Advanced Higher
Coursework
Assessment Task

Advanced Higher Computing Science

Project

Assessment task

This document provides information for teachers and lecturers about the coursework component of this course in terms of the skills, knowledge and understanding that are assessed. It **must** be read in conjunction with the course specification.

Valid from session 2025-26 and until further notice.

This edition: August 2025 (version 3.0)

Contents

Introduction

This document contains instructions for teachers and lecturers, marking instructions and instructions for candidates for the Advanced Higher Computing Science project. You must read it in conjunction with the course specification.

This project has **80 marks** out of a total of 135 marks available for the course assessment.

This is one of two course assessment components. The other component is a question paper.

Instructions for teachers and lecturers

Time

There is no time limit for the project. It is recommended that the project is completed within 40 hours. This can be broken down for each stage as follows:

- Analysis — 5 hours
- Design — 10 hours
- Implementation — 15 hours
- Testing — 8 hours
- Evaluation — 2 hours

Candidates should start at an appropriate point in the course.

Supervision, control and authentication

The project is conducted under some supervision and control.

Candidates can complete part of the work outwith the learning and teaching setting, therefore you must exercise professional responsibility to ensure that evidence submitted by a candidate is their own work.

You should put in place ways to authenticate candidate evidence, for example:

- regular checkpoint or progress meetings with candidates
- checklists which record activity and progress

Group work approaches can be helpful to simulate real-life situations, share tasks and promote team-working skills; however, you can only use these to prepare candidates for assessment. Group work is not allowed once formal work on assessment has started.

Resources

This is an open-book assessment. Candidates can access any appropriate resources.

Candidates are required to design and code their solution, and should be aware that extensive use of resources, such as pre-written module libraries, frameworks and software plug-ins may not allow them to demonstrate these skills and access all marks available.

Reasonable assistance

Candidates must carry out the assessment independently. However, you can provide reasonable assistance prior to, and during, the formal assessment process.

The term ‘reasonable assistance’ is used to balance the need for support with the need to avoid giving too much help. If candidates need more than what is thought to be ‘reasonable assistance’, they may not be ready for assessment.

Reasonable assistance must be limited to constructive comment and/or questioning. You must not adopt a directive role or provide specific advice on how to rephrase, improve responses or provide model answers. Helping candidates on a one-to-one basis in the context of something they have already produced, could become support for assessment and would be going beyond reasonable assistance. For example, you should not prompt candidates to revisit their initial analysis and design as the project develops, and before submitting the final evidence to SQA.

You can give advice on a generic basis, such as how to produce a project plan or how to collate evidence. Where this happens, you should give it to the whole class.

You should advise candidates on their choice of problem, to ensure that it meets the criteria for the Advanced Higher project and is achievable. The purpose of the project is to assess the practical skills of the course, so the project marking criteria is mainly focused on the functionality of the solution. If a project does not meet the criteria, or it relies heavily on frameworks and software plug-ins to do so, it will not allow candidates to demonstrate these skills and access all the marks available.

You should work with individual candidates to ensure that their proposed project meets the criteria set out in the ‘Mandatory requirements’ section. You can use the pre-populated functional requirements in the templates to support this.

Once you have agreed a suitable project with the candidate, they must work independently, with your input limited to constructive comment and/or questioning.

You can support candidates with the following aspects of their projects:

- ensuring they use the appropriate template to collate their evidence, to ensure it is in the format specified by SQA
- ensuring they have all the materials and equipment they need to complete their project
- ensuring they understand the conditions of assessment, and any administrative arrangements around submitting and storing evidence
- technical support

Once projects are completed and submitted, they must not be returned to candidates for further work.

Research

As candidates implement their solution, their project requirements might lead them to implement some code that extends beyond the content of the Advanced Higher course.

Where this is the case, candidates would need to develop new skills/knowledge, so a small number of marks for this are included in the implementation stage.

For some candidates, this could be a distraction and you might advise them to focus on the Advanced Higher concepts and integration, to ensure they can maximise marks in these sections.

Evidence

All candidate evidence (whether created manually or electronically) must be submitted to SQA in paper-based format. There is no need for evidence to be printed single sided or in colour.

Candidates must use one of four templates for gathering evidence, provided by SQA. These are available on the Advanced Higher Computing Science subject page.

You should advise candidates that evidence, especially code, must be clear and legible. This is particularly important when pasting screenshots into a document.

Volume

There is no word count. The volume of evidence is limited by the following criteria, which limit the scope of the project.

The project must have:

- no more than six end-user requirements
- four functional requirements that relate to the Advanced Higher concepts
- three functional requirements that relate to integration
- between four and eight additional functional requirements, including at least two functional requirements that relate to validation of keyboard input

SQA markers will apply a penalty to projects with more than six end-user requirements or more than eight additional functional requirements.

Marking instructions

In line with SQA's normal practice, the following marking instructions for the Advanced Higher Computing Science project are addressed to the marker. They will also be helpful for those preparing candidates for course assessment.

Candidates' evidence is submitted to SQA for external marking.

General marking principles

Always apply these general principles. Use them in conjunction with the detailed marking instructions, which identify the key features required in candidates' responses.

a Always use positive marking. This means candidates accumulate marks for the demonstration of relevant skills, knowledge and understanding; marks are not deducted for errors or omissions.

b If a candidate response does not seem to be covered by either the principles or detailed instructions, and you are uncertain how to assess it, you must seek guidance from your team leader.

c Assess 'completeness' of evidence according to each project. Complete evidence:
- meets all requirements
- relates to the problem
- meets the quality and technical accuracy of Advanced Higher

d Award 0 marks where evidence is:
- not provided
- not related to the problem
- not appropriate to Advanced Higher

e Where bands refer to minor, significant, or major errors and/or omissions, these terms do not indicate the volume required, but the importance of the errors and/or omissions in the context of the project.

f Select the band that most closely describes the evidence provided. Where a range of marks is available for a band, you should determine:
- if the evidence is a closer match to the band above, and if so, award the highest available mark from the range
- if the evidence is a closer match to the band below, and if so, award the lowest available mark from the range

Detailed marking instructions

Analysis of the problem (10 marks)

Evidence requirements	Marks	Marking instructions	
Description of the problem, including: ♦ an outline of the problem, identifying Advanced Higher concepts and integration ♦ any constraints	2	**2 marks** **1 mark**	Complete and detailed description of the problem that meets all of the evidence requirements, including integration. Description of the problem, that meets some of the evidence requirements.
UML use case diagram, showing integration, and defining: ♦ actors ♦ use cases ♦ relationships	2	**2 marks** **1 mark**	Complete, detailed and integrated use case diagram that meets all of the evidence requirements, including integration. Use case diagram that meets some of the evidence requirements.
Requirements specification, including: ♦ end-user requirements ♦ functional requirements	4	**3-4 marks** **1-2 marks**	Complete or almost complete and detailed requirements specification that meets all end-user and functional requirements for a fully-working, integrated solution. Requirements specification, with some missing information for a fully-working, integrated solution.
Project plan for each stage, including: ♦ identified tasks ♦ resources required ♦ estimate of timings	2	**2 marks** **1 mark**	Complete and detailed project plan that meets all of the evidence requirements. Project plan that meets some of the evidence requirements.

Design of the solution (20 marks)

Evidence requirements	Marks	Marking instructions	
Design of Advanced Higher concepts	**6**	**5-6 marks**	Complete or almost complete and detailed design.
		3-4 marks	Partially complete and detailed design, with some errors and/or omissions.
		1-2 marks	Incomplete design, with a number of significant errors and/or omissions.
Design of integration	**5**	**4-5 marks**	Complete or almost complete and detailed design of integration.
		2-3 marks	Partially complete design of integration, with some errors and/or omissions.
		1 mark	Minimal design of integration.
Design of additional functional requirements	**4**	**3-4 marks**	Complete or almost complete and detailed design of additional functional requirements, including input validation.
		1-2 marks	Partially complete design of additional functional requirements.
User-interface design shows inputs, processes and outputs, and matches the end-user and functional requirements	**5**	**4-5 marks**	Complete or almost complete and detailed user-interface design, showing validated inputs and outputs, matching the end-user and functional requirements, and indicating the underlying processes.
		2-3 marks	Partially complete user-interface design, with some errors and/or omissions.
		1 mark	Minimal user-interface design.

Implementation (30 marks)

Evidence requirements	Marks	Marking instructions	
Implemented Advanced Higher concepts and additional functional requirements that match the design	**12**	**11-12 marks**	Complete or almost complete and fully-working implementation that matches the design.
		9-10 marks	Partially complete and working implementation that closely matches the design, but has some minor errors and/or omissions.
		7-8 marks	Partially complete and working implementation that matches the design, with some significant errors and/or omissions.
		5-6 marks	Partially complete implementation that matches some aspects of the design, and has a number of significant errors and/or omissions
		3-4 marks	Incomplete implementation, with limited match to the design due to major errors and/or omissions.
		1-2 marks	Minimal implementation that does not match the design.
Implemented integration, that matches the design	**6**	**5-6 marks**	Complete or almost complete and fully-working integration that matches the design.
		3-4 marks	Partially complete and working integration that matches some aspects of the design, but with some significant errors and/or omissions.
		1-2 marks	Incomplete implementation, with limited match to the design and a number of significant errors and/or omissions.

Implementation (30 marks) (continued)

Evidence requirements	Marks	Marking instructions	
Implemented user interface, that matches the design	**3**	**3 marks**	Complete or almost complete and fully-working user interface that matches the design.
		2 marks	Partially complete and working user interface that matches some aspects of the design, but with some significant errors and/or omissions.
		1 mark	Incomplete user interface, with limited match to the design and a number of significant errors and/or omissions.
Description of new skills and/or knowledge researched and developed	**4**	**3-4 marks**	Complete, or almost complete, and detailed description of research and application of new skills and/or knowledge, that extends beyond what is required for the Advanced Higher course, developed during the implementation stage.
		1-2 marks	Partially complete description of research and application of new skills and/or knowledge developed during the implementation stage.
Log of ongoing testing, including: ♦ a description of issues resolved ♦ references used to resolve these issues	**5**	**4-5 marks**	Complete, or almost complete, and detailed log of ongoing testing, describing issues resolved, and evidencing solutions and references throughout the implementation stage.
		2-3 marks	Partially complete log of ongoing testing.
		1 mark	Incomplete log of ongoing testing.

Testing the solution (15 marks)

Evidence requirements	Marks	Marking instructions	
Description of persona and test cases including: ♦ description of persona ♦ description of test cases ♦ description of results of testing using persona and test cases	**3**	**3 marks** **2 marks** **1 mark**	Complete and detailed description of persona, test cases and results of testing. Partially complete description of persona, test cases and results of testing. Incomplete description of persona, test cases and results of testing.
A comprehensive plan for carrying out final testing of all functional requirements	**6**	**5-6 marks** **3-4 marks** **1-2 marks**	Complete and detailed test plan. Partially complete test plan. Incomplete test plan.
Evidence of requirements testing	**6**	**5-6 marks** **3-4 marks** **1-2 marks**	Complete evidence of requirements testing that matches the test plan. Partially complete evidence of requirements testing. Incomplete evidence of requirements testing.

Evaluation of the solution (5 marks)

Evidence requirements	Marks	Marking instructions	
Evaluation of the solution in terms of fitness for purpose, by discussing: ♦ how closely the solution matches the requirements specification ♦ the results of testing	**3**	**3 marks** **2 marks** **1 mark**	Complete and detailed evaluation of the solution's fitness for purpose that meets all of the evidence requirements. Partially complete evaluation of the solution that meets some of the evidence requirements. Incomplete evaluation of the solution that meets minimal evidence requirements.
Evaluation of the solution in terms of: ♦ future maintainability ♦ robustness	**2**	**2 marks** **1 mark**	Complete and detailed evaluation of the solution's future maintainability and robustness. Partially complete evaluation of the solution's future maintainability and robustness.

Instructions for candidates

This assessment applies to the project for Advanced Higher Computing Science.

This project has **80 marks** out of a total of 135 marks available for the course assessment.

It assesses the following skills, knowledge and understanding:

- applying computational thinking to solve a complex computing problem
- analysing a complex problem within a computing science context
- designing, developing, implementing, testing, and evaluating a digital solution to a complex problem
- demonstrating advanced skills in computer programming
- communicating understanding of complex concepts related to computing science, clearly and concisely, using appropriate terminology

Your teacher or lecturer will let you know if there are any specific conditions for doing this assessment.

For this project, you have to identify a computing science problem, agreed with your teacher or lecturer. You need to develop a solution to the problem, from analysis through to evaluation. You gain marks for the following stages of the project:

- analysis of the problem (**10 marks**)
- design of the solution (**20 marks**)
- implementation (**30 marks**)
- testing the solution (**15 marks**)
- evaluation of the solution (**5 marks**)

In this document, there is guidance on:

- how much support and assistance your teacher or lecturer can give you
- what evidence you need to collect
- choosing a suitable problem for your project
- what you need to do at each stage of the project

You must read these instructions with the template for your project. You must use one of the four templates provided by SQA, which are available on the Advanced Higher Computing Science subject page. The template contains detailed guidance on what you need to do at each stage of your project and the type of evidence required for your project.

Support and guidance from your teacher or lecturer

You must complete this project independently; however, your teacher or lecturer can provide you with guidance to help develop your thinking as you progress. This could be:

- general support in class on broad areas, such as project planning
- constructive questioning with you on an individual basis
- constructive comments to help you find a solution

Your teacher or lecturer cannot tell you specifically how to proceed with your project, how to rephrase or improve responses, or provide you with model answers.

Evidence to be gathered

You need to gather evidence for each stage of the project. Evidence can include program listings, screenshots, web page source files, data files or similar, as appropriate. You must print your evidence and submit it to SQA for marking. You must use the appropriate Advanced Higher Computing Science Project Template to present your evidence.

You should ensure that you:

- print code in a format that is legible (suggested minimum font size 11pt. If the programming environment does not have a printing facility, consider copying and pasting into a word processor rather than screenshots)
- print screenshots so that all content is legible
- use a colour contrast between background and text (especially in code) that makes evidence legible
- do not delete any headings or page numbers from the template
- update the contents table before you submit your completed template
- follow the prompts in the template to provide evidence of the design and implementation of each functional requirement
- add your full code to the appendix at the end of the template

You will probably work on your project for several weeks and during that time, you will produce many types of evidence.

Although there is no page limit or maximum word count for your evidence, marks are awarded for the quality of your work, not the quantity.

Choosing a suitable problem

You must choose a suitable problem for your project. You may already have an idea, or you can explore ideas with other candidates and/or your teacher or lecturer. You can also get ideas from online resources, industry news, television, local business partners or STEM ambassadors. A successful project is likely to be about something you are interested in.

It is possible to complete some projects within your centre, but you could consider a project that requires collaboration with a university, college or local industry. Your teacher or lecturer can advise you about this.

Your chosen problem must allow you to meet the criteria below.

It is essential that you are clear on what Advanced Higher concepts your project includes and how it integrates with the other area of the course. The mandatory requirements on the following pages will help you.

You should focus on the functionality of your solution, rather than its appearance.

If you spend too much time learning to use frameworks, software plug-ins and graphical tools, this could distract you from ensuring your project meets all the criteria. The project itself does not have to be overly complex. If you remain focused on the essential criteria, you can access all the marks.

You must discuss your project idea with your teacher or lecturer. This ensures that it meets the project criteria set out below and is achievable within the constraints of time, expertise and resources available.

Project criteria

Your project must be one of the following:

- software design and development (using object-oriented programming) integrating with database design and development
- software design and development (using procedural programming) integrating with database design and development
- database design and development integrating with software design and development or web design and development
- web design and development integrating with database design and development

Your project must have:

- no more than six end-user requirements
- four functional requirements that relate to the Advanced Higher concepts
- three functional requirements that relate to integration
- between four and eight additional functional requirements, including at least two functional requirements that relate to validation of keyboard input

A single functional requirement must relate to one single process that the system will carry out, for example a single SQL query.

If you include more than six end-user requirements or more than eight additional functional requirements in your project, or if you do not use one of the mandatory templates, SQA markers will apply a penalty.

Once you have completed your project, based on these requirements, and submitted your evidence to your teacher or lecturer, you can continue to develop your project by adding more functionality and developing the user-interface. You must not submit this additional work to SQA.

You should review your proposed project against the mandatory requirements on the following pages. Examples of suitable projects for each possible combination are on the following pages. You can choose or adapt one of these examples, or use an idea of your own, however, remember to discuss your project idea with your teacher or lecturer to ensure it meets the criteria and is achievable.

Mandatory requirements

Software design and development (SDD) and database design and development (DDD) project (object-oriented programming)

An object-oriented program with an array of objects

with

one standard algorithm (applied to the data structure above), from the following:

- binary search
- insertion sort
- bubble sort

integrating with:

a database with at least one table. The solution should:

- receive a query input
- open and close the database
- execute an SQL query
- format the query results

Example

A computer game requires a variety of regular polygons to appear in the top, left, right or bottom of the screen.

Players respond to each shape by entering the correct number of sides and the letter T, L, R or B to indicate the position of the shape.

This will require an object-oriented program that:

- stores players' names and scores in a Player class
- connects to a database, executes a query that retrieves data from the database table, and stores it in an array of player objects
- will validate the letter and name entered by the player
- uses methods to update players' names and scores in the array of objects
- uses the insertion sort algorithm to arrange the names and scores in descending order of score
- displays the top 10 scores
- connects to the database to execute a query to add a player's name and score to the database table

For this type of project, you must complete the SDD (object-oriented) template and submit it as evidence.

Software design and development (SDD) and database design and development (DDD) project (procedural programming)

A procedural program with a 2D array or an array of records

with

one standard algorithm (applied to the data structure above), from the following:

- binary search
- insertion sort
- bubble sort

integrating with:

a database with at least one table. The solution should:

- receive a query input
- open and close the database
- execute an SQL query
- format the query results

Example
A computer game requires a variety of regular polygons to appear in the top, left, right or bottom of the screen.

Players respond to each shape by entering the correct number of sides and the letter T, L, R or B to indicate the position of the shape.

This will require a procedural program that:

- is modular and makes use of parameter passing
- stores players' names and scores in a database table
- connects to the database, executes a query that retrieves data from the database table, and stores it in an array of records
- will validate the letter and name entered by the player
- adds a player's name and score to the array of records
- uses the bubble sort algorithm to arrange the names and scores in descending order of score
- displays the top 10 scores
- connects to the database to execute a query to add a player's name and score to the database table

For this type of project, you must complete the SDD (procedural) template and submit it as evidence.

Database design and development (DDD) and software or web design and development (SDD or WDD) project

a database with a minimum of four related tables, created using SQL

with
three SQL queries (applied to data stored in the tables created above):

- one query that incorporates a subquery using ANY or EXISTS
- one query that extends across at least three tables
- one query that incorporates HAVING

integrating with:

a programming or web interface to receive a query input and a display-formatted query output

Example
A relational database is needed to store the personal details, meter readings, tariffs and bills of electricity customers.

This will require an SQL database:

- with four tables
- with an SQL query that uses a sub-query to display details of customers who have not paid their bill
- with an SQL query extending across three tables to calculate and display the bill for any customer whose customer number and new meter reading is entered by the user
- will validate the customer number and meter reading entered by the user
- with a query that uses HAVING to list customers whose bill exceeds the value of the average bill
- that integrates with a program or web page to generate an interface that enables query execution

For this type of project, you must complete the DDD template and submit it as evidence.

Web design and development (WDD) and database design and development (DDD) project

A website that includes:

- form elements (action, method, and name)
- multiple layouts using a media query

with

server-side processing (PHP) used to assign:

- form variables and process form data
- session variables

integrating with:

a database with at least one table. The solution should:

- receive a query input
- open and close the database
- execute an SQL query
- format the query results

Example

Members of a swimming club need to be able to register for the club's annual swimming competition by completing an HTML form on a website.

This will require a website that:

- uses a database table to store members' registration details
- uses HTML form elements to receive member registration details
- uses a media query to display the website on a mobile phone
- assigns registration details to PHP variables
- uses PHP session variables to display a personalised 'registration successful' message on a separate web page
- connects to the database and executes a query to retrieve data (from the database table) to display a list of members who have registered for any race the user is interested in
- will validate the race duration and age category entered by the user

For this type of project, you must complete the WDD template and submit it as evidence.

Tips for candidates

1 You must be sure from the outset which of the four project combinations you are following. Being clear on how you will use the Advanced Higher concepts and integration within your solution will help you focus on these essential elements as you develop your solution.

2 You should view your requirements specification as a 'golden thread' that is vital to every stage of the project. The template will help you to track and provide evidence of each of these through each stage of the project.

3 Ensure you highlight or label evidence, such as code and diagrams, to cross-reference with the requirements specification.

4 Remember that markers need to see evidence of the Advanced Higher concepts that you have coded. This is especially important if you are using a framework or software plug-in that generates lots of code. The template will help you to present this evidence. Provide the full code as an appendix.

5 Many marks are available for the design and implementation of Advanced Higher concepts and integration. It is important that you focus on this. While a small number of marks are available for demonstrating new skills or knowledge, don't let this (or making your user interface look nice) distract you from the functionality, as it is the functionality that carries the majority of the marks.

Guidance for each stage

Developing a solution

As you work through your project, you must follow these five stages of development:

- analysis
- design
- implementation
- testing
- evaluation

You can follow these stages using an iterative approach or using an agile methodology — where you break the project down into several small iterations of design, implement and test.

Whatever approach you use, each stage of development should continue from the previous stage. For example, you should create your design from the requirements identified at the analysis stage; you should implement a solution from the design you created; and so on.

The template has details of the specific evidence requirements for each stage of the project you complete. You gain marks based on the evidence you submit for each of these stages.

If you need to go back and revisit a previous stage (for example to add detail to analysis or improve a design), you should ensure that you submit only the final version as evidence.

Analysis of the problem 10 marks

Before you begin designing and developing a solution, you must analyse the problem that you are going to solve, to ensure that you fully understand every aspect of it. This stage of your project should take around 5 hours.

Description of the problem (2 marks)

Describe your problem. Your description should include:

- an outline of the problem, identifying the Advanced Higher concepts and integration (see the examples earlier in this document)
- any constraints you identify

UML use case diagram (2 marks)

Draw a UML use case diagram for your problem. Your diagram should define the following:

- actors
- use cases
- relationships

Requirements specification (4 marks)

Produce a requirements specification. Your requirements specification should list:

- no more than six end-user requirements
- four functional requirements that relate to the Advanced Higher concepts
- three functional requirements that relate to integration
- between four and eight additional functional requirements, including at least two functional requirements that relate to validation of keyboard input

Project plan (2 marks)

Create a project plan for the four remaining stages of your project.

Your project plan should include:

- the tasks you complete in each stage
- any resources you need to implement your solution
- an estimate of how long each stage and tasks will take

Design of the solution 20 marks

Now, design your solution based on your requirements specification. This stage of your project should take around 10 hours.

Project design (15 marks)

Design your solution, using appropriate design methodologies or techniques.

Your design should meet the end-user and functional requirements identified at the analysis stage.

User-interface design (5 marks)

Design the user interface for your solution using appropriate design methodologies or techniques.

Implementation 30 marks

Now implement your solution. This stage of your project should take around 15 hours.

Implementation (21 marks)

Implement your solution, including the user interface, ensuring it matches your completed design.

Log of ongoing testing (5 marks)

Produce a log of the ongoing testing you carry out during implementation. Your log should include:

- what you are testing
- descriptions of issues you encounter during testing
- descriptions of how you resolve these issues
- lists of references you use to resolve each issue

Research and development of new skills and/or knowledge (4 marks)

Describe:

- any new skills and/or knowledge that you researched
- why those new skills and/or knowledge were necessary
- how you applied these new skills and/or knowledge to your project

You should reference the resources you used to research and develop these new skills and/or knowledge.

Testing the solution 15 marks

Once you have fully implemented your design, you must carry out final testing on your solution. This testing should be systematic and comprehensive, and based on a test plan.

This stage of your project should take around 8 hours.

Persona and test cases (3 marks)

Describe the:

- characteristics of one persona that you will adopt to test the end-user requirements
- test cases that you will use to test each end-user requirement and indicate the end-user requirement or requirements that will be the focus of each test case

Test the end-user requirements by carrying out the test cases above.

Describe the results of testing using the persona and test cases described above.

Final test plan (6 marks)

Create a plan of how you will carry out final testing of all functional requirements.

Requirements testing (6 marks)

Test your solution and provide evidence of each functional requirement test identified in your plan.

Evaluation of the solution 5 marks

You must now evaluate your solution. This stage of your project should take around 2 hours.

Evaluation report (5 marks)

Produce a report to evaluate your solution. This should include:

- the fitness for purpose of your solution, discussing:
  - how closely your solution matches all requirements stated in your requirements specification
  - the results of your testing
- the future maintainability and robustness of your solution

Administrative information

Published: August 2025 (version 3.0)

History of changes

Version	Description of change	Date
1.2	Updated the guidance for teachers, lecturers and candidates on: ♦ selecting a suitable project ♦ using frameworks and software plug-ins ♦ developing new skills/knowledge ♦ gathering evidence Updated the diagrams to clarify that these are mandatory requirements and to define ‘Advanced Higher concepts’. Included more detailed evidence requirements for design, implementation and testing, incorporating the requirement for input validation. Removed the requirement for scope and boundaries in analysis. Added in ‘Tips for candidates’ section.	September 2022
1.3	‘Candidate checklist’ replaced with detailed ‘Evidence checklists’ for each type of project.	September 2023
2.0	Updated guidance for teachers, lecturers and candidates to: ♦ add suggested timings ♦ introduce a maximum limit of end-user and functional requirements ♦ refer to checklist and template as appropriate ♦ change total mark from 160 to 135 (pages 1 and 11) ♦ replace reference to parallel arrays with arrays of records in example of SDD/DDD project (page 15)	September 2024

Version	Description of change	Date
3.0	For session 2025–26, we have: ♦ removed the option to complete SDD/WDD and WDD/SDD projects ♦ introduced a mandatory template for each project option. These contain bespoke prompts and headings for the functional requirements relating to mandatory Advanced Higher concepts and integration. Any further functional requirements are referred to as ‘Additional functional requirements’ ♦ reviewed the mandatory requirements to ensure a comparable level of demand and workload between projects ♦ introduced a penalty that will be applied to projects with more than six end-user requirements or more than eight additional functional requirements ♦ reviewed the requirements and mark allocations in the design stage ♦ clarified the requirements for persona, test cases and the test plan in the testing stage We have revised this document to: ♦ remove references to SDD/WDD and WDD/SDD projects ♦ update the criteria to reflect the project options and the maximum number of user-requirements and additional functional requirements ♦ update the mandatory requirements and examples for each project in line with pre-populated mandatory requirements, to ensure comparable level of demand and workload and to improve accessibility ♦ include references to the templates and amended guidance, such as ‘Evidence’ and ‘Tips for candidates’ as required ♦ update the detailed marking instructions to include reference to ‘additional functional requirements’ (worth 4 marks) ♦ update the detailed marking instructions for ‘design of integration’, increasing this to 5 marks (from 4 marks) ♦ clarify the requirements for persona and test cases and update the detailed marking instructions ♦ remove general stage-by-stage guidance and candidate evidence checklists as these are now embedded into templates	August 2025

Version	Description of change	Date
	What you need to do differently You must use this Advanced Higher Computing Science Project Assessment Task and new accompanying templates (dated August 2025). Use this document to update your teaching notes and teaching practice. Make sure candidates are aware of the criteria for end-user and functional requirements, the penalty for exceeding these, and the requirement to use one of the four templates from SQA's website.	

Note: you are advised to check SQA's website to ensure you are using the most up-to-date version of this document.

Security and confidentiality

This document can be used by SQA approved centres for the assessment of National Courses and not for any other purpose.