Highfield Level 4 End-Point Assessment for ST0116 Software Developer

End-Point Assessment Kit

Highfield Level 4 End-Point Assessment for ST0116 Software Developer

EPA Kit

Contents

Please click on the headings below to navigate to the associated section of the EPA Kit.

Versions:

ST0116 / v1.1 / AP06

SD v1.2

How to use this EPA Kit

Welcome to the Highfield End-Point Assessment Kit for the Software Developer apprenticeship standard.

Highfield is an independent end-point assessment organisation that has been approved to offer and carry out the independent end-point assessments for the Level 4 Software Developer apprenticeship standard. Highfield internally quality assures all end-point assessments in accordance with its IQA process, and additionally all end-point assessments are externally quality assured by the relevant EQA organisation.

The EPA Kit is designed to outline all you need to know about the end-point assessments for this standard and will also provide an overview of the on-programme delivery requirements. In addition, advice and guidance for trainers on how to prepare apprentices for the end-point assessment is included. The approaches suggested are not the only way in which an apprentice may be prepared for their assessments, but trainers may find them helpful as a starting point.

In this kit, you will find:

- an overview of the standard and any on-programme requirements
- a section focused on delivery, where the standard and assessment criteria are presented in a suggested format that is suitable for delivery
- guidance on how to prepare the apprentice for gateway
- detailed information on which part of the standard is assessed by which assessment method
- suggestions on how to prepare the apprentice for each part of the end-point assessment
- a section focused on the end-point assessment method where the assessment criteria are presented in a format suitable for carrying out 'mock' assessments

Introduction

Standard overview

This occupation is found across every sector, for example, financial services, computer gaming, retail, transport, and security and defence in organisations ranging from large multi-nationals, public sector bodies and government projects developing multi-billion-pound software solutions to support key projects to small consultancy firms designing bespoke software solutions for clients.

The broad purpose of the occupation is to understand a client's requirements as provided in the design specification and then build and test high-quality code solutions to deliver the best outcome.

Software developers are the creative minds behind computer programs. Some develop the applications that allow people to do specific tasks on a computer or another device. Others develop the underlying systems that run the devices or that control networks.

Organisations use software to ensure that their operations become ever more effective and robustly reduce the incidence of downtime by building quality tested software solutions to give a better service. For example, in commercial organisations this can give them a competitive advantage by being able to analyse significant amounts of data quickly and efficiently to provide the business with information and management systems. This can save time and help the business spot profit making opportunities. For public sector bodies, the right software solution can drive up performance and help target scarce resources more effectively and ensure that customer expectations are more likely to be met.

In their daily work, a software developer interacts with internal and external parties including users/customers (to understand their needs and test the software developed through user testing) and team members from a range of specialist fields including designers, developers, engineers, analysts and project/delivery managers (to ensure the effective implementation of software solutions). A developer will typically be working as part of a larger team, in which they will have responsibility for some of the straightforward elements of the overall project. The developer will need to be able to interpret design documentation and specifications. The customer requirements will typically be defined and agreed by more experienced or specialist members of the team, such as a business analyst or technical architect.

An employee in this occupation will be responsible for developing software solutions across the full software development life cycle from research and development, through continuous improvement, to product/service retirement. They may work both autonomously and as part of wider teams, typically reporting to a more senior member of their team.

On-programme requirements

Although learning, development and on-programme assessment is flexible, and the process is not prescribed, the following is the recommended baseline expectation for an apprentice to achieve full competence in line with the Software Developer apprenticeship standard.

The on-programme assessment approach will be agreed between the training provider and employer. The assessment will give an ongoing indication of an apprentice's performance against the final outcomes defined in the standard. The training provider will need to prepare the apprentice for the end-point assessment, including preparation for the interview and collation of the portfolio of evidence (such as a provision of recordings of professional discussions or workplace evidence).

The training programme leading to end-point assessment should cover the breadth and depth of the standard using suggested on-programme assessment methods that integrate the knowledge, skills and behaviour (KSB) components, and which ensure that the apprentice is sufficiently prepared to undertake the end-point assessment. Training, development and ongoing review activities should include:

- achievement of level 2 English and maths. If the apprentice began their apprenticeship training before their 19th birthday, they will still be subject to the mandatory requirement to study towards and achieve English and maths. The requirements for English and maths are optional for apprentices aged 19+ at the start of their apprenticeship training.
- completion of a portfolio through which the apprentice gathers evidence of their progress.

Portfolio of evidence

The apprentice must compile a portfolio of evidence during their time on-programme that is mapped against the knowledge, skills and behaviours assessed in the professional discussion underpinned by a portfolio of evidence.

There should be at least 1 piece of evidence relating to each of the knowledge, skills and behaviours mapped to the professional discussion. Evidence may be used to demonstrate more than 1 KSB as a qualitative as opposed to quantitative approach is suggested.

It is expected that there will typically be **10 pieces of evidence** in total.

The portfolio should contain evidence of work projects completed, such as:

- written accounts of activities that have been completed
- photographic evidence and work products
- work instructions
- safety documentation
- technical reports
- drawings

- company policies and procedures as appropriate to the activities

Progress review documentation, witness testimonies, and feedback from colleagues and/or clients should also be included.

Reflective accounts and self-assessments **must not** be included as evidence.

The portfolio **must** be accompanied by a Portfolio Matrix. This can be downloaded from our website. The Portfolio Matrix must be fully completed including a declaration by the employer and the apprentice to confirm that the portfolio is valid and attributable to the apprentice.

The portfolio of evidence must be submitted to Highfield at gateway. It is not directly assessed but underpins the professional discussion.

In cases where the apprentice is working in a confidential environment, the employer may insist that the assessor reviews the portfolio at the employer's premises.

Work-based project summary

The apprentice will scope out and provide a summary of what the project will cover including a stakeholder specification. The apprentice will submit this to Highfield at gateway. This should demonstrate that the work-based project will provide sufficient opportunity for the apprentice to cover the knowledge, skills and behaviours (KSBs) mapped to this assessment method.

The summary is not assessed and will typically be **no longer** than **500 words**.

The summary needs to outline the project plan, including high level implementation steps and associated timeframes as well as the date the work-based project has to be submitted to the independent assessor, taking into account the deadlines stipulated within this assessment method. The project may be based on any of the following:

- a customer or stakeholder specification requiring the apprentice to respond to any of the following:
  - a specific problem
  - a recurring issue
  - an idea/opportunity

Highfield Assessment will sign-off the work-based project in consultation with the employer within **2 weeks** of gateway.

A **project summary document** should be completed by the apprentice which includes a declaration from the employer that the project provides sufficient scope for the assigned knowledge, skills and behaviours (KSBs) to be assessed. This form is available to download from the Highfield Assessment website. It will also need to be indicated as completed on the Gateway Readiness Report (available from Highfield Assessment website).

Once the project summary has been approved, after gateway, the apprentice will expand this project summary into a project report, which **will** be assessed.

Use of artificial intelligence (AI) in the EPA

Where AI has been used as part of the apprentice's day-to-day work and forms part of a project report, presentation, or artefact, it should be referenced as such within the work. AI must not be used to produce the report or portfolio.

Where AI has been used as part of a portfolio that underpins an interview or professional discussion or any other assessment method, it should be fully referenced within the portfolio.

Readiness for end-point assessment

For an apprentice to be ready for the end-point assessments:

- the apprentice must have achieved level 2 English and maths. The requirements for English and maths are mandatory for all apprentices aged between 16-18 at the start of their apprenticeship training. The requirements for English and maths are optional for apprentices aged 19+ at the start of their apprenticeship training.
- the apprentice must have gathered a **portfolio of evidence** against the required elements to be put forward to be used as the basis for the professional discussion.
- the apprentice must submit the **work-based project summary**, typically no longer than 500 words, outlining the project plan, research requirements and an overview of time frames, taking into account the deadlines stipulated within the case study.
- the line manager (employer) must be confident that the apprentice has developed all the knowledge, skills and behaviours defined in the apprenticeship standard and that the apprentice is competent in performing their role. To ensure this, the apprentice must attend a formal meeting with their employer to complete the Gateway Readiness Report.
- the apprentice and the employer should then engage with Highfield to agree a plan and schedule for each assessment activity to ensure all components can be completed within a 6-month end-assessment window. Further information about the gateway process is covered later in this kit.

If you have any queries regarding the gateway requirements, please contact your EPA customer engagement manager at Highfield Assessment.

Order of end-point assessments

There is no stipulated order of assessment methods. This will be discussed with the apprentice, training provider and/or employer with our scheduling team when scheduling the assessments to ensure that the learner is provided with the best opportunity to attempt the assessment.

Click here to return to contents

The Highfield approach

This section describes the approach Highfield has adopted in the development of this end-point assessment in terms of its interpretation of the requirements of the end-point assessment plan and other relevant documents.

Documents used in developing this end-point assessment

Standard (2021)

https://skillsengland.education.gov.uk/apprenticeships/st0116-v1-1

End-point assessment plan (ST0116/v1.1/AP06)

https://skillsengland.education.gov.uk/media/cd4amafd/st0116_software-developer_l4_ap-for-publication_270521.pdf

Specific considerations

The assessment plan does not stipulate what is required for a resit/retake of the work-based project with questioning. Highfield will allow apprentices to rework their project report rather than completing a different project.

Click here to return to contents

Gateway

How to prepare for gateway

After apprentices have completed their on-programme learning, they should be ready to pass through 'gateway' to their end-point assessment.

Gateway is a meeting that should be arranged between the apprentice, their employer and training provider to determine that the apprentice is ready to undertake their end-point assessment. The apprentice should prepare for this meeting by bringing along work-based evidence, including:

- customer feedback
- recordings
- manager statements
- witness statements

As well as evidence from others, such as:

- mid and end-of-year performance reviews
- feedback to show how they have met the apprenticeship standards while on-programme

In advance of gateway, apprentices will need to have completed the following. The requirements for English and maths listed below are mandatory for all apprentices aged between 16-18 at the start of their apprenticeship training. The requirements for English and maths listed below are optional for apprentices aged 19+ at the start of their apprenticeship training.

- Achieved level 2 English
- Achieved level 2 maths
- Submitted a suitable portfolio of evidence to be used as the basis for the professional discussion (see the Portfolio Matrix)
- Submitted a work-based project summary typically no longer than 500 words

Therefore, apprentices should be advised by employers and providers to gather this evidence and undertake these qualifications during their on-programme training. It is recommended that employers and providers complete regular checks and reviews of this evidence to ensure the apprentice is progressing and achieving the standards before the formal gateway meeting is arranged.

The gateway meeting

The gateway meeting should last around an hour and must be completed on or after the apprenticeship on-programme end date. It should be attended by the apprentice and the relevant people who have worked with the apprentice on-programme, such as the line manager/employer or mentor, the on-programme trainer/training provider and/or a senior manager (as appropriate to the business).

During the meeting, the apprentice, employer and training provider will discuss the apprentice's progress to date and confirm if the apprentice has met the full criteria of the apprenticeship standard during their on-programme training. The **Gateway Readiness Report** should be used to log the outcomes of the meeting and agreed by all 3 parties. This report is available to download from the Highfield Assessment website.

The report should then be submitted to Highfield to initiate the end-point assessment process. If you require any support completing the Gateway Readiness Report, please contact your EPA customer engagement manager at Highfield Assessment.

Please note: a copy of the standard should be available to all attendees during the gateway meeting.

Reasonable adjustments and special considerations
Highfield Assessment has measures in place for apprentices who require additional support. Please refer to the Highfield Assessment Reasonable Adjustments policy for further information/guidance.

ID requirements
Highfield Assessment will need to ensure that the person undertaking an assessment is indeed the person they are claiming to be. All employers are, therefore, required to ensure that each apprentice has their identification with them on the day of the assessment so the end-point assessor can check.

Highfield Assessment will accept the following as proof of an apprentice's identity:

- a valid passport (any nationality)
- a signed UK photocard driving licence
- a valid warrant card issued by HM forces or the police
- another photographic ID card, such as an employee ID card or travel card

Click here to return to contents

The Software Developer apprenticeship standard

Below are the knowledge, skills and behaviours (KSBs) from the standard and related assessment criteria from the assessment plan. On-programme learning will be based upon the KSBs and the associated assessment criteria are used to assess and grade the apprentice within each assessment method.

Work-based project with questioning		
Knowledge	**Skills**	**Behaviours**
K2 Roles and responsibilities within the software development lifecycle (who is responsible for what). **K6 How teams work effectively to produce software** and how to contribute appropriately. **K9** Principles of **algorithms, logic and data structures relevant to software development** for example: Arrays, Stacks, Queues, Linked Lists, Trees, Graphs, Hash Tables, Sorting Algorithms, Searching Algorithms, Critical sections and race conditions. **K11 Software designs and functional/technical specifications**.	**S1 Create logical and maintainable code**. **S4 Test code and analyse results** to correct errors found using unit testing. **S6 Identify and create test scenarios**. **S7** Apply **structured techniques to problem solving**, can debug code and can understand the structure of programmes to identify and resolve issues. **S10 Build, manage and deploy code into the relevant environment**. **S11** Apply an appropriate software development approach according to the **relevant paradigm** (for example object oriented, event driven or procedural). **S12 Follow software designs and functional/technical specifications**. **S16 Apply algorithms, logic and data structures**.	**B2** Applies logical thinking. For example, uses **clear and valid reasoning** when making decisions related to undertaking work instructions. **B3** Maintains a **productive, professional, and secure working environment**.

Pass criteria	Distinction criteria
WP1 Explains the **roles and responsibilities** of all people working within the software development lifecycle, and how they relate to the project. (K2) **WP2** Outlines **how teams work effectively to produce software** and how to contribute appropriately. (K6) **WP3** Outlines and applies the rationale and use of algorithms, logic and data structures. (K9, S16) **WP4** Reviews methods of software design with reference to functional/technical specifications and applies a justified approach to software development. (K11, S11, S12) **WP5** Creates logical and maintainable code to deliver project outcomes, explaining their choice of approach. (S1) **WP6** Analyses unit testing results and reviews the outcomes correcting errors. (S4) **WP7** Identifies and creates test scenarios which satisfy the project specification. (S6) **WP8** Applies **structured techniques to problem solving** to identify and resolve issues and debug basic flaws in code. (S7) **WP9** Reviews and justifies their contribution to building, managing and deploying code into the relevant environment in accordance with the project specification. (S10) **WP10** Establishes a logical thinking approach to areas of work which require valid reasoning and/or justified decision making. (B2)	***WP12*** *Compare and contrast the requirements of a software development team, and how they would ensure that each member (including themselves) were able to make a contribution. (K6)* ***WP13*** *Evaluates the advantages and disadvantages of different coding and programming techniques to* ***create logical and maintainable code****. (S1)* ***WP14*** *Analyses the software to identify and debug complex issues using a fix that provides a permanent solution. (S7)* ***WP15*** *Evaluates different software development approaches in order justifying the best alignment with a given paradigm. (for example, object oriented, event driven or procedural) (S11)*

WP11 Describes how they have maintained a productive, professional and secure working environment throughout the project activity. (B3)	

Amplification and guidance

- **Roles and responsibilities** could include:
  - the business analyst who gathers and documents requirements, translates them into functional specifications and ensures the alignment between stakeholders and the development team
  - the project manager who manages project plans, schedules, risks, resources and deliverables to keep the project on track with timelines and budgets
  - the software architect who designs system architecture, makes key decisions on frameworks and tools and ensures that the design meets requirements
  - the developers who write, test, review and maintain code throughout the software life cycle

- **How teams work effectively to produce software** could include:
  - using agile methodologies, such as Scrum or Kanban, with tools such as Jira for task tracking and continuous improvement
  - utilising tools such as Slack for real-time communication and regular meetings to maintain alignment
  - implementing strategies like mediation to resolve conflicts and maintain productivity
  - clearly defining roles to minimise overlap and ensure smooth collaboration

- **Algorithms, logic and data structures relevant to software development** could include:
  - understanding common algorithms such as sorting, searching and graph algorithms. Understanding the time and space complexity of these algorithms using Big O notation.
  - choosing appropriate data structures, such as Arrays and Trees, based on the problem at hand for efficient development.
  - using logical reasoning to break down problems and apply structured programing practices, such as loops and conditionals.
  - choosing among synchronous and asynchronous programing mechanisms to ensure data concurrency or race conditions.

- **Software designs and functional/technical specifications** could include:

  - creating and designing documents, including Unified Modelling Language (UML) diagrams and flowcharts
  - writing clear, functional and technical specifications with user stories and acceptance criteria
  - detailing how the system will be implemented, covering:
    - architecture
    - component design
    - data models
    - application programming interface (API)
    - security
    - deployment plans
  - following industry standards and best practices, such as design patterns and compliance with security benchmarks

- **Create logical and maintainable code** by:
  - following community coding standards for consistency and readability
  - writing modular code with single-responsibility functions for easier maintenance
  - using descriptive names for variables and functions
  - providing clear documentation for each module and function
  - adopting SOLID (single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle and dependency inversion principle), test driven development and error handling plus logging practices

- **Test code and analyse results** could include:
  - using frameworks, such as Junit or Mocha for isolated unit tests
  - aiming for high coverage to ensure critical paths are tested
  - integrating tests into continuous integration/continuous delivery (CI/CD) pipelines to prevent bugs in code changes
  - reviewing and fixing bugs based on test results

- **Identify and create test scenarios** could include:
    - understanding requirements to outline various test cases, including edge cases and covering both functional and non-functional tests
    - documenting scenarios clearly, using tools such as JIRA or TestRail, specifying steps and expected outcomes, and success criteria
    - unit, integration, system, user acceptance testing (UAT), performance and security testing
    - utilising automation tools such as, Selenium, Postman or LoadRunner

- **Structured techniques to problem solving** could include:
    - using methods such as, root cause analysis (RCA), the 5 Whys or fishbone diagrams to identify and resolve issues
    - leveraging integrated development environment (IDE) debugging tools, such as, Visual Studio Code, to trace and fix code errors
    - understanding and applying coding patterns, such as object-oriented programming (OOP) or modular programing, for effective problem-solving
    - recording solutions in a knowledge base or similar tool for team reference

- **Build, manage and deploy code into the relevant environment** could include:
    - CI/CD pipelines to automate build, test and deployment processes using tools such as Jenkins
    - Using Git for version controls with best practices, such as branching strategies
    - Setting up deployment pipelines that manage code integration, deployment, testing, approval gates and rollback mechanisms
    - Implementing strategies like Blue-Green Deployments to minimise risk
    - Managing configurations and dependencies across environments

- **Relevant paradigm** could include:
    - understanding and applying paradigm, such as object-oriented procedural (OOP) or event-driven programing based on project needs
    - using OOP principles, such as inheritance and polymorphism, for complex tasks, event-driven for interactive applications and procedural for linear tasks
    - implementing with languages, such as Python, JavaScript or C#

- **Follow software designs and functional/technical specifications** could include:
    - translating high-level software designs into functional code and ensuring that the implementation closely follows the specifications in the design documents
    - communicating regularly with stakeholders to ensure the software meets their needs and using iterative development and progress demos
    - ensuring codes meet the performance, security and regulatory requirements

- **Apply algorithms, logic and data structures** could include:
    - applying common algorithms to optimise performance
    - using appropriate structures, such as Arrays, Trees or Hash Tables based on task requirements
    - employing conditionals, loops and recursion while addressing issues
    - utilising IDE tools for visualising data flow

- **Clear and valid reasoning** could include:
    - using logical reasoning to select appropriate algorithms or data structures
    - breaking down complex problems into manageable parts and applying systematic solutions, including effective debugging
    - providing clear, evidence-based reasoning for development decisions, such as choosing a scalable architecture
    - identifying potential challenges in advance and proper planning for future risk and conflict handling

- **Productive, professional, and secure working environment** could include:
    - maintaining professionalism in all interactions, including punctuality, meeting deadlines and respecting colleagues
    - maximising productivity using time management tools and agile methods
    - following security best practices, using strong passwords, keeping software updated and adhering to data protection protocols

Professional discussion underpinned by portfolio		
Knowledge	**Skills**	**Behaviours**
K1 All **stages of the software development lifecycle** (what each stage contains, including the inputs and outputs). **K3** The **roles and responsibilities of the project lifecycle** within your organisation, and your role. **K4** How best to communicate using the **different communication methods** and how to adapt appropriately to different audiences. **K5** The similarities and differences between **different software development methodologies**, such as agile and waterfall. **K7 Software design approaches and patterns**, to identify reusable off-the-shelf solutions to commonly occurring problems. **K8 Organisational policies and procedures** relating to the tasks being undertaken, and when to follow them. For example, the storage and treatment of GDPR sensitive data. **K10** Principles and uses of **relational and non-relational databases**. **K12** Software testing **frameworks and methodologies**.	**S2** Develop **effective user interfaces**. **S3 Link code to data sets**. **S5** Conduct a **range of test types**, such as Integration, System, User Acceptance, Non-Functional, Performance and Security testing. **S8** Create **simple software designs** to effectively communicate understanding of the program. **S9 Create analysis artefacts**, such as use cases and/or user stories. **S13** Follow **testing frameworks and methodologies**. **S14** Follow **company, team or client approaches** to continuous integration, version and source control. **S15 Communicate software solutions and ideas to technical and non-technical stakeholders**. **S17** Interpret and implement a given design whist **remaining compliant with security and maintainability requirements**.	**B1** Works independently and takes responsibility. For example, has a disciplined and responsible approach to risk, and stays motivated and committed when **facing challenges**. **B4 Works collaboratively** with a wide range of people in different roles, internally and externally to the team, with a positive attitude to inclusion & diversity. **B5 Acts with integrity** with respect to ethical, legal and regulatory ensuring the protection of personal data, safety and security. **B6 Shows initiative and takes responsibility** for solving problems within their own remit, being resourceful when faced with a problem to solve. **B7** Communicates effectively in a variety of situations to both a **technical and non-technical audience**. **B8** Shows curiosity to the **business context** in which the solution will be used, displaying an inquisitive approach to solving the problem. This includes the curiosity to explore new opportunities, and techniques; the tenacity to improve methods and maximise performance of

		the solution; and creativity in their approach to solutions. **B9** Committed to **continued professional development**.

Pass criteria	Distinction criteria
PD1 Describes all **stages of the software development lifecycle**. (K1) **PD2** Describes the **roles and responsibilities of the project lifecycle** within their organisation, and their role. (K3) **PD3** Describes methods of communicating with all stakeholders that is determined by the audience and/or their level of technical knowledge. (K4, S15) **PD4** Describes the similarities and differences between **different software development methodologies**, such as agile and waterfall. (K5) **PD5** Suggests and applies different **software design approaches and patterns**, to identify reusable solutions to commonly occurring problems (include Bespoke or off-the-shelf). (K7) **PD6** Explains the relevance of **organisational policies and procedures** relating to the tasks being undertaken, and when to follow them including how they have followed **company, team or client approaches** to continuous integration, version, and source control. (K8, S14) **PD7** Applies the principles and uses of **relational and non-relational databases** to software development tasks. (K10) **PD8** Describes basic software testing **frameworks and methodologies**. (K12) **PD9** Explains, their own approach to development of user interfaces. (S2)	***PD22*** *Compares and contrasts the different types of communication used for technical and non-technical audiences and the benefits of these types of communication methods. (K4, S15, B7)* ***PD23*** *Evaluates and recommends approaches to using reusable solutions to common problems. (K7)* ***PD24*** *Evaluates the use of various software testing **frameworks and methodologies** and justifies their choice. (K12)*

PD10 Explains, how they have linked code to data sets. (S3) **PD11** Illustrates how to conduct test types, such as Integration, System, User Acceptance, Non-Functional, Performance and Security testing including how they have followed **testing frameworks and methodologies**. (S5, S13) **PD12** Creates **simple software designs** to communicate understanding of the programme to stakeholders and users of the programme. (S8) **PD13** Creates analysis artefacts, such as use cases and/or user stories to enable effective delivery of software activities. (S9) **PD14** Explains, how they have interpreted and implemented a given design whilst **remaining compliant with security and maintainability requirements**. (S17) **PD15** Describes, how they have operated independently to complete tasks to given deadlines which reflect the level of responsibility assigned to them by the organisation'. (B1) **PD16** Illustrates how they have worked collaboratively with people in different roles, internally and externally, which show a positive attitude to inclusion & diversity. (B4) **PD17** Explains how they have established an approach in the workplace which reflects integrity with respect to ethical, legal, and regulatory matters and ensures the protection of personal data, safety and security. (B5) **PD18** Illustrates their approach to meeting unexpected minor changes at work and outlines their approach to delivering within their remit using their initiative. (B6)	

PD19 Explains how they have communicated effectively in a variety of situations to both a **technical and non-technical audience**. (B7) **PD20** Illustrates how they have responded to the **business context** with curiosity to explore new opportunities and techniques with tenacity to improve solution performance, establishing an approach to methods and solutions which reflects a determination to succeed. (B8) **PD21** Explains how they reflect on their **continued professional development** and act independently to seek out new opportunities. (B9)	

Amplification and guidance

- **Stages of the software development lifecycle** could include:
  - gathering requirements
  - design
  - implementation
  - testing
  - deployment
  - maintenance

- **Roles and responsibilities of the project lifecycle** could include:
  - understanding the key roles in the project life cycle, such as project manager, developer, tester and business analyst
  - knowing what each role is responsible for, including planning, execution, testing and delivery
  - recognising specific duties and contributions within the project and ensuring alignment with the team's objectives

- **Different communication methods** could include:
  - emails
  - meetings
  - presentations

    - reports
    - instant messaging

- **Different software development methodologies** could include:
    - Agile/Scrum:
        - high flexibility and change management
        - continuous feedback
    - Waterfall:
        - limited flexibility
        - sequential phases
        - clear milestones
    - DevOps:
        - collaborates and bridges gap in development and operations
        - supports Agile and Kanban
        - automated processes
        - continuous integration and delivery
    - Lean/Kanban:
        - prioritises efficiency
        - reduces waste
        - optimises workflow
    - Spiral:
        - combines iterative development with risk management

- **Software design approaches and patterns** could include:
    - singleton, which ensures a class has only 1 instance and provides a global point of access
    - factory, which creates objects without specifying the exact class of object that will be created

- observer, which defines a one-to-many dependency, which allows multiple objects to be notified of changes to another object
- strategy, which encapsulates algorithms within a class and allows them to be interchangeable
- object oriented design, which organises software around objects rather than functions, emphasising reusability and modularity
- component-based design, which builds software from pre-existing components

- **Organisational policies and procedures** could include:
  - General Data Protection Regulations (GDPR)
  - Code of Conduct
  - Data Protection Act

- **Relational and non-relational databases** could include:
  - SQL/NoSQL
  - Atomicity, consistency, isolation and durability (ACID) properties versus basically available, soft state and eventual consistency (BASE) properties
  - Structured data
  - Normalisation
  - Flexible schemes
  - Document stores

- **Frameworks and methodologies** could include:
  - Junit
  - Selenium
  - TestNG
  - Unit testing
  - Integration testing
  - System testing

    - User acceptance testing (UAT)
    - Performance testing

- **Effective user interfaces** could include:
    - usability principles, which design intuitive and consistent interfaces, ensuring easy navigation across the application
    - responsive design, which creates interfaces that adapt to different devices and screen sizes
    - integrating accessibility standards to make the user interface (UI) usable for all users
    - visual design, which uses colour schemes, typography and layout to create visually appealing and user-friendly interfaces
    - user-centred design, which involves users in the design process via feedback and testing
    - developing prototyping to visualise and test interfaces before full implementation

- **Link code to data sets** could include:
    - database connections, data files access, API access, AWS Cloud and AZURE Blob storage
    - data retrieval
    - data manipulation

- **Range of test types** could include:
    - integration testing
    - system testing
    - user acceptance testing (UAT)
    - non-functional testing
    - performance testing
    - automated testing

- **Simple software designs** could include:
    - Using diagrams and sketches to convey design ideas
    - Pseudocode describes design logic in plain language

  - Prototypes to validate design ideas

- **Create analysis artefacts** could include:
  - developing detailed user cases that outline how users will interact with the system
  - improving communication among stakeholders and supports project planning and execution
  - creating a software requirement specification document, which could include:
    - stakeholder analysis
    - a traceability matrix
    - a scoping document
    - a wireframe
    - a mock up
    - user stories
    - case diagrams
    - data models

- **Testing frameworks and methodologies** could include:
  - implementing testing frameworks like Junit, Pytest or Selenium
  - verifying individual components with unit tests to ensure each part functions correctly
  - confirming that different modules or services work together smoothly
  - validating that the entire application is fully integrated to meet specific requirements
  - user acceptance testing (UAT)
  - test driven development (TDD)

- **Company, team or client approaches** could include:
  - branching strategies
  - automated builds

  - version control best practices
  - source control
  - documentation
  - access control
  - backup and restore strategies

- **Communicate software solutions and ideas to technical and non-technical stakeholders** could include:
  - tailoring communication based on the audience
  - simplifying complex concepts using easy-to-understand language and analogies
  - providing tailored documentation
  - collaborations
  - presentations
  - prototypes

- **Remaining compliant with security and maintainability requirements** could include:
  - secure coding practices
  - maintainability
  - adhering to organisational standards
  - conducting thorough testing
  - providing clear documentation to support maintenance

- **Facing challenges** could include:
  - self-motivation
  - decision-making
  - problem ownership
  - accountability

    - perseverance

- **Works collaboratively** could include:
    - team collaboration
    - inclusion
    - diversity
    - positive attitude
    - conflict resolution

- **Acts with integrity** could include:
    - securely handles sensitive data, adhering to regulations such as General Data Protection Regulations (GDPR)
    - maintains an ethical conduct by being honest and transparent in all interactions
    - ensures that all work complies with relevant laws and industry regulations

- **Shows initiative and takes responsibility** could include:
    - identifies and addresses issues early
    - takes responsibility for tasks and outcomes
    - uses available resources to solve problems effectively
    - improves processes by suggesting new tools or automating tasks
    - seeks ways to continually improve

- **Technical and non-technical audience** could include:
    - knowing and understanding the audience
    - using technical details for developers and simplified language for non-technical stakeholders
    - engages with stakeholders by using two-way communication
    - crafts clear emails, reports and documents to suit all audiences

- adjusts content and style of presentation tools based on the audience’s expertise

- **Business context** could include:
  - understanding business goals
  - inquisitively assessing the business’s needs
  - having business impact awareness

- **Continued professional development** could include:
  - lifelong learning
  - skill enhancement
  - professional networking
  - mentorship
  - self-reflection

Click here to return to contents

Assessment summary

The end-point assessment for the Software Developer apprenticeship standard is made up of 2 assessment methods:

1. A **4,500-word** project report **(+/-10%)** to be completed over a **9-week** period, and **60-minute (+10%)** questioning. A minimum of **12 questions** will be asked.
2. A **60-minute (+10%)** professional discussion underpinned by portfolio. A minimum of **12 open questions** will be asked.

As an employer/training provider, you should agree a plan and schedule with the apprentice to ensure all assessment components can be completed effectively.

Each component of the end-point assessment will be assessed against the appropriate criteria laid out in this kit, which will be used to determine a grade for each individual. The grade will be determined using the combined grades.

Work-based project with questioning

The work-based project with questioning is weighted equally with all other assessment methods. Apprentices will be marked against the pass and distinction criteria outlined in this kit.

- To achieve a **pass**, apprentices must achieve all of the pass criteria
- To achieve a **distinction**, apprentices must achieve all of the pass criteria **and** all of the distinction criteria
- **Unsuccessful** apprentices will not have achieved all of the pass criteria

The work-based project with questioning should be conducted in a suitable location such as an employer's or training provider's premises.

Professional discussion underpinned by portfolio

The professional discussion is weighted equally with all other assessment methods. Apprentices will be marked against the pass and distinction criteria outlined in this kit.

- To achieve a **pass**, apprentices must achieve all of the pass criteria
- To achieve a **distinction**, apprentices must achieve all of the pass criteria **and** all of the distinction criteria
- **Unsuccessful** apprentices will not have achieved all of the pass criteria

The professional discussion may be conducted using technology such as video link, as long as fair assessment conditions can be maintained.

Grading

The apprenticeship includes pass, merit and distinction grades, with the final grade based on the apprentice's combined performance in each assessment method.

To achieve a pass, the apprentice is required to pass each of the 2 assessment methods.

To achieve a merit, the apprentice is required to pass 1 assessment method and gain a distinction in the other.

To achieve a distinction, the apprentice is required to have achieved all of the pass criteria and all of the distinction criteria in each of the 2 assessment methods.

The overall grade for the apprentice is determined using the matrix below:

Work-based project with questioning	**Professional discussion underpinned by portfolio**	**Overall grade awarded**
Fail any of the 2 assessment methods		**Fail**
Pass	Pass	**Pass**
Pass	Distinction	**Merit**
Distinction	Pass	**Merit**
Distinction	Distinction	**Distinction**

Retake and resit information

Apprentices who fail one or more assessment method will be offered the opportunity to take a re-sit or a re-take. A resit does not require further learning, whereas a retake does. The apprentice's employer will need to agree that either a resit or retake is an appropriate course of action. If a resit is chosen, please call the Highfield scheduling team to arrange the resit. If a retake is chosen, the apprentice will require a period of further learning and will need to complete a retake checklist. Once this is completed, please call the Highfield scheduling team to arrange the retake.

A resit is typically taken within 2 months of the EPA outcome notification. The timescale for a retake will be dependent on how much retraining is required but is typically taken within 4 months of the EPA outcome notification.

When undertaking a resit or retake, the assessment method(s) will need to be reattempted in full, regardless of any individual assessment criteria that were passed on any prior attempt. The EPA Report will contain feedback on areas for development and resit or retake guidance.

Apprentices will not need to complete a different project where a resit or retake is required but will need to revise their existing project report. Apprentices will be asked different questions in the case of a resit or retake.

Any EPA component resit/retake must be taken within a 6-month period, otherwise, the entire EPA must be retaken in full, unless in the opinion of Highfield exceptional circumstances apply outside the control of the apprentice or their employer. Apprentices should have a supportive action plan to prepare for the resit/retake.

Resits and retakes are not offered to apprentices wishing to move from pass to merit/distinction or merit to distinction.

Where any assessment method has to be resat or retaken, the apprentice will be awarded a maximum grade of distinction.

Click here to return to contents

Assessing the work-based project with questioning

This end-point assessment method consists of 2 components:

- project report
- questioning

Component 1: project report

The project report is compiled after the apprentice has gone through gateway. The apprentice will conduct their project and submit a project report to Highfield after a maximum of 9 weeks following the sign off of the project's summary and stakeholder specification. The project will typically take 7 weeks and the report will take a further 2 weeks to write.

The employer will ensure that the apprentice has sufficient time and the necessary resources, within this period, to plan and undertake the project.

The project report should be in the form of an electronic report comprising narrative, one or more coded artefacts, and visual infographics as necessary. Coded artefacts may consist of numerous lines of code in one language or be made up of multiple code languages.

As a minimum all project reports must include the following sections:

- an introduction
- the scope of the project (including key performance indicators)
- a project plan
- consideration of legislation, regulation, industry and organisational policies, procedures and requirements
- analysis and problem-solving in response to challenges within the project
- research and findings
- project outcomes explained by referencing artefacts within the appendices to convey the software solution and design of the software development outputs
- recommendations and conclusions
- an explanation of how the stages of the software development life cycle which are involved have been evidenced, for example:
    - planning
    - analysis
    - design
    - implementation/build
    - test
    - deploy
    - maintain

The report has a maximum word limit of **4,500 words** (+/- 10%) and appendices, references and diagrams will **not** be included in this total. The appendices must include artefacts comprising examples of relevant coding undertaken and visual infographics conveying the software solution and design of the software development outputs sufficient to demonstrate the KSBs assigned to this assessment method.
The apprentice may work as part of a team, which could include technical internal or external support, however, the report will be the apprentice's own work and will be reflective of their own role and contribution.

The apprentice must produce and include a mapping of KSBs in an appendix, showing how the report evidences the KSBs mapped to this assessment method. The project mapping document is available to download from the Highfield Assessment website.

End-point assessors will only mark reports up to **4,950 words**, at which point, assessors will stop marking and only credit the criteria covered to that point. Reports that fall short of the word count will be marked in full, against all criteria. The assessor will review and assess the project report in advance of the questioning.

Regard needs to be given to confidentiality and security requirements to ensure that proprietary commercially sensitive coding data is not compromised. Similarly, where required Highfield must ensure that appropriate security clearance procedures and policies are agreed at the gateway. Where redaction of key information is necessary it should not materially compromise the ability of the independent assessor to reliably assess the project report.

The report **must** be uploaded in PDF format and comprised of narrative, 1 or more coded artefacts and visual infographics as necessary. The report must be accompanied by the **written submission sheet**, which is available to download from the Highfield Assessment website. On the written submission sheet, the apprentice and their employer must verify that the submitted report is the apprentice's own work and must map how it evidences the relevant knowledge, skills and behaviours for this assessment method, as outlined in this kit.

Component 2: questioning

The questioning will be conducted using the project report as a basis.

The questioning will take place in a quiet room free from distractions and influence, such as the employer's premises or online using video conferencing.

The questioning must last for **60 minutes**. The independent assessor has the discretion to increase the time of the questioning by up to 10% to allow the apprentice to complete their last answer. The independent assessor will ask a minimum of **12 questions**.

Before the assessment

Employers/training providers should:

- give the apprentice time to work on their project and report during the end-point assessment window
- ensure the apprentice knows the date, time and location of the assessment
- ensure the apprentice knows which software developer criteria will be assessed (outlined on the following pages)
- encourage the apprentice to reflect on their experience and learning on-programme to understand what is required to meet the standard and identify real-life examples
- be prepared to provide clarification to the apprentice, and signpost them to relevant parts of their on-programme experience as preparation for this assessment

Grading the work-based project with questioning

Apprentices will be marked against the pass and distinction criteria included in the tables on the following pages (under ‘work-based project with questioning criteria’).

- To achieve a **pass**, apprentices must meet all of the pass criteria
- To achieve a **distinction**, apprentices must meet all of the pass **and** distinction criteria
- Unsuccessful apprentices will not have achieved all of the pass criteria

Work-based project with questioning mock assessment

It is suggested that a mock assessment is carried out by the apprentice in advance of the end-point assessment with the training provider/employer giving feedback on any areas for improvement. It is the employer/training provider’s responsibility to prepare apprentices for their end-point assessment and Highfield recommend that the apprentice experiences a mock questioning in preparation for the real thing. The most appropriate form of mock assessment will depend on the apprentice’s setting and the resources available at the time.

When planning a mock assessment, the employer/training provider should include the following elements:

- mock questioning should be **60 minutes**.
- consider a recording of the mock assessment and allow it to be played back to other apprentices, especially if it is not practicable for the employer/training provider to carry out a separate mock assessment with each apprentice.

- ensure that the apprentice's performance is assessed by a competent trainer/assessor, and that feedback is shared with the apprentice to complete the learning experience.
- mock assessment sheets are available to download from the Highfield Assessment website and may be used for this purpose.
- use 12 structured, 'open' questions that do not lead the apprentice but allow them to express their knowledge in a calm and comfortable manner. Some examples of this may include the following:
    - how do the roles and responsibilities of the people working within the software development life cycle relate to your project?
    - explain how you contribute to effective safe working practices within your team when working on your project.
    - explain the rationale behind your use of algorithms and data structures within your project.
    - review the methods you used in software design with reference to the specifications you had within your project.
    - provide an example of when you have analysed unit testing results and have had to correct errors as an outcome within your project.
    - explain your approach when creating logical and maintainable code within your project.
    - provide an example of a test scenario you created for your project specification.
    - explain your contribution to building and managing code in the relevant environment for your project.
    - describe how you maintained a productive working environment while remaining professional and secure when working on your project.
    - explain a structured technique you used to debug basic flaw in the codes within your project.
    - evaluate the pros and cons of different coding techniques that you have used within your project.
    - provide an example of when you analysed software to identify complex issues and the permanent solution you found within your project.

Work-based project with questioning criteria

Throughout the work-based project with questioning, the assessor will review the apprentice's competence in the criteria outlined below.

Apprentices should prepare for the work-based project with questioning by considering how the criteria can be met and reflecting on their past experiences.

Work-based project with questioning
To pass, the following must be evidenced.
WP1 Explains the roles and responsibilities of all people working within the software development lifecycle, and how they relate to the project. (K2) **WP2** Outlines how teams work effectively to produce software and how to contribute appropriately. (K6) **WP3** Outlines and applies the rationale and use of algorithms, logic and data structures. (K9, S16) **WP4** Reviews methods of software design with reference to functional/technical specifications and applies a justified approach to software development. (K11, S11, S12) **WP5** Creates logical and maintainable code to deliver project outcomes, explaining their choice of approach. (S1) **WP6** Analyses unit testing results and reviews the outcomes correcting errors. (S4) **WP7** Identifies and creates test scenarios which satisfy the project specification. (S6) **WP8** Applies structured techniques to problem solving to identify and resolve issues and debug basic flaws in code. (S7) **WP9** Reviews and justifies their contribution to building, managing and deploying code into the relevant environment in accordance with the project specification. (S10) **WP10** Establishes a logical thinking approach to areas of work which require valid reasoning and/or justified decision making. (B2) **WP11** Describes how they have maintained a productive, professional and secure working environment throughout the project activity. (B3)
To gain a distinction, the following must be evidenced.
WP12 *Compare and contrast the requirements of a software development team, and how they would ensure that each member (including themselves) were able to make a contribution. (K6)* ***WP13*** *Evaluates the advantages and disadvantages of different coding and programming techniques to create logical and maintainable code. (S1)* ***WP14*** *Analyses the software to identify and debug complex issues using a fix that provides a permanent solution. (S7)* ***WP15*** *Evaluates different software development approaches in order justifying the best alignment with a given paradigm. (for example, object oriented, event driven or procedural) (S11)*

Click here to return to contents

Assessing the professional discussion underpinned by portfolio

In the professional discussion underpinned by a portfolio of evidence, the assessor and the apprentice will have a formal two-way conversation. It will consist of the independent assessor asking the apprentice questions to assess their competence against the relevant criteria outlined in this kit.

The portfolio is submitted to Highfield at gateway. A copy of the portfolio can be retained by the apprentice and brought by them to the professional discussion. The portfolio will be used by the apprentice to refer to exemplify a point.

The professional discussion will be scheduled at least 2 weeks after gateway. It will take place in a suitable environment and can be conducted by video conferencing. It will last for **60 minutes**. The independent assessor can increase the time of the professional discussion by up to 10% to allow the apprentice to complete their last answer.

The assessor will ask **at least 12 open questions**. Follow up questions will then be used to draw out further evidence or where clarification is required.

Before the assessment

Employers/training providers should:

- ensure the apprentice knows the date, time and location of the assessment
- ensure the apprentice knows which criteria will be assessed (outlined on the following pages)
- encourage the apprentice to reflect on their experience and learning on-programme to understand what is required to meet the standard
- be prepared to provide clarification to the apprentice, and signpost them to relevant parts of their on-programme experience as preparation for this assessment

Grading the professional discussion underpinned by portfolio

Apprentices will be marked against the pass and distinction criteria included in the tables on the following pages (under 'professional discussion underpinned by portfolio criteria').

- To achieve a **pass**, apprentices must achieve all of the pass criteria
- To achieve a **distinction**, apprentices must achieve all of the pass criteria **and** all of the distinction criteria
- **Unsuccessful** apprentices will have not achieved all of the pass criteria

Professional discussion underpinned by portfolio mock assessment

It is the employer/training provider's responsibility to prepare apprentices for their end-point assessment. Highfield recommends that the apprentice experiences a mock professional discussion underpinned by portfolio in preparation for the real thing. The most appropriate form of mock professional discussion underpinned by a portfolio of evidence will depend on the apprentice's setting and the resources available at the time.

In designing a mock assessment, the employer/training provider should include the following elements in its planning:

- the mock professional discussion underpinned by portfolio should take place in a suitable location.
- a **60-minute** time slot should be available to complete the professional discussion underpinned by portfolio, if it is intended to be a complete professional discussion covering all relevant standards. However, this time may be split up to allow for progressive learning.
- consider a video or audio recording of the mock professional discussion underpinned by portfolio and allow it to be available to other apprentices, especially if it is not practicable for the employer/training provider to carry out a separate mock assessment with each apprentice.
- ensure that the apprentice's performance is assessed by a competent trainer/assessor, and that feedback is shared with the apprentice to complete the learning experience. Mock assessment sheets are available to download from the Highfield Assessment website and may be used for this purpose.
- use 12 structured, 'open' questions that do not lead the apprentice but allows them to express their knowledge and experience in a calm and comfortable manner. For example:
    - describe a method of communication you use when relaying information to stakeholders.
    - describe some similarities between different software development methodologies.
    - provide an example of a suggestion you have made for different software design approaches and patterns, to identify reusable solutions.
    - explain when you have had to follow your organisation's policies and procedures relating to tasks you have undertaken.
    - describe a testing methodology you have used in practice.
    - provide an example of a software design you created to enable understanding of the programme to stakeholders.

- provide an example of a time when you needed to interpret a given design while ensuring you remained compliant with security requirements.
- provide an example of a time when you have had to work independently to complete tasks to a deadline.
- describe an approach you have established in the workplace that reflects your integrity with respect to ethical, legal and regulatory matters.
- explain how you reflect on your continuous professional development.
- compare the different types of communication methods you have had to use in practice for technical and non-technical audiences and explain the benefits.
- explain the use of various software testing frameworks and your choice in which you think is the best to aid you in any given project.

Professional discussion underpinned by portfolio criteria

Throughout the **60-minute** professional discussion underpinned by portfolio, the assessor will review the apprentice's competence in the criteria outlined below.

Apprentices should prepare for the professional discussion underpinned by portfolio by considering how the criteria can be met.

Professional discussion underpinned by portfolio
To pass, the following must be evidenced.
PD1 Describes all stages of the software development lifecycle. (K1)
PD2 Describes the roles and responsibilities of the project lifecycle within their organisation, and their role. (K3)
PD3 Describes methods of communicating with all stakeholders that is determined by the audience and/or their level of technical knowledge. (K4, S15)
PD4 Describes the similarities and differences between different software development methodologies, such as agile and waterfall. (K5)
PD5 Suggests and applies different software design approaches and patterns, to identify reusable solutions to commonly occurring problems (include Bespoke or off-the-shelf). (K7)
PD6 Explains the relevance of organisational policies and procedures relating to the tasks being undertaken, and when to follow them including how they have followed company, team or client approaches to continuous integration, version, and source control. (K8, S14)
PD7 Applies the principles and uses of relational and non-relational databases to software development tasks. (K10)
PD8 Describes basic software testing frameworks and methodologies. (K12)
PD9 Explains, their own approach to development of user interfaces. (S2)
PD10 Explains, how they have linked code to data sets. (S3)
PD11 Illustrates how to conduct test types, such as Integration, System, User Acceptance, Non-Functional, Performance and Security testing including how they have followed testing frameworks and methodologies. (S5, S13)
PD12 Creates simple software designs to communicate understanding of the programme to stakeholders and users of the programme. (S8)
PD13 Creates analysis artefacts, such as use cases and/or user stories to enable effective delivery of software activities. (S9)
PD14 Explains, how they have interpreted and implemented a given design whilst remaining compliant with security and maintainability requirements. (S17)
PD15 Describes, how they have operated independently to complete tasks to given deadlines which reflect the level of responsibility assigned to them by the organisation'. (B1)

PD16 Illustrates how they have worked collaboratively with people in different roles, internally and externally, which show a positive attitude to inclusion & diversity. (B4) **PD17** Explains how they have established an approach in the workplace which reflects integrity with respect to ethical, legal, and regulatory matters and ensures the protection of personal data, safety and security. (B5) **PD18** Illustrates their approach to meeting unexpected minor changes at work and outlines their approach to delivering within their remit using their initiative. (B6) **PD19** Explains how they have communicated effectively in a variety of situations to both a technical and non-technical audience. (B7) **PD20** Illustrates how they have responded to the business context with curiosity to explore new opportunities and techniques with tenacity to improve solution performance, establishing an approach to methods and solutions which reflects a determination to succeed. (B8) **PD21** Explains how they reflect on their continued professional development and act independently to seek out new opportunities. (B9)
To gain a distinction, the following must be evidenced.
PD22 *Compares and contrasts the different types of communication used for technical and non-technical audiences and the benefits of these types of communication methods. (K4, S15, B7)* ***PD23*** *Evaluates and recommends approaches to using reusable solutions to common problems. (K7)* ***PD24*** *Evaluates the use of various software testing frameworks and methodologies and justifies their choice. (K12)*

Click here to return to contents