Next Generation Higher National Unit Grading Pack

Higher National Diploma Software Development

Qualification code: GV22 48

This qualification is available in a restricted delivery model from academic session 2025

This grading pack provides information about the process of grading the Higher National Diploma (HND) Software Development. It is for lecturers and assessors, and contains all the mandatory information you need to grade the HND.

You must read it alongside the Educator Guide.

Published: October 2025 (version 1.0)

Contents

Approach to grading

Grading in Next Generation: Higher National (NextGen: HN) Qualifications produces a valid and reliable record of a learner's level of achievement across the breadth of the qualification content.

As well as grading the whole qualification, you assess individual units on a pass or fail basis. Each unit has evidence requirements that learners must achieve before you can consider them for whole-qualification grading.

Whole-qualification grade outcomes

Learners who pass NextGen: HN Qualifications receive one of the following grade outcomes for the qualification as a whole:

- Achieved with Distinction
- Achieved with Merit
- Achieved

To determine a learner's whole-qualification grade, you use the grading matrix to assess and judge their performance across the key aspects of the HND. You must align your judgements with the following whole-qualification grade descriptors.

Whole-qualification grade descriptors

Achieved with Distinction

The learner has achieved an excellent standard across the course content, going significantly beyond meeting the qualification requirements. They showed a comprehensive knowledge and understanding of course concepts and principles, and consistently used them to apply skills to complete high-quality work. They engaged significantly with the process of developing their meta-skills in the context of their HN Qualification.

Achieved with Merit

The learner has achieved a very good standard across the course content, going beyond meeting the qualification requirements. They showed a very good knowledge and understanding of course concepts and principles, and consistently used them to apply skills to complete work of a standard above that expected for an Achieved grade. They actively engaged with the process of developing their meta-skills in the context of their HN Qualification.

Achieved

The learner has achieved a good standard across the course content, credibly meeting the qualification requirements. They showed a good knowledge and understanding of course concepts and principles, and used them to apply skills to complete work of the required standard. They engaged with the process of developing their meta-skills in the context of their HN Qualification.

What the whole-qualification grade descriptors do and how they are used

The whole-qualification grade descriptors outline the skills, knowledge and understanding a learner needs to show across the whole qualification to achieve that specific grade. They align with the Scottish Credit and Qualifications Framework (SCQF) level descriptors.

NextGen: HND qualifications are at SCQF level 8. Learners who complete a NextGen: HND can:

- convey an insightful understanding of the subject's core theories, concepts and principles, along with its scope and defining features
- apply skills, knowledge and understanding of the subject in relevant practical and professional contexts, showing some specialist knowledge and using a range of relevant techniques and materials
- describe and explain significant topical issues and specific areas of interest

- exercise autonomy and initiative in carrying out activities, and have developed their professional practice and behaviours relevant to the context of the qualification
- formulate and critically evaluate evidence-based responses to issues in the context of the subject area, appropriately applying research and academic processes

Please use this information, as well as the whole-qualification grade descriptors, to help you understand the standard at which learners should be assessed and graded.

Higher education institutes (HEIs) can use the grade descriptors to set admissions requirements, and employers can use them to help make decisions during a recruitment process.

SQA's quality assurance teams use the grade descriptors and the grading matrix to ensure that grades awarded in a particular NextGen: HN Qualification are at a consistent national standard, regardless of the setting in which they are achieved.

Successful learners receive their grade, along with the grade descriptor, on their certificate.

Using the grading matrix

You must use the grading matrix to judge the learner's whole-qualification grade. You can use the grading matrix at any time, but you only make a whole-qualification grading judgement when you are confident the learner has met all the evidence requirements of all the required units.

The criteria in the grading matrix reflect the knowledge, skills and qualities HEIs and employers can expect of a learner who has completed the qualification. These criteria align with the overall purpose of the qualification, and remain the same for its duration.

Each criterion has sector-specific descriptors of a typical learner's performance standard, aligned to the whole-qualification grade outcomes of Achieved, Achieved with Merit and Achieved with Distinction. These descriptors describe the standard a learner of that whole-qualification grade is expected to show.

The guidance accompanying each criterion can include, but is not limited to, information on:

- relevant types of assessment that may produce useful or meaningful evidence for judging that criterion
- mapping to content that is particularly relevant to that criterion
- mapping to meta-skills

This guidance may be updated over time.

When you make your final grading judgement, you must use a 'best fit' approach based on the learner's achievement across the grading matrix. This may be straightforward — for example, if the learner's evidence shows a consistent standard across the grading matrix criteria. If it is not straightforward, you must make a 'best fit' judgement — for example, if a learner shows a mix of standards across the grading matrix criteria, with no clear pattern. The criteria may not always have equal value. You can decide some are more important to the final grade than others.

Meta-skills

Meta-skills are a key part of NextGen: HN Qualifications and learners can develop them throughout the qualification. A learner's engagement with developing their own meta-skills contributes to their qualification grade. You do not assess or grade competence or progress in individual meta-skills — for example, by judging the quality of a learner's feeling or creativity. Instead, you look at the process of development learners go through. This means learners need to provide evidence of planning, developing and reflecting on their meta-skills.

If qualification content also contributes to meta-skills development, it contributes to a learner's whole-qualification grading through the grading matrix approach.

Learning for Sustainability

Learning for Sustainability does not contribute to a learner's whole qualification grade.

The exception is where Learning for Sustainability content is part of the qualification content. In which case the Learning for Sustainability content will contribute to a learner's whole-qualification grade through the grading matrix.

Grading matrix

Criterion 1 descriptors

Criterion 1	Achieved	Merit	Distinction
Demonstrate knowledge of concepts relating to software development	The learner: • provides a basic explanation of object-oriented design (OOD) and object-oriented programming (OOP) concepts • provides a basic description of programming techniques, such as decomposition, abstraction and modularity • demonstrates adequate knowledge of the syntax, semantics and constructs of a programming language	The learner: • provides a clear explanation of OOD and OOP concepts and outlines their benefits over other designs • provides a clear description of programming techniques such as decomposition, abstraction and modularity • demonstrates sound knowledge of the syntax, semantics and constructs of a programming language	The learner: • provides a detailed and clear explanation of OOD and OOP concepts and fully describes their benefits over other designs • provides a detailed and clear description of programming techniques such as decomposition, abstraction and modularity and explains their benefits • demonstrates sound knowledge of the syntax, semantics and constructs of more than one programming language and describes the use case of each language

Criterion 1	Achieved	Merit	Distinction
Demonstrate knowledge of concepts relating to software development (continued)	The learner: • demonstrates adequate knowledge of software development frameworks and methods • demonstrates adequate knowledge of programming standards, such as code style and security • demonstrates adequate knowledge of algorithms and their use cases • selects and uses appropriate data structures in creating software solutions • provides an adequate explanation of the goals and principles of test-driven development • provides an adequate explanation of version control and its benefits to software development	The learner: • demonstrates sound knowledge of software development frameworks and methods and compares the benefits of more than one software development method • demonstrates a sound understanding of programming standards such as code style, security and quality • demonstrates sound knowledge of algorithms by outlining how they work and their use cases • selects, justifies and uses appropriate data structures in creating software solutions • provides a clear explanation of the goals and principles of test-driven development	The learner: • demonstrates a clear and detailed understanding of a software development method and compares and contrasts the risks and benefits of more than one software development method • demonstrates a clear and detailed understanding of programming standards such as code style, security and quality, and explains their benefits • demonstrates detailed knowledge of algorithms by describing how they work, their use cases and their complexity measure

Criterion 1	Achieved	Merit	Distinction
Demonstrate knowledge of concepts relating to software development (continued)		The learner: • provides a clear explanation of version control and its benefits to software development	The learner: • selects, justifies and efficiently uses highly appropriate data structures in creating software solutions • provides a detailed clear explanation of the goals and principles of test-driven development and compares levels of testing • provides a detailed and clear explanation of version control, the risks of not doing it, and its benefits to software development

Criterion 1 guidance

This criterion requires learners to demonstrate their understanding of foundational programming concepts. The rubrics benchmark learners' ability to articulate, analyse, and apply their conceptual understanding through description, discussion, justification, and critical evaluation of the programming fundamentals they need to develop software solutions.

The criterion reflects the requirement for methodical and systematic approaches to software development through modern and traditional models for managing the development cycle. This competence can be evidenced in the following units:

Software Development (J7D9 48)

- Coding a software application using OOP concepts provides evidence of OOD and OOP concepts and their benefits over other designs.
- Using UML to visualise software design provides evidence of abstraction and modularity in software design.
- Using standard object libraries and modular coding provides evidence of programming constructs, decomposition, and modularity.
- Using appropriate data structures and algorithms provides evidence of selecting, justifying and using data structures, and understanding algorithm use cases.
- Creating and executing test plans provides evidence of understanding the goals and principles of test-driven development.
- Using version control tools provides evidence of understanding of version control and its benefits.

Software Engineering Methods (J7EF 48)

- Creating and justifying use of architectural and design patterns provides evidence of OOD and OOP design benefits.
- Describing and comparing software development lifecycle models provides evidence of comparing the benefits of different software development methods.
- Creating UML models of system components provides evidence of abstraction and modular system design.
- Applying structured testing strategies provides evidence of understanding the principles of test-driven development.

Professional Practice in Software Development (J7EE 48)

- Creating a working software solution using a full project lifecycle provides evidence of knowledge of the syntax, semantics, and constructs of a programming language.
- Producing and maintaining code with version control provides evidence of understanding version control and its benefits.
- Designing, developing, testing and documenting a team software project provides evidence of knowledge of development frameworks and methods.
- Reflecting on individual contributions and standards provides evidence of understanding of code quality, security, and professional standards.

Code Security (J7EB 48)

- Coding an application using secure coding practices provides evidence of understanding programming standards, including security and quality.
- Using version control tools for secure software development provides evidence of understanding version control and its benefits.
- Creating and applying a threat model and mitigation testing provides evidence of testing principles and quality control aligned with secure development.

Programming for Data (J7EG 48)

- Writing modular, reusable, well-documented code provides evidence of sound knowledge of syntax, semantics, code quality and standards.

- Selecting and implementing appropriate data structures and performing algorithmic data cleaning provide evidence of selecting and using appropriate data structures, and understanding algorithms.
- Testing and debugging data processing programs provides evidence of understanding the goals and principles of test-driven development.

Event-Driven Programming (J7EC 48)

- Designing and coding a modular, event-driven application provides evidence of understanding programming constructs, syntax, and modularity.
- Using exception handling and parameter passing in app logic provides evidence of understanding of programming constructs and security practices.
- Creating and applying a test plan provides evidence of understanding of test-driven development principles.

Database Design Fundamentals (J8FK 47)

- Demonstrating understanding of data types, normalisation, data modelling, and integrity provides evidence of abstraction and modularity, and proper structuring of information.
- Applying or implementing security measures to protect a database provides evidence of competence in understanding security and quality standards.
- Use of structured query language (SQL) and understanding of relational concepts shows understanding of syntax and semantics, and logic flow in programming.

DevOps Principles in Practice (J897 48)

- Detailed coverage of version control (Git), repositories, and branching strategies helps to provide evidence of DevOps principles.
- Knowledge and implementation of software development methods, including comparison with Waterfall, contributes to evidence of comparing frameworks and understanding benefits.
- Emphasis on test automation, continuous integration and continuous deployment (CI/CD), and infrastructure as code (IaC) provides evidence of understanding of test-driven development and quality assurance principles.
- Use of modern modular tools, containerisation, automation, and iteration provides evidence of understanding of principles of modularity and modern software development standards.

Database Design and Development (J7DV 48)

- In-depth use of SQL, query construction, joins, transactions, and user-defined functions evidence advanced knowledge of syntax, semantics, and logic constructs.
- Working with both relational and not only SQL (NoSQL) databases provides evidence of selection and justification of different data structures.
- Developing entity relationship diagrams (ERDs), normalisation, and referential integrity provides evidence of abstraction, decomposition, and modular data design.

Criterion 2 descriptors

Criterion 2	Achieved	Merit	Distinction
Design and develop software	The learner: • creates a software requirements specification that adequately identifies the problem to be solved • selects and implements an appropriate methodology to guide the development process • creates a software design document that describes the architecture of the solution and its data structures • writes source code that correctly implements a software design and demonstrates good coding practices • selects and uses software development tools for coding and debugging	The learner: • creates a software requirements specification that identifies the problem to be solved and specifies the requirements for the solution • selects, justifies and implements an appropriate methodology to guide the development process • creates a software design document that clearly describes the architecture of the solution and its data structures • writes source code that correctly and efficiently implements a software design and demonstrates competence in coding practice	The learner: • creates a software requirements specification that identifies the problem to be solved, specifies the requirements for the solution and identifies the constraints on the solution • selects, justifies and demonstrates competence in implementing a highly appropriate methodology to guide the development process • creates a detailed, clear software design document that fully describes the architecture of the solution, its data structures and the user interface

Criterion 2	Achieved	Merit	Distinction
Design and develop software (continued)	The learner: • creates a project plan and meets some milestones in its execution	The learner: • selects, justifies and efficiently uses software development tools for coding and debugging • creates a clear project plan and adheres to it, meeting most milestones	The learner: • writes source code that correctly and efficiently implements a software design and demonstrates a high level of competence in coding practice • selects, justifies and efficiently uses an appropriate range of software development tools for coding and debugging • creates a clear and detailed project plan and successfully executes it, with all milestones met

Criterion 2 guidance

This criterion reflects the software development activities and processes necessary for the establishment of the client (end-user) requirements, documenting them and developing use cases to guide solution development. The requirements must be validated. This competence can be evidenced in the following units:

Software Development (J7D9 48)

- Creating a user requirements specification and clarifying assumptions provide evidence of creating a software requirements specification that identifies the problem and specifies requirements.
- Applying agile methodologies and documenting the development lifecycle provide evidence of selecting, justifying and implementing an appropriate methodology.
- Using UML diagrams and rewriting a software specification provide evidence of creating a software design document describing architecture and data structures.
- Writing secure, modular, maintainable code with standard libraries provides evidence of writing source code that implements the design and demonstrates competence in coding.
- Using an IDE and version control to develop and debug code provides evidence of efficient use of software development tools for coding and debugging.
- Managing time and meeting project milestones provides evidence of creating a clear project plan and meeting milestones.

Software Engineering Methods (J7EF 48)

- Discovering and documenting system requirements and use cases provides evidence of creating a software requirements specification.
- Applying and justifying lifecycle models such as Waterfall, Agile, and Scrum provides evidence of selecting and justifying a methodology.
- Designing and documenting UML models and architectural patterns provides evidence of creating a software design document.
- Illustrating and applying software testing and quality assurance provides evidence of competence in coding practice and validation of implementation.

Professional Practice in Software Development (J7EE 48)

- Creating a detailed project plan with a timeline, diary and evidence of project meetings provides evidence of creating a clear project plan and adhering to it.
- Using a range of tools for planning, design, coding and testing provides evidence of selecting and using software development tools.
- Producing individual and team documentation for requirements, design, testing and code provides evidence of creating requirements and design documentation.
- Coding and testing a working application from a design provides evidence of writing source code that correctly implements a software design.

Code Security (J7EB 48)

- Creating a secure software design from a scenario provides evidence of creating a software design document that describes architecture and security-related structures.
- Writing secure code and applying threat mitigation provide evidence of writing source code that demonstrates competence in secure coding practice.
- Using development tools to prevent vulnerabilities and manage code provides evidence of using development tools for secure coding and debugging.

Application Development for Web (J7E1 48)

- Building a secure full-stack web app from a design brief provides evidence of writing source code that implements a design and shows coding competence.
- Using version control and collaboration tools to manage code provides evidence of efficient use of development tools.
- Working from a given design specification to build and deploy a solution provides evidence of following a software design document.

Event-Driven Programming (J7EC 48)

- Creating a requirements specification for an event-driven application provides evidence of creating a software requirements specification.
- Implementing a design using an event-driven programming framework provides evidence of writing source code that implements a software design.
- Using modular code and test planning provide evidence of competent coding practice and tool use for testing and debugging.

Programming for Data (J7EG 48)

- Selecting and applying data structures and tools for efficient coding provides evidence of writing efficient source code using appropriate tools.
- Producing documentation and debugging tested programs provide evidence of competent use of tools for coding and debugging.

Database Design Fundamentals (J8FK 47)

- Creating entity-relationship diagrams, relational models, and normalised tables helps to evidence production of software and data structure design documentation.
- Creating secure and functional database schemas from user need helps to evidence identifying data-related requirements.

DevOps Principles in Practice (J897 48)

- Using development methodologies, including traditional (Waterfall) versus DevOps, helps to evidence methodology selection and justification.
- Configuring tools including Git, CI/CD pipelines, containers, automation and version control help to evidence tool selection for coding, deployment and debugging.
- Demonstrating understanding of build, test and release processes, iterative improvement and feedback helps to evidence process planning and milestone delivery.

Database Design and Development (J7DV 48)

- SQL programming, including functions, transactions and joins helps to evidence coding skills within a defined domain (SQL).

Criterion 3 descriptors

Criterion 3	Achieved	Merit	Distinction
Test, deploy and document software	The learner: • creates and implements a test plan that confirms that the solution satisfies the software requirement specification • creates and executes a plan that outlines the steps to deploy the software to a target system • demonstrates security practices through authentication and authorisation • creates technical and user documentation to an adequate standard	The learner: • creates, justifies and implements a test plan that confirms that the solution satisfies the software requirement specification • creates, documents and executes a clear plan for deployment of the software to a target system • demonstrates good security practices to protect data through code inspection, encryption, authentication and authorisation • creates technical and user documentation that is clear, concise and easy to understand	The learner: • creates, justifies and implements a comprehensive test plan that verifies that the solution correctly satisfies the software requirement specification • creates a detailed, clear plan for deployment that includes instructions for installing and configuring the software on a target system and executes it • demonstrates comprehensive security practices to protect data through code inspection and analysis, encryption, authentication and authorisation • creates technical and user documentation that is comprehensive, clear and easy to understand

Criterion 3 guidance

This criterion reflects the essential technical, academic, and professional competences required for effective and responsible software development. This competence can be evidenced in the following units:

Software Development (J7D9 48)

- Creating and executing a test plan with documentation of test logs and errors provides evidence of creating, justifying and implementing a test plan to confirm the solution meets the specification.
- Providing evidence of software deployment and executables provides evidence of creating, documenting and executing a deployment plan.
- Producing technical documentation and an application user guide provides evidence of creating technical and user documentation that is clear and concise.
- Using secure coding practices and documenting test runs indirectly supports demonstrating good security practices through inspection and testing.

Professional Practice in Software Development (J7EE 48)

- Creating a test plan and reporting on test outcomes for a software application provides evidence of creating and implementing a test plan aligned with the requirements specification.
- Deploying a software application and producing user and technical documentation provide evidence of executing a deployment plan and producing clear documentation.
- Including ethical and security considerations in reflection supports evidence of demonstrating good security practices, including awareness of protection needs.

Code Security (J7EB 48)

- Performing security testing and applying mitigation solutions provide evidence of implementing a test plan with a focus on data protection and security.
- Using encryption, authentication, and secure coding provides evidence of demonstrating good security practices through encryption and authorisation.
- Creating documentation including threat models and security-focused design provides evidence of clear technical documentation relating to security and code inspection.

Application Development for Web (J7E1 48)

- Carrying out and documenting stringent testing of front-end and back-end provides evidence of creating and executing a test plan to validate specification compliance.
- Deploying a full-stack web app to a cloud platform provides evidence of executing a deployment plan for a target environment.
- Ensuring code meets secure development standards provides evidence of good security practices, including authentication and secure coding.
- Documenting results of testing and providing user documentation provide evidence of producing clear, concise technical and user documentation.

Event-Driven Programming (J7EC 48)

- Developing a test plan and documenting test results provide evidence of creating and implementing a test plan.

- Deploying a front-end app and evaluating user feedback provide evidence of executing a deployment plan and assessing outcomes.
- Including accessibility and security in UX evaluation supports evidence of security practices through awareness and testing.
- Preparing technical and user documentation provides evidence of producing clear documentation for users and developers.

Programming for Data (J7EG 48)

- Testing and debugging programs and revising code accordingly provides evidence of implementing and refining a test plan.
- Producing technical and user documentation for the program provides evidence of creating clear and understandable documentation.
- Including secure handling of data in program development provides evidence of good security practices through code inspection and data protection.

Database Design Fundamentals (J8FK 47)

- Database security operations, including authentication, provide evidence of security practices (authentication and authorisation).
- Explaining database structure and use helps to evidence production of technical documentation.

DevOps Principles in Practice (J897 48)

- Using CI/CD pipelines and automated build and test tools provides evidence of test planning and execution in a DevOps context.

- Deploying applications using containers, IaC, and repo-based automation provides evidence of deployment planning and execution.
- Monitoring, verification, and telemetry provide evidence of validation and iteration post-deployment.
- Producing documentation (for example for infrastructure and tools) provides evidence of producing technical documentation.

Criterion 4 descriptors

Criterion 4	Achieved	Merit	Distinction
Collaborate and communicate in a team context	The learner: • communicates adequately with team members, using appropriate channels, tools, and language • collaborates with team members by giving and receiving feedback, and contributing to solutions • contributes actively to the team's tasks and decisions, by sharing ideas, knowledge, and skills • reflects on their own and their team's performance, strengths and weaknesses, and identifies areas for improvement	The learner: • communicates effectively and respectfully with team members, using appropriate channels, tools, and language • collaborates creatively with team members by giving, receiving feedback and generating solutions • contributes actively and constructively to the team's tasks, goals, and decisions, freely sharing ideas, knowledge, and skills • reflects on their own and their team's performance, strengths, and weaknesses, and identifies actions for improvement	The learner: • communicates very effectively and respectfully with their team members, using appropriate channels, tools, and language • collaborates creatively and critically with team members by giving and receiving feedback, resolving conflicts, and generating solutions • demonstrates leadership in contributing actively and constructively to the team's tasks, goals, and decisions, by sharing ideas, knowledge, and skills • reflects critically on their own and their team's performance, strengths, and weaknesses, and identifies and initiates actions for improvement

Criterion 4 guidance

This criterion reflects the professional attitudes and behaviours that we expect of a software developer, including a commitment to adhering to best practice, planning and managing work schedules, and seeking continuous improvement. It also relates to software development being a team process, requiring collaboration and communication skills. We expect collaboration between individuals, professionals and groups, but you should also consider other forms of collaboration. This competence can be evidenced in the following units:

Professional Practice in Software Development (J7EE 48)

- Participating in a team-based project, with each learner taking a lead role in at least one stage, provides evidence of collaborating creatively and contributing actively to team tasks and goals.
- Producing a project diary, meeting evidence, and recordings of project meetings provides evidence of communicating effectively and respectfully using appropriate tools and channels.
- Presenting a software solution as a team to a client audience provides evidence of sharing ideas and communicating respectfully in a team context.
- Writing an individual evaluation of personal contribution and team experience provides evidence of reflecting on team and individual strengths and identifying actions for improvement.

Software Engineering Methods (J7EF 48)

- Gathering requirements from client consultation and document use cases provides evidence of communicating effectively with others in a professional context.

- Designing solutions based on team-derived inputs and analysis provides evidence of collaborating to generate and agree solutions.
- Performing validation and verification tasks collaboratively supports contributing constructively to team decisions.

Application Development for Web (J7E1 48)

- Using collaborative version control and code-sharing tools provides evidence of communicating effectively using appropriate tools and sharing ideas and knowledge.
- Working to a client brief using team-based communication and collaboration tools provides evidence of collaborating and contributing to team goals and outputs.

DevOps Principles in Practice (J897 48)

- Collaboration between development and operations teams provides evidence of teamwork, collaboration, and shared problem-solving.
- Using tools (for example Git, repositories, boards, sprints) provides evidence of using communication channels and collaborative tools.
- Feedback loops, continuous improvement, and iterative development provide evidence of reflective practice and solution-oriented collaboration.

Criterion 5 descriptors

Criterion 5	Achieved	Merit	Distinction
Demonstrate regard for legal requirements and consideration of ethical and sustainability issues	The learner: • demonstrates awareness of the ethical implications of professional activities • identifies and complies with the relevant laws, regulations and standards, such as data protection, intellectual property, and cyber security • respects the rights, interests, and perspectives of different stakeholders, such as college staff, peers and other learners, and seeks to balance them in a fair and inclusive manner • reflects on the ethical and social implications of their actions and decisions, and considers the potential benefits and harms for themselves, others, and the environment	The learner: • evaluates the ethical implications of professional activities • identifies and complies with the relevant laws, regulations and standards, such as data protection, intellectual property, and cyber security • recognises and respects the rights, interests, and perspectives of different stakeholders, such as college staff, peers and other learners, and seeks to balance them • evaluates and reflects on the ethical and social implications of their actions and decisions, and considers the potential benefits and harms for themselves, others, and the environment	The learner: • evaluates the ethical implications of professional activities and promotes ethical behaviour • identifies and complies with the relevant laws, regulations and standards that apply to their field of software development • recognises and fully respects the rights, interests, and perspectives of all stakeholders and seeks to balance them in a fair and inclusive manner

Criterion 5	Achieved	Merit	Distinction
Demonstrate regard for legal requirements and consideration of ethical and sustainability issues (continued)	The learner: • applies the basic principles and practices of sustainability in their work, reducing waste through efficient development processes	The learner: • applies the principles and range of practices of sustainability in their work, including reducing waste through efficient development processes and saving energy	The learner: • evaluates and critically reflects on the ethical and social implications of the actions and decisions taken by themselves and the team, and considers the potential benefits and harms for themselves, others, and the environment • effectively applies the principles and range of practices of sustainability in their work, including reducing waste through Lean processes, saving energy, and promoting innovation

Criterion 5 guidance

This criterion relates to learners understanding that they should not develop ethically unsound software applications, such as those that promote discriminatory practices or enable unlawful activity. Software should also operate within the bounds of applicable legislation, such as the legal requirement for data protection and accessibility. Learners should understand the impact of software systems and applications on the environment and other aspects of sustainability, and take specific action. This competence can be evidenced in the following units:

Professional Practice in Software Development (J7EE 48)

- Evaluating ethical considerations in the team project reflection provides evidence of evaluating the ethical implications of professional activities.
- Reflecting on individual and team contributions, including social and professional impacts, provides evidence of evaluating the ethical and social implications of actions and decisions.
- Documenting and presenting ethical and sustainability considerations during a final presentation provides evidence of considering potential benefits and harms to self, others, and the environment.
- Developing sustainability knowledge and applying it in a vocational context provides evidence of applying principles and practices of sustainability in development processes.
- Understanding and complying with professional standards supports identifying and complying with relevant laws, regulations, and standards.

Code Security (J7EB 48)

- Using secure coding practices and threat mitigation techniques provides evidence of complying with cybersecurity standards and data protection.
- Identifying and replacing vulnerable components and logging audit data provide evidence of complying with regulations and ensuring ethical handling of code and data.
- Understanding OWASP and legal responsibilities in security provides evidence of identifying and complying with cybersecurity and legal standards.

Software Engineering Methods (J7EF 48)

- Performing validation and design reviews with stakeholder input provides evidence of respecting stakeholder perspectives and balancing interests.
- Modelling system behaviours and assessing usability, reliability and maintainability supports evidence of considering ethical implications and sustainability in design choices.

Database Design Fundamentals (J8FK 47)

- Data protection and database security, including authentication, provide evidence of following standards on data protection and cybersecurity.
- Integrity constraints, data access control, and responsible handling of data provide evidence of ethical handling of user data and system access.

DevOps Principles in Practice (J897 48)

- Secure coding, dependency checks, pipeline integrity, and infrastructure security provide evidence of cybersecurity standards and responsible software practices.
- Process efficiency through automation, CI/CD, and containerisation provides evidence of sustainability through efficient resource use and reduced redundancy.

Database Design and Development (J7DV 48)

- Implementation of user permissions, data integrity, and referential control provides evidence of legal and ethical compliance in data management.
- Management of user accounts and privileges provides evidence of data protection and responsible access control.

Criterion 6 descriptors

Criterion 6	Achieved	Merit	Distinction
Develop meta-skills	The learner adequately engages with the process of meta-skills development in the context of the qualification by: • carrying out a self-assessment of meta-skills, giving reasons for ratings or judgements made • setting clear and measurable goals, plus action strategies to develop meta-skills in all three categories • using reflective practice strategies to track progress and analyse the links between course activities, experiences, and meta-skills development	The learner demonstrates a clear commitment to the process of meta-skills development in the context of the qualification by: • carrying out a self-assessment of meta-skills, giving some insightful reasons for ratings or judgements made • setting clear and measurable goals, plus action strategies to develop meta-skills in all three categories • using reflective practice strategies to track progress and demonstrate some insight into the impact of their course activities and experiences on their meta-skills development	The learner demonstrates a strong commitment to the process of meta-skills development in the context of the qualification by: • carrying out a self-assessment of meta-skills, giving some insightful reasons for ratings or judgements made • setting clear and measurable goals, plus action strategies to develop meta-skills in all three categories, and updating these as required • using reflective practice strategies effectively to track progress and demonstrate insight into the impact of their course activities and experiences on their meta-skills development

Criterion 6 guidance

You must refer to the meta-skills assessment guidance when grading meta-skills. You can find meta-skills teaching, learning and assessment resources on SQA's meta-skills web page.

Competence in individual meta-skills is not being judged here, for example the quality of a learner's feeling or creativity. Rather, it is the process of development the learner goes through — planning, developing, and reflecting — that should be evidenced and assessed.

Although a meta-skills outcome is located in one unit, evidence of meta-skills development can be gathered from any activity at any time during the course. For meaningful reflection to take place, the process of meta-skills development should happen continually throughout the course. The range of contexts in which this can happen is very wide, and dependent on the sector, as well as individual preferences. Each unit signposts opportunities for meta-skills development.

Additional grading guidance

Grading model

The competence criteria reflect the academic, technical, and professional skills and behaviours learners should demonstrate in their performance in this qualification. The competence criteria are described in generic terms, so you can apply them regardless of which optional units a learner completes. This allows you to use evidence from any mandatory or optional unit when evaluating the competencies, as indicated in the grading matrix.

Each criterion has a grading matrix entry with performance statements at three levels in the form of a rubric that will help you evaluate and grade consistently. There is separate guidance on grading the meta-skills competence criterion in Meta-skills — assessment and grading information for centres.

When grading an individual criterion, you should refer to the grading matrix, which identifies where you are most likely to find relevant evidence across the course units. You should determine which rubric statement best reflects the quality and depth of the learner's submitted evidence for each contributing unit. Where multiple units contribute to a single criterion (for example 'Test, deploy and document software'), you should use the highest level of performance demonstrated in any unit to inform the grade. For instance, if the evidence for Unit A indicates that the learner 'creates technical and user documentation to an adequate standard' while the evidence for Unit B shows that the learner 'creates technical and user documentation that is clear, concise and easy to understand', then the evidence from Unit B should be applied in forming a grade judgement for this criterion. We have provided a more complete example for the criterion below. For this example, you would assign a grade of Merit to the criterion:

Criterion 3	Achieved	Merit	Distinction
Test, deploy and document software	The learner: • creates and implements a test plan that confirms that the solution satisfies the software requirement specification **[Unit A]** • creates and executes a plan that outlines the steps to deploy the software to a target system **[Unit A]** • demonstrates security practices through authentication and authorisation **[Unit B]** • creates technical and user documentation to an adequate standard **[Unit A]**	The learner: • creates, justifies and implements a test plan that confirms that the solution satisfies the software requirement specification **[Unit B]** • creates, documents and executes a clear plan for deployment of the software to a target system **[Unit B and Unit C]** • demonstrates good security practices to protect data through code inspection, encryption, authentication and authorisation • creates technical and user documentation that is clear, concise and easy to understand **[Unit B]**	The learner: • creates, justifies and implements a comprehensive test plan that verifies that the solution correctly satisfies the software requirement specification **[Unit C]** • creates a detailed, clear plan for deployment that includes instructions for installing and configuring the software on a target system and executes it • demonstrates comprehensive security practices to protect data through code inspection and analysis, encryption, authentication and authorisation • creates technical and user documentation that is comprehensive, clear and easy to understand

In the example above, you would discount the evidence from Unit A, as for each rubric there is evidence at a higher level of performance from either Unit A or Unit B. While Unit C provides evidence for one rubric at Distinction, in the main the learner performance is evidenced at Merit.

The tables below illustrate the process of moving from the evidence of performance level as described in the individual rubrics to an assigned grade for each criterion. These examples are for illustrative purposes only.

The process requires the exercise of your professional judgement, taking into account the relative contributions that each rubric makes to the key competence expressed in the criterion.

Criterion 1

Achieved	Merit	Distinction
Rubric 1 Achieved	**Rubric 1 Merit**	Rubric 1 Distinction
Rubric 2 Achieved	Rubric 2 Merit	Rubric 2 Distinction
Rubric 3 Achieved	**Rubric 3 Merit**	Rubric 3 Distinction

Assigned grade: Merit

Criterion 2

Achieved	Merit	Distinction
Rubric 1 Achieved	Rubric 1 Merit	**Rubric 1 Distinction**
Rubric 2 Achieved	**Rubric 2 Merit**	Rubric 2 Distinction
Rubric 3 Achieved	**Rubric 3 Merit**	Rubric 3 Distinction
Rubric 4 Achieved	Rubric 4 Merit	Rubric 4 Distinction

Assigned grade: Merit

Criterion 3

Achieved	Merit	Distinction
Rubric 1 Achieved	Rubric 1 Merit	Rubric 1 Distinction
Rubric 2 Achieved	Rubric 2 Merit	Rubric 2 Distinction
Rubric 3 Achieved	**Rubric 3 Merit**	Rubric 3 Distinction

Assigned grade: Achieved

Criterion 4

Achieved	Merit	Distinction
Rubric 1 Achieved	Rubric 1 Merit	Rubric 1 Distinction
Rubric 2 Achieved	Rubric 2 Merit	Rubric 2 Distinction
Rubric 3 Achieved	**Rubric 3 Merit**	Rubric 3 Distinction

Assigned grade: Achieved

Criterion 5

Achieved	Merit	Distinction
Rubric 1 Achieved	**Rubric 1 Merit**	Rubric 1 Distinction
Rubric 2 Achieved	Rubric 2 Merit	**Rubric 2 Distinction**
Rubric 3 Achieved	Rubric 3 Merit	**Rubric 3 Distinction**
Rubric 4 Achieved	**Rubric 4 Merit**	Rubric 4 Distinction

Assigned grade: Distinction

Criterion 6

Achieved	Merit	Distinction
Rubric 1 Achieved	Rubric 1 Merit	Rubric 1 Distinction
Rubric 2 Achieved	**Rubric 2 Merit**	Rubric 2 Distinction
Rubric 3 Achieved	Rubric 3 Merit	Rubric 3 Distinction

Assigned grade: Achieved

After completing the grading matrix for a learner using available evidence, you assign the whole qualification grade holistically. You must consider the relative contribution of each criterion to the overall aims of the qualification. The first two criteria in the grading matrix encapsulate the learner's knowledge and understanding of software development concepts and their skills in applying them to designing and developing software solutions. The remaining four criteria can be considered as having similar relative importance in grading.

The final grade must reflect how well the learner has demonstrated the expected academic, technical, and professional knowledge, skills and behaviours over the course of their studies and in the work submitted as evidence.

Worked example of grading model

The table below illustrates the process of arriving at a final grade for a learner, adopting a holistic approach to judgement.

Criterion	Achieved	Merit	Distinction
1. Demonstrate knowledge of concepts relating to software development		Merit	
2. Design and develop software		Merit	
3. Test, deploy and document software		Merit	

Criterion	Achieved	Merit	Distinction
4. Collaborate and communicate in a team context	Achieved		
5. Demonstrate regard for legal requirements and consideration of ethical and sustainability issues	Achieved		
6. Develop meta-skills			Distinction

In arriving at a final grade of Merit for this learner, you would note that the learner does not reach Merit level in demonstrating regard for legal requirements and consideration of ethical and sustainability issues, nor in their contribution to teamwork. However, the learner's strengths in designing and developing software, testing and deploying software solutions and demonstrating knowledge and understanding of software development concepts are vital competences for a software developer and support the decision to award a Merit grade.

Administrative information

Published: October 2025 (version 1.0)

History of changes

Version	Description of change	Date

Please check SQA's website to ensure you are using the most up-to-date version of this guide.

If a unit is revised:

- no new centres can be approved to offer the previous version of the unit
- centres should only enter learners for the previous version of the unit if they can complete it before its finish date

For more information on NextGen: HN Qualifications please visit the NextGen: HN web page.