  ○ professional practice
	  ○ software licensing
	  ○ inclusion and diversity.
	(E5, M4, M5, M6, D1, D5, D6)

2.2	**Identify and manage risks that apply to software development**
	• Assess the potential risks associated with developing a software product appropriate to the context and market environment in which it is being developed:
	  ○ risk identification in terms of:
	  – data and system security (malicious versus accidental damage)
	  – compatibility with other systems
	  – speed of development
	  – meeting functional and non-functional requirements
	  – meeting key performance indicators (KPIs)
	  – legal and ethical considerations
	  – user engagement
	  – product reach

	○ assessment of risk (likelihood versus seriousness) ○ potential impact of risk ○ potential ways to mitigate identified risks ○ contingency planning ○ ongoing monitoring ○ investigate policies and procedures that apply to software development to manage and mitigate risks ○ backup ○ security ○ confidentiality, integrity and availability (CIA) ○ personnel, skills and training ○ business continuity planning ○ disaster recovery planning. • Be able to make and justify software development decisions and recommendations based on effective assessment of risk versus reward in relation to the context and market environment in which they are developing software. (E5, M4, M5, M6, D1, D5, D6)

Content area 3: Discover, evaluate and apply reliable sources of knowledge

What underpinning knowledge do students need?	

3.1	**Understand and evaluate the reliability of different sources of knowledge**
	• Use different sources to find reliable information:
	  ○ use search engines to find reliable websites
	  ○ read wikis
	  ○ read blogs
	  ○ read academic papers
	  ○ talk to peers
	  ○ join forums
	  ○ look at code comments
	  ○ use code repositories.
	• Evaluate the reliability of different sources:
	  ○ reputation – find out who the author is and whether they are credible
	  ○ understand bias – sources written by a particular individual/organisation
	  ○ look at the evidence used to support the content of the digital source
	  ○ cross-referencing/triangulation – compare to other sources
	  ○ check how current the content is – date website was last updated.
	(E1, E4, E5, E6, M2, M4, M6, M7, M8, D1, D3, D4, D5, D6)

3.2	**Select and use techniques to obtain qualitative and quantitative data to be able to evaluate software solutions** • Verbal feedback (formal and informal). • Create and deliver surveys/questionnaires. • Performance and use data. • Conduct user observation and complete observation records. • Create a focus group that represents a cross-section of the target audience. • Conduct interviews to gather individual thoughts on a software solution. • Use of peer mentoring. • Formal line management and appraisal procedures. (E1, E4, E5, E6, M2, M4, M6, M7, M8, D1, D3, D4, D5, D6)

Content area 4: Design

What underpinning knowledge do students need?	

4.1	**Understand the use of common design approaches**
	• Be able to make use of critical thinking in order to select appropriate design approaches when developing a software product and apply them within a larger project methodology:
	  ○ understand features and approaches of function orientated (top-down) design:
	  – consideration of how data flows through the system
	  – system made up of many sub-systems (functions)
	  – use of data flow diagrams to show how each function handles or changes the data
	  – problem is logically broken down (divide and conquer) based on what each function should do within the whole system
	  ○ understand features and approaches of object-oriented programming (OOP) design:
	  – core concept of making code reusable through standard OOP structures (methods, classes, objects/instances)
	  – characteristics of OOP (encapsulation, data abstraction, polymorphism and inheritance)
	  – common OOP design patterns (creational, structural, behavioural)
	  ○ understand features and approaches of data model design:
	  – visualisation of the data needed and how it will be organised
	  – common data models (conceptual, logical, physical, hierarchical, relational)
	  – common data modelling tools:
	  ▪ Entity Relationship Model (E-R Model)
	  ▪ Unified Modelling Language (UML)
	  ○ understand features and approaches of test-driven development (TDD):
	  – core concept that each new feature starts by writing a test that defines the improvements or functions of that feature
	  – common approaches to test-driven development:
	  ▪ add a test
	  ▪ run all tests until the new test fails
	  ▪ write some code
	  ▪ run tests and refactor code
	  ▪ repeat
	  ○ understand features and approaches of behavioural-driven development (BDD):
	  – core concept that development is informed by the required behaviour of a software unit which is specified before coding starts
	  – desired behaviours must have business value and relate to specified requirements

	– common behaviour specification structure (title, narrative, acceptance criteria) ○ understand features and approaches of functional design: – core concept that a program is built and structured as a set series of modules that perform a single defined process/function – characteristics of functional programming (recursion, closures, first-class functions, higher-order functions, anonymous functions, currying) – common components of functional design (arguments, statements, blocks, procedures, functions/sub-routines). (E1, E2, E3, E4, E5, M1, M2, M3, M4, M5, M6, M7, M8, M10, D1, D2, D4, D6)
4.2	**Be able to select platforms used for source code and content management** • Understand the features of different software development platforms used at different stages of developing a software product: ○ coding ○ repositories ○ branching ○ building ○ testing ○ deployment. • Understand the differences between proprietary and open-source platforms and how these may affect software design. • Understand the process required to manage the development workflow (gitFlow, gitHubFlow). • Understand the differences between different platforms and be able to make informed decisions about which to use and when in relation to: ○ target audience ○ budget ○ technical features ○ staff and training ○ ease/speed of development ○ platform updates ○ security ○ reliability ○ performance ○ compatibility. (E1, E3, E5, M10, D1, D2, D3, D4, D5, D6)

4.3	**Understand and be able to design a software solution**
	• Understand and apply user experience (UX) design principles:
	  ○ consistency to ensure a product is intuitive for the user, aesthetically pleasing and promotes brand recognition
	  ○ information hierarchy to navigate the product more easily
	  ○ visual hierarchy to enable more important content to stand out
	  ○ confirmation to ensure the user is aware of any actions performed and their impact
	  ○ user control to allow navigation of the product, efficient workflow and error correction
	  ○ accessibility to ensure the product is appropriate for a wide range of users.
	• Understand and apply user interface (UI) design principles:
	  ○ wireframes
	  ○ style guides
	  ○ clickable prototype.
	• Understand the features of content management systems and how they are used during design and development:
	  ○ search engine optimisation (SEO)
	  ○ page/screen management
	  ○ social media integration
	  ○ analytics tools
	  ○ workflow management
	  ○ publishing controls
	  ○ security features and management
	  ○ versioning and rollback
	  ○ content repositories
	  ○ open APIs
	  ○ multilingual support
	  ○ support.
	• Create program designs using accepted conventions:
	  ○ pre-defined code
	  ○ flowcharts using standard BCS symbols
	  ○ pseudocode using standard notation
	  ○ control structures (sequence, selection/branching, iteration)
	  ○ data types
	  ○ data validation
	  ○ data structures.
	• Considerations when selecting assets (graphics, audio, video, code) to be used in software design including:
	  ○ file types
	  ○ file size
	  ○ compression

	○ stream or encode audio ○ stream or embed video ○ use of metadata ○ quality required ○ bandwidth and storage available ○ target platform. • Understand and follow legal and ethical ways of working when using AI to produce content and assets for inclusion in a software solution. • Understand the features of different target platforms and how they affect the design and development of a software solution: ○ operating systems ○ file systems ○ server and other infrastructure (physical, virtual) ○ programming language stack ○ mobile and web. • Understand how and when to use databases to support software solutions: ○ user management ○ e-commerce tasks (stock, order processing, page personalisation) ○ diagnostics ○ performance analysis. • Create database designs to support software solutions using: ○ data dictionary/library ○ entity relationship diagram (ERD) ○ normalisation to third normal form. • Identify and understand network integration points: ○ which data is processed locally, e.g. user input for computer games ○ which data is processed remotely, e.g. actions of all players in multi-player environments ○ how data is transferred between local and remote sources, e.g. remote system calls ○ how local and remote systems will be connected, e.g. type of network, combination of networks ○ system boundaries to identify which tasks are carried out locally or remotely ○ which external systems to integrate with, e.g. OPTA for sports data. (E1, E2, E3, E4, E5, M1, M2, M3, M4, M5, M6, M7, M8, M10, D1, D2, D3, D4, D6)

Content area 5: Create solutions in a social and collaborative environment

What underpinning knowledge do students need?	

5.1	**Understand the reasons for using collaborative techniques**
	• Team working on common projects that allows for:
	  ○ a reduction in development time
	  ○ better communication
	  ○ sharing of knowledge
	  ○ development of software development skills
	  ○ code reviews:
	  – paired programming
	  – informal walkthroughs
	  – formal inspections.
	• Understand when to collaborate and when working independently is more appropriate.
	(E5, M10, D1, D3)

5.2	**Understand and be able to select appropriate technologies used in a social and collaborative environment**
	• Collaborative technologies to aid working as part of a team:
	  ○ communication (e.g. email, instant messaging)
	  ○ resource management (e.g. cloud storage, backup, synchronisation)
	  ○ knowledge (collaboration hubs, wikis, community forums, news sites)
	  ○ documentation for technical and non-technical audiences.
	• Code collaboration technologies.
	• Version control.
	• Source control.
	• Integrated development environments (IDEs).
	(E5, M10, D1, D3, D6)

Content area 6: Implement a solution using at least two appropriate languages

What underpinning knowledge do students need?	

6.1	**Select and use languages to create a software solution for a software project appropriate to the context and market environment in which they are developing software**
	• Select and use at least **two** appropriate languages from the following list to implement front-end and back-end solutions:
	  ○ Python 3 (version 3.10 or later)
	  ○ C#
	  ○ SQL
	  ○ Javascript
	  ○ PHP.
	• Be able to select and use appropriate tools, application programming interfaces (APIs), packages, modules and libraries to add functionality and compatibility:
	  ○ generate dynamic page content
	  ○ containerisation
	  ○ stateful versus stateless components
	  ○ form handling
	  ○ file and data handling:
	  – local files
	  – create, open, read, write, delete and close files on a server
	  – send and receive cookies
	  – add, delete, modify data in a database
	  ○ interface components
	  ○ media content
	  ○ adaptive/responsive layout
	  ○ working with existing applications, operating systems, cloud-based and traditional platforms
	  ○ working with specific devices
	  ○ communication over a network
	  ○ infrastructure as code
	  ○ security features:
	  – control user access
	  – encrypt data.
	• Packaging and organising front-end and back-end code as a single usable product (e.g. use of HTML, CSS and related frameworks to deploy a solution, compiling and encapsulating as a single executable file).
	• Understand the use of continuous integration pipelines and how they can be used to build and deliver software solutions:
	  ○ the concept of ‘continuous integration – continuous deployment’ (CI/CD)

	○ common stages of continuous integration pipelines: – source code control – build automation – unit test automation – deployment automation – monitoring. • Use common coding conventions: ○ naming conventions ○ code annotations/commenting ○ modularisation ○ structure/indentation ○ version control. • Use good practice when developing digital products (12 factor principles): ○ use one codebase tracked in revision control, many deploys ○ explicitly declare and isolate dependencies ○ store config in the environment ○ treat backing services as attached resources ○ strictly separate build and run stages ○ execute the app as one or more stateless processes ○ export services via port binding ○ scale out via the process model ○ maximise robustness with fast startup and graceful shutdown ○ keep development, staging and production as similar as possible ○ treat logs as event streams ○ run admin/management tasks as one-off processes. (E5, M1, M2, M3, M4, M5, M6, M7, M8, M10, D1, D2, D3, D4, D6)
6.2	**Select and use appropriate tools and features to create a user interface that applies user experience (UX) design principles** • Be able to select and use appropriate user interface features: ○ images and animation ○ audio ○ effects ○ interactions: – user input – output/user feedback: ▪ textual ▪ graphical ▪ audio ▪ haptic ○ data visualisation: – dashboard – graphing – data presentation.

	• Be able to select and use appropriate user interface techniques:
	  ○ layout grids
	  ○ layout and use of space
	  ○ font selection and typesetting
	  ○ letter spacing
	  ○ line spacing
	  ○ justification
	  ○ use of colour and contrast
	  ○ input focus
	  ○ hover controls.
	• Be able to make appropriate design decisions considering:
	  ○ browser support
	  ○ target device/platform
	  ○ user characteristics
	  ○ available bandwidth
	  ○ style and branding
	  ○ accessibility
	  ○ user input method including:
	  – voice
	  – text
	  – touch screen
	  – mouse.
	(E5, M1, M4, M7, M8, M10, D1, D2, D6)

6.3	**Connect code to data sources as part of a software project**
	• Create data sources to support software solutions using a database
	• Connect to data sources using different connections
	  ○ application programming interface (API)
	  – types of requests and request methods
	  – endpoints
	  – retrieve and parse data
	  – display data
	  – API keys
	  ○ Java database connectivity (JDBC)
	  – core API
	  – driver manager
	  – connection statement
	  – prepared statement
	  – result set
	  – SQL queries
	  ○ open database connectivity (ODBC)
	  – application
	  – driver manager
	  – driver
	  – data source

	○ connection method – database name or data source – credentials – username/password – optional parameters ○ extracting data ○ storing data ○ updating data ○ deleting data ○ connect to network resources using tools within the development environment. • Be able to select and use data sources and connection methods appropriate to the context and market environment in which they are developing software. (E5, M4, M5, M6, M10, D1, D3, D4, D5, D6)
6.4	**Select and use deployment methods for a software project** • Use suitable deployment methods, such as: ○ local installation ○ network/server installation ○ mobile platforms ○ web-based platforms ○ cloud-based platforms ○ containerisation ○ container-scheduling platforms. (E5, M5, M6, M10, D1, D4, D6)

Content area 7: Testing a software solution

What underpinning knowledge do students need?	
7.1	**Understand, select and apply functional, non-functional and front-end testing** • Be able to select and carry out appropriate functional, non-functional and front-end testing relevant to the component or product being tested and the stage within the software development lifecycle: ○ functional testing: – unit testing – smoke testing – integration testing – system testing ○ non-functional testing: – availability testing – compatibility testing – configuration testing – load testing ○ front-end testing to check: – code/script performance and functionality – browser compatibility – operating system compatibility – cross-browser performance – formatting and rendering – loading times – responsiveness ○ security testing: – vulnerability scanning – static analysis – dynamic analysis – integration analysis. (E1, E4, E5, E6, M2, M4, M5, M6, M8, M10, D1, D2, D4, D6)
7.2	**Understand, select and apply testing techniques** • Selection of appropriate manual and automatic testing techniques (including use of AI tools) to ensure a software solution meets identified requirements: ○ acceptance testing ○ alpha testing ○ beta testing ○ closed box testing ○ open box testing/structural testing. (E1, E4, E5, E6, M2, M4, M5, M6, M8, M10, D1, D2, D4, D6)

7.3	**Select appropriate tests and test data to test the functionality of software**
	• Purpose of the identified test.
	• Test data:
	  ○ valid test data
	  ○ invalid test data
	  ○ valid extreme test data
	  ○ invalid extreme test data
	  ○ erroneous test data.
	• Pre-requisite to each test.
	• Expected test results.
	• Update the plan to include:
	  ○ actual results
	  ○ changes made
	  ○ re-tests/regression testing following changes.
	(E1, E4, E5, E6, M2, M4, M5, M6, M8, M10, D1, D2, D4, D6)

Content area 8: Change, maintain and support software

What underpinning knowledge do students need?	

8.1	**Understand the changing nature of digital products and the factors that drive change**
	• How business-driven development affects the types of maintenance to be performed:
	  ○ to prevent identified and foreseeable issues, such as:
	  – changes to regulatory requirements
	  – compatibility with new products or technology
	  – changes in business process
	  – release of a new product/service
	  ○ to correct unforeseen or previously unpreventable errors, such as:
	  – new vulnerabilities due to changes in other products or systems (zero day)
	  – targeted attack
	  – data corruption
	  – system failures
	  ○ iterative development of digital products to maintain relevance:
	  – review following user/client feedback
	  – developments in technology
	  – competition with other organisations
	  – need to improve efficiency
	  – need to future-proof products.
	(E1, E2, E3, E5, D3, D6)

8.2	**Understand the stages involved in the software change management process**
	• Identify issues/changes made during the feedback/review process.
	• Document developments and changes in a software project.
	• Communication with different audiences:
	  ○ technical
	  ○ non-technical.
	• Planning the changes required.
	• Scheduling the changes.
	• Regression testing.
	• Control and release of updated products:
	  ○ planned
	  ○ reactive.
	(E1, E2, E3, E5, M6, M10, D1, D3, D6)

8.3	**Understand how to maintain code as part of a larger team**
	• Understand and apply good coding principles when developing, adapting and maintaining code:
	  ○ separating code and use of modularity
	  ○ readability
	  ○ use of common/accepted coding conventions
	  ○ informative commenting/annotation within the code
	  ○ updating change logs and other documentation.
	(E1, E4, E5, E6, M2, M4, M5, M6, M8, M10, D1, D2, D3, D4, D6)

8.4	**Understand how to support software users**
	• Be able to communicate with technical and non-technical audiences using appropriate tone and levels of technical vocabulary through:
	  ○ face-to-face communication
	  ○ remote conferencing
	  ○ written:
	  – blogs
	  – formal reports
	  – technical documentation
	  – release notes
	  – user guides/help files
	  – FAQs
	  ○ visual and audio:
	  – user demonstrations
	  – screencast videos
	  – narration (recorded voice, text-to-speech)
	  ○ machine-readable application programming interface (API) contract.
	• Apply systematic processes to users and resolve issues:
	  ○ identify or replicate the issue
	  ○ investigate the possible cause of issues:
	  – user error
	  – system error
	  – application error
	  – security breach
	  ○ apply testing techniques to:
	  – identify errors in the system/code
	  – make changes to the system/code as required
	  – ensure error does not return
	  – no additional issues have been caused as a result of the changes made
	  ○ communicate how and when the issue was resolved to appropriate stakeholders
	  ○ document lessons learned.
	(E1, E4, E5, E6, M2, M4, M5, M6, M8, M10, D1, D2, D3, D4, D6)

Scheme of Assessment

There is a single synoptic assessment for this Occupational Specialism, which is an extended project. The synoptic element of the project is important to ensure students can demonstrate threshold competence and are able to evidence all the skills required by the Performance Outcomes.

The project consists of several activities grouped into four substantive tasks.

Each task is completed during a window set by Pearson, during which Providers schedule supervised assessment sessions. In some cases, tasks also include opportunities for unsupervised activities, where the requirements of the skills being assessed make this necessary.

For Task 1 Activity C students are expected to 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 Providers should use; however, the nature of the task would suggest the use of a natural language processing model such as ChatGPT or Google Gemini.

Occupational Specialism project – Digital Software Development
Internally assessed project: 50 hours 30 minutes **144 marks**
Performance Outcomes In this project students will: **PO1** – Analyse a problem to define requirements and acceptance criteria, aligned to user needs **PO2** – Design, implement and test software **PO3** – Change, maintain and support software **PO4** – Create solutions in a social and collaborative environment **PO5** – Discover, evaluate and apply reliable sources of knowledge **PO6** – Apply ethical principles and manage risks in line with legal and regulatory requirements when developing software

Occupational Specialism project – Digital Software Development

Assessment overview

There are four parts to the assessment:

- Task 1: Analysing a problem and designing a solution
- Task 2: Developing the solution
- Task 3a: Gathering feedback
- Task 3b Evaluating feedback

Students respond to a given scenario to complete a substantial project. They are assessed on their application of the skills listed for the Performance Outcomes.

Students are not assessed against specific 'knowledge' outcomes but are expected to draw on and apply related knowledge to ensure appropriate outcomes when applying the skills in response to an assessment scenario.

Students undertake the project under a combination of supervised and controlled conditions.

Internet access is permitted for all tasks except task 3b.

The assessment takes place over multiple sessions, up to a combined duration of 50.5 supervised hours.

The project outcomes consist of a portfolio of evidence submitted electronically.

This project is externally marked by Pearson.

Administration

Providers must follow the guidance in the following:

- General Administrative Support Guide
- Administration Support Guide for the specific Technical Qualification Occupational Specialism Project (if applicable).

These are located on the Training and Admin Support webpage.

Performance Outcome		Weighting	
		Raw marks	% of total marks
PO1	Analyse a problem to define requirements and acceptance criteria, aligned to user needs	24	16.7%
PO2	Design, implement and test software	57	39.6%
PO3	Change, maintain and support software	18	12.5%
PO4	Create solutions in a social and collaborative environment	9	6.3%
PO5	Discover, evaluate and apply reliable sources of knowledge	18	12.5%
PO6	Apply ethical principles and manage risks in line with legal and regulatory requirements when developing software	18	12.5%

Resources for the delivery of Occupational Specialism: Digital Software Development

Providers are required to have the following resources to deliver this OS:

- teachers with qualifications and/or experience in the digital sector
- a curriculum team with experience and knowledge that span the breadth of the qualification content.

	Resource required
General	• Software that supports Python 3.10 or later version and selected coding languages • Word processing and spreadsheet software (such as MS Office) • Data back-up solutions (for example, external hard drives, network storage, cloud file stores) • Data encryption software (for example, MS BitLocker) • Internet access • Printer
Specific	• IDE and debuggers appropriate to the chosen languages • Automated testing tools for: UI testing, performance testing, load/stress testing, compatibility testing • Access to communication, collaboration and data collection tools, including online code repositories (such as GitHub), forums (e.g. stack overflow), questionnaire/survey tools (e.g. Google Forms, SurveyMonkey), email • Access to AI code-generation resources • Access to online third-party digital content (e.g. video and graphics) • Screencasting and video editing software • Microphones and headphones.

5 Technical Qualification grading, T Level grading and results transfer

How the Technical Qualification is graded and awarded

Calculation of the Technical Qualification grade

The Technical Qualification components are awarded at the grade ranges below.

Component	Available grade range
Core (including Core examination/s and Employer Set Project)	A* – E and Unclassified
Occupational Specialism	Distinction, Merit, Pass and Unclassified

The Core uses an aggregation of points from each of the Core assessments to calculate the A* to E grade.

Students whose level of achievement for either component is below the minimum judged by Pearson to be of sufficient standard receive an Unclassified (U) result.

Awarding the components

Grade boundaries will be set for each component and/or sub-component (Core Examinations, Employer Set Project and Occupational Specialism) in each series they are offered through a process known as awarding. Awarding is used to set grade boundaries and ensure standards are maintained over time. This is important as we must ensure students have the same opportunity to achieve, regardless of the assessment opportunity.

Uniform Mark Scale

For the Core component, students' raw component and/or sub-component marks are converted to a Uniform Mark Scale (UMS). The UMS is used to convert students' 'raw' marks into uniform marks. This is done to benchmark outcomes from one series to another to account for any variety of difficulty in assessments. For example, a student who produces a response worthy of a C grade in the Employer Set Project in one series will receive the same uniform mark as a student achieving that same grade and level of performance in another series, regardless of their raw marks.

The maximum number of uniform marks available for each sub-component, and the uniform marks relating to each grade boundary, are fixed. These are shown below.

Grade	Core Exam	Core ESP	Core Overall
Maximum	**240**	**160**	**400**
A*	216 – 240	144 – 160	360 – 400
A	192 – 215	128 – 143	320 – 359
B	168 – 191	112 – 127	280 – 319
C	144 – 167	96 – 111	240 – 279
D	120 – 143	80 – 95	200 – 239
E	96 – 119	64 – 79	160 – 199
U	0 – 95	0 – 63	0 – 159

Where the Core component has two Core Exams, the results are combined before conversion to UMS.

Calculation of the T Level grade

The T Level grade look-up table shows the minimum thresholds the Department for Education uses for calculating the T Level grade.

Students must complete both components and achieve a minimum of a grade E in the Core and a Pass in the Occupational Specialism. In addition, they must successfully complete the other elements of the T Level, such as the industry placement.

Students who do not meet the minimum requirements will not be certificated.

Results transfer to Providers

Technical Qualification result days

Assessment series	Results day
Summer	August (Level 3 Results Day)
November	March (normally the third week − Level 3 Results Day)

Pearson issues the results directly to you and makes available:

- Scorecards: outlining the achievement in percentage terms against each Assessment Objective
- Results Plus: a service whereby achievement will be presented in an item-by-item format. This means Providers will be able to ascertain trends across and within cohorts, and clearly label the associated Assessment Objective
- Statement of Provisional Results: we will offer a provisional component result slip, clearly watermarked as a provisional component result.

As we are not required to issue Technical Qualification certificates, T Level certificates or T Level statements of achievement, we do not require you to complete any forms or processes to claim the Technical Qualification from Pearson.

T Level Results reporting

The Technical Qualification forms part of the T Level.

The Department for Education will issue T Level results on Level 3 results day in August.

The Department for Education will provide T Level certificates to students who successfully complete all elements of the T Level.

Appendix 1: General Competency Frameworks for T Levels

The General Competency Framework for T Levels articulates English, maths and digital competencies that students are required to develop over the course of the qualification. The tables below list the competencies from the framework that are relevant to the *T Level Technical Qualification in Digital Software Development.*

Competencies that can be developed in relation to a specification element of content are referenced in the column next to this content element in the Occupational Specialism. These competencies should be delivered through the content of this qualification and teachers should seek opportunities to allow students to develop the relevant skills to enable them to reach threshold competence in the specialism.

The English, maths and digital competencies are embedded in both the Core Component and the Occupational Specialism Component of the *T Level Technical Qualification in Digital Software Development*. This is so that students can demonstrate their knowledge and understanding of these skills over the course of the qualification.

General English competencies

Students should be supported to develop the English knowledge and skills needed in order to:

E1	Convey technical information to different audiences
E2	Present information and ideas
E3	Create texts for different purposes and audiences
E4	Summarise information/ideas
E5	Synthesise information
E6	Take part in/leading discussions

General maths competencies

Students should be supported to develop the maths knowledge and skills needed in order to:

M1	Measure with precision
M2	Estimate, calculate and spot errors
M3	Work with proportion
M4	Use rules and formulae
M5	Process data
M6	Understand data and risk
M7	Interpret and represent with mathematical diagrams
M8	Communicate using mathematics
M9	Cost a project
M10	Optimise work processes

General digital competencies

Students should be supported to develop the digital knowledge and skills needed in order to:

D1	Use digital technology and media effectively
D2	Design, create and edit documents and digital media
D3	Communicate and collaborate
D4	Process and analyse numerical data
D5	Be safe and responsible online
D6	Code and program

Command word taxonomy list

The following table shows the command words that will be used consistently in our assessments to ensure students are rewarded for demonstrating the necessary skills. The list below will not necessarily be used in every paper and is provided for guidance only.

Command word	Definition
Give/state/name	Provide a response (e.g. a feature, a characteristic, a use of or a justification for something).
Identify	Select the correct answer from the given context or stimulus.
Write	Write code using information from the given context or stimulus.
Describe	Provide responses that are linked in an appropriate logical order.
Explain	Requires identification of a point and linked justification of that point.
Explain with additional justification	Requires identification of a point, a linked justification and a third point, which is a further justification of the first justification.
Discuss	Consider the factors that apply in relation to a specific context. Give careful consideration to opposing aspects of an issue, situation or problem. A conclusion is not required.
Evaluate	Consider various aspects of a subject's qualities in relation to its context, such as strengths and weaknesses, advantages and disadvantages, pros and cons. Come to a judgement supported by evidence which will often be in the form of a conclusion.
Draw	Produce a diagram (e.g. an algorithm, a process flow) using information from a given context or scenario.
Complete	Provide the missing information for a diagram so that it is complete (contains all the necessary information).

Appendix 2: Flowchart symbols and Python commands

This appendix provides additional information about the digital technologies that students are expected to learn about within the core and occupational specialism content.

Students may choose to explore Python in greater detail than outlined in this appendix as one of their chosen languages for the Occupational Specialism.

This appendix does **not** replace the specification but should be used alongside the specification content to provide additional guidance and scope.

Sections of the specification that do not require additional expansion are not included.

Python commands

Input, output, type conversions

- input()
- print()
- int()
- str()
- float()
- bool()

Selection

- if
- elif
- else
- match case

Iteration

- while
- for

Functions

- def

Standard libraries

- import
- math
- math.floor()
- math.ceil()
- math.trunc()
- math.sqrt()
- math.pow()

- math.pi
- random
- random.random()
- random.randint()
- random.uniform()
- random.sample()

Built-in functions

- range()
- round()
- max()
- min()
- count()
- chr()
- ord()
- len()

String handling

- isupper()
- islower()
- upper()
- lower()
- isalpha()
- split()
- len()
- find()
- index()
- isalnum()
- isdigit()
- replace()
- strip()
- format()
- concatenation using ‘+’

Data structures (lists, arrays, dictionaries)

- index()
- append()
- insert()
- remove()
- count()
- pop()
- sort()
- in
- not in
- len()

Text files

- open()
- write()
- close()
- read()
- readline()
- readlines()
- line.strip()
- line.split()

Working with times and dates

- datetime.now()
- strftime()
- strptime()

Additional libraries and commands

For questions in the Employer Set Project, students will also be expected to have a working knowledge of these additional libraries:

pandas

Students should be able to use the *pandas* library to perform data analysis on a given data file (.csv). They should be able to:

- import data from a provided .csv file
- create and manipulate data frames which utilise all or part of the imported data (as necessary to meet given requirements)
- perform mathematical operations and statistical analysis on the contents of a data frame (or associated variable), such as:
  - identifying trends and patterns over time
  - calculating a total or average from a range of data
  - counting the number of occurrences of a specific item of data.

Matplotlib

Students should be able to use the *matplotlib* library to format and output data, in conjunction with the *pandas* library, as part of analysis on a given data file (.csv).

They should be able to:

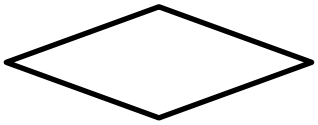
- select data and output to appropriate graphs to meet the requirements of a brief
- format graph outputs to ensure they are meaningful and easy to use (e.g. axis
- labels, colour schemes, legends etc.).

Flowchart symbols

Denotes the start and end of an algorithm

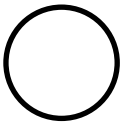
Denotes a process to be carried out

Denotes a sub-process



Denotes a decision to be made

Denotes input or output



Denotes a connection to part of a flowchart that cannot easily be linked using an unbroken flow arrow

Shows the logical flow of the program

Explore Pearson's T Levels offering at
https://qualifications.pearson.com/en/qualifications/t-levels.html