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 2024-25 and until further notice.

This edition: September 2024 (version 2.0)

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 of one of the six possible project combinations set out in the 'Mandatory requirements' section. You can use the 'Choosing a suitable problem — checklist' 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:

- printing, collating and labelling 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.

The evidence checklists at the end of this document detail all evidence to be gathered for each of the possible project combinations. You should encourage candidates to use them to ensure they submit all evidence to SQA. A template for gathering evidence is also 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.

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	**4**	**3–4 marks**	Complete or almost complete and detailed design showing integration.
		1–2 marks	Partially complete design.
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.
Overall design matches the requirements specification	**5**	**4–5 marks**	Design matches all or almost all of the requirements specification.
		2–3 marks	Design matches some of the requirements specification.
		1 mark	Minimal match to the requirements specification.

Implementation (30 marks)

Evidence requirements	Marks	Marking instructions	
Implemented Advanced Higher concepts and 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	
A comprehensive plan for carrying out final testing of the fully implemented solution, including: ♦ all requirements ♦ description of tests ♦ persona and test cases	**6**	**5–6 marks** **3–4 marks** **1–2 marks**	Complete and detailed test plan that meets all evidence requirements. Partially complete test plan that meets some evidence requirements. Incomplete test plan that meets minimal evidence requirements.
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.
Description of the results of the test cases	**3**	**3 marks** **2 marks** **1 mark**	Complete and detailed description of the results of the test cases that matches the test plan. Partially complete description of the results of the test cases. Incomplete description of the results of the test cases.

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**	Complete and detailed evaluation of the solution's fitness for purpose that meets all of the evidence requirements.
		2 marks	Partially complete evaluation of the solution that meets some of the evidence requirements.
		1 mark	Incomplete evaluation of the solution that meets minimal evidence requirements.
Evaluation of the solution in terms of: ♦ future maintainability ♦ robustness	**2**	**2 marks**	Complete and detailed evaluation of the solution's future maintainability and robustness.
		1 mark	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

There are also evidence checklists that you should use to ensure that you are submitting all the evidence required for the project you have completed. A template for gathering evidence is also available on the Advanced Higher Computing Science subject page.

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 can use the Advanced Higher Computing Science Project Template to present your evidence. You can find this on the Advanced Higher Computing Science subject page.

You should ensure that you:

- include all your evidence, by completing the evidence checklist appropriate for your project
- clearly label your evidence
- annotate code to highlight Advanced Higher concepts and integration
- 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
- submit your evidence in a logical order, with appropriate headings for each stage (you may want to include a contents page and page numbers)

The template is designed to make it easier for you to do this.

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

Each stage of the project provides more detail on the evidence required. For the design and implementation stages, the evidence will depend on the problem you are solving — the evidence checklists detail the specific evidence required for design and implementation for each project combination.

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 areas of the course. The diagrams on the following pages show the minimum mandatory requirements, and will help you.

You should focus on the functionality of your solution, rather than its appearance. Your project must have no more than:

- 6 end-user requirements
- 24 functional requirements

If you spend too much time learning to use frameworks and software plug-ins, 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 based on **one** of the following areas of the course:
  - software design and development
  - database design and development
  - web design and development
- include **two** concepts from this area of the course
- integrate with **one** of the other two areas of the course
- validate all inputs

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) project

The mandatory requirements for an SDD project are shown below, followed by an example of each project.

Concept 1
Object-oriented programming with an array of objects

or

Concept 1
Procedural programming with a 2D array or an array of records

with

Concept 2
One standard algorithm (using the data structure from above), from the following:

- binary search
- insertion sort
- bubble sort

Your SDD project should integrate with either:

Database design and development
Create a database with at least one table

and

open and/or close the database connection to execute an SQL query and (if required), format the results

or

Web design and development
Web user interface to receive input and/or display formatted output

Example

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

Players respond to each shape by pressing keys corresponding to the correct number of sides and the position of the shapes.

The reaction time of players is calculated and stored.
Analyse the problem, create a design and then implement a procedural program that:

- reads previous players' times from a database table, into an array of records
- prepares a 2D array storing randomised shapes and co-ordinates
- displays each shape in the 2D array and calculates the total time taken by each player to press the correct keys
- adds each player's time to the array of records , and bubble sorts by time
- displays the top 10 times
- inserts the players' times into a table

Test the program with a variety of sample data.

Evaluate the solution.

Example

The results of a survey are stored in a file.

A simple web page, with embedded program code, is required to allow a user to search for data within the file and display the results of the search.

Analyse the problem, create a design and then implement object-oriented code that:

- reads the survey data from a file and stores it within an array of objects
- uses a binary search to find user input within the array
- outputs the search results within the web page, formatted using HTML table elements and Inline CSS

Test the program with a variety of survey data and search scenarios.

Evaluate the solution.

Database design and development (DDD) project

The mandatory requirements for a DDD project are shown below, followed by an example of each project.

Concept 1

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

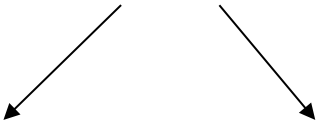
with

Concept 2

SQL queries (using the tables above), that incorporate any two of the following:

- subquery
- one logical operator (NOT, BETWEEN, ANY, EXISTS)
- query across at least three tables

Your DDD project should integrate with either:



Software design and development

Programming interface to receive input and/or display formatted output

or

Web design and development

Web user interface to receive input and/or display formatted output

Example

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

Analyse the problem, design and implement a suitable database using only SQL statements.

Design and implement additional SQL statements to maintain the database, for example to:

- insert, update and remove customers
- generate bills
- update electricity costs

Design and implement a small program to search for a customer, and insert a new meter reading for that customer.

Test the solution with sample data.

Evaluate the solution.

Example

A relational database is needed to store the personal details and health data (for example steps and average heart rate) of members of a gym.

Analyse the problem, design and implement a suitable database using only SQL statements.

Design and implement additional SQL statements to maintain the database, for example to:

- insert, update and remove members
- create statistical output for members

Design and implement a simple web page to allow members to input their step count and average heart rate.

Test the solution with sample data.

Evaluate the solution.

Web design and development (WDD) project

The mandatory requirements for a WDD project are shown below, followed by an example of each project.

Concept 1

Complete website that includes:

- form elements (action, method, and name)
- external CSS
- multiple layouts using a media query
- use of session variables

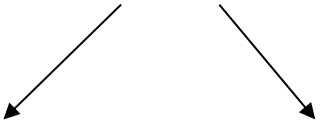
with

Concept 2

Server-side processing (PHP) used to:

- assign variables
- process form data

Your WDD project should integrate with either:



Database design and development

Create a database with at least one table

and

open or close the database connection to execute an SQL query and (if required), format the results

or

Software design and development

Embedded programming using one Advanced Higher level data structure or standard algorithm

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.

Analyse the problem, design and implement a website for the swimming club.
Design and implement a single-table database, along with the required PHP code to validate and store the form details.

Test the solution with sample data.

Evaluate the solution.

Example

Online revision quizzes are needed to prepare learner drivers for their theory test.

Analyse the problem, design, and implement a website for the quizzes that stores the quiz questions and answers, within a 2D array.

When a user submits a quiz answer, a function is called to check the answer and display a result.

Design and implement the required code to store and check users' answers.

Test the solution with sample quiz data.

Evaluate the solution.

Choosing a suitable problem — checklist

Use the following checklist to help you decide on the idea for your project.

a Will the solution to your problem involve implementing two Advanced Higher concepts from one area of the course? **yes** ☐ **no** ☐

These are:

1 _____

2 _____

b Will the solution to your problem involve integration with one of the other two areas of the course? **yes** ☐ **no** ☐

Integrates with _____

c Will the solution to your problem validate all inputs? **yes** ☐ **no** ☐

d Will you be able to complete the project in the time available? **yes** ☐ **no** ☐

e Can you overcome all potential barriers to carrying out your project, for example permissions, logistics, and access to necessary hardware and software? **yes** ☐ **no** ☐

f Does your teacher or lecturer agree with your answers above? **yes** ☐ **no** ☐

Your answers to questions a–c will help you outline your problem in the analysis stage.

If you answer 'no' to any of the above questions, you will need to reconsider your project idea.

Tips for candidates

1 You must be sure from the outset which of the six 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. Consider numbering the requirements to make it easier to cross reference them when designing, implementing, testing, and evaluating.

3 Follow the guidance and use the headings as a template for organising your evidence. Well organised evidence can help ensure you have provided evidence for all stages. You can also highlight or label evidence such as code and diagrams to identify the Advanced Higher concepts and integration, and 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. Highlight and annotate your own work to show you have implemented the concepts being assessed. If there are many pages of code, consider extracting the Advanced Higher concepts into the main body of your evidence and 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/knowledge, don't let this (or making your user interface look nice) distract you from the functionality, as this carries most of the marks.

6 Use the Advanced Higher Computing Science Project Template to help you present your evidence. It is designed to follow the tips above.

7 Use the evidence checklists at the end of this document to ensure that you submit the appropriate evidence, especially for design and implementation, as this will vary depending on the project you complete.

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 following pages detail the requirements for each stage. 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 between 5 and 6 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

Your diagram should include the integration you intend to implement, for example a web project connecting to a database.

Requirements specification (4 marks)

Produce a requirements specification. Your requirements specification should list a maximum of:

- 6 end-user requirements
- 24 functional requirements

Your requirements specification should consider all input validation that is required for your problem.

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

Guidance for producing a project plan

Your list of tasks for each stage could include:

- user-interface design
- implementation of input validation
- ongoing testing

Resources can include access to development tools and end users. Some of these could be available at any time, while others may only be available at certain times. You need to plan to ensure that your project is not held up waiting for resources.

Your timings should allow for holidays, or other events that affect how much time you can spend on your project.

You should review and update your project plan as you work through each stage. **You must submit your final version of the plan as evidence.**

Design of the solution 20 marks

Now, design your solution based on your requirements specification. This stage of your project should take between 10 and 12 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, **and include the integrated part of the project**.

User-interface design (5 marks)

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

Your user-interface design should include annotated wireframes that show all inputs (with notes on validation), underlying processes and outputs. It should also meet the end-user and functional requirements identified at the analysis stage.

Design methodologies

The problem you are solving will determine the design methodologies you use. The following may be suitable:

Software design and development

- use structure diagrams or pseudocode to show:
  - top level design with data flow
  - refinements of Advanced Higher concepts, functional requirements and input validation
- use a UML class diagram — including class names, properties and data types, methods (including constructor) and arguments, and (where appropriate), public and/or private, inheritance to show class structure(s)
- design structure and data type(s) of array of records or 2D arrays
- use wireframes to show user-interface design

Database design and development

- entity relationship diagram — including (where appropriate) entity name, entity type (strong, weak), attributes, relationship participation (mandatory, optional), relationship name and cardinality
- data dictionary — using SQL attribute types and indicating input validation
- query designs

Web design and development

- use structure diagram or pseudocode to show Advanced Higher concepts, functional requirements and input validation
- site navigation structure showing use of session variables if appropriate
- use wireframes to show:
  - user-interface design
  - effect of media query

Implementation 30 marks

Now implement your solution.

Implementation (21 marks)

Implement your solution, including the user interface, ensuring it matches your completed design. This stage of your project should take between 15 and 18 hours.

The problem you are solving will determine the evidence you provide of your implemented solution. The following may be suitable:

Software design and development

- program code
- screenshots of program user interface

Database design and development

- SQL code
- screenshots to show
  - that the structure of the implemented table(s) matches the design
  - any initial value stored in table(s)

Web design and development

- PHP code
- HTML code and page content
- CSS declarations
- screenshots of pages to show
  - user interface
  - effect of media queries

Remember you must provide evidence of the integrated part of your solution.

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

When implementing your functional requirements, you may need to make use of some coding that extends beyond the content of the Advanced Higher course. If this is the case, you will need to carry out some research.

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.

Log of ongoing testing (5 marks)

As you implement your solution, you will encounter errors or problems that you need to solve before you can continue. Take notes of errors, solutions and any reference materials you use, for example websites, forums, textbooks or learning resources. You will need to refer to these notes to produce evidence of ongoing testing

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

You could present your log as a table, using the above bullet points as column headings.

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 between 7 and 9 hours.

Final test plan (6 marks)

Create a plan of how you will carry out final testing of your fully implemented solution.

Your plan should be comprehensive, to ensure that your solution meets all the requirements identified at the analysis stage. It should include:

- all end-user and functional requirements
- a description of the tests you will carry out
- a description of a persona, with a list of test cases that will be used to test the solution with an end user

Note: the end-user can be another candidate, a teacher or a lecturer, who adopts the persona and carries out the test cases.

Requirements testing (6 marks)

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

Evidence should show inputs (including errors if any are generated) and outputs to show that all functional requirements are working correctly.

The problem you are solving will determine the evidence you require. The following may be suitable:

- screenshots showing inputs and any errors generated
- screenshots showing successful implementation of Advanced Higher algorithms
- screenshots showing successful implementation of SQL queries
- screenshots showing successful implementation of media queries

Testing with persona and test cases (3 marks)

Test your solution using the persona and test cases identified in your plan.

Describe the results of each test case — this could be in the form of a short report or a table.

Evaluation of the solution 5 marks

You must now evaluate your solution. This stage of your project should take between 2 and 3 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

Evidence checklist for SDD projects

Analysis

Description of the problem that includes:	Complete
♦ outline of the problem:	
— SDD Advanced Higher (AH) concepts	
— integration with DDD/WDD	
♦ constraints	
UML use case diagram that defines:	**Complete**
♦ actors	
♦ use cases	
♦ relationships	
Requirements specification that includes:	**Complete**
♦ end-user requirements	
♦ functional requirements for SDD	
♦ functional requirements for integration with DDD/WDD	
♦ validation of inputs	
Project plan for each stage that includes:	**Complete**
♦ tasks for each stage	
♦ resources required for implementation	
♦ estimate of timings	

Design

Design of Advanced Higher concepts that includes:	Complete
♦ UML class diagram (object-oriented only)	
♦ Data structure design (2-D array and/or array of records)	
♦ structure diagrams or pseudocode showing:	
— top-level design with data flow	
— refinement of AH algorithm(s) making use of AH data structure	
— input validation	
Design of integration with either:	**Complete**
♦ DDD, including:	
— connection to database	
— data dictionary	
— entity-relationship diagram (if two or more tables)	
— design of query/queries	
OR	
♦ WDD, showing:	
— connection with website	
— how program will send data to web code for display	
— how program will receive input from web code for processing	

User interface design that includes:	Complete
♦ wireframe designs of:	
— input forms showing input validation	
— buttons and menu options showing underlying processes	
— output screens	

Implementation

Implementation of Advanced Higher concepts that includes:	Complete
♦ evidence of implemented code showing:	
— AH algorithm(s)	
— data structure (2-D array and/or array of records)	
— classes (object-oriented only)	
— all end-user requirements met	
— all functional requirements met	
— input validation	
♦ screenshot evidence to show before and after execution of AH algorithm	
Implementation of integration with either:	**Complete**
♦ DDD, including evidence of:	
— database structure	
— initial values in tables	
— connection code	
— code for query/queries	
— results of SELECT query/queries	
— tables updated by INSERT, DELETE, UPDATE query/queries	
OR	
♦ WDD, including evidence of:	
— connection code	
— working integration with web code	
Implementation of user interface, including:	**Complete**
♦ screenshot evidence of:	
— input screens	
— output showing results of processing	
— gameplay (if appropriate)	
Description of new skills and/or knowledge, including a description of:	**Complete**
♦ each new skill and/or knowledge	
♦ research required to acquire new skills and/or knowledge	
♦ the use made of new skills and/or knowledge in implementation stage	

Log of ongoing testing that includes:	Complete
♦ description of all tests performed during implementation	
♦ description of details of non-trivial errors or problems encountered, including:	
— description of each issue	
— references used to resolve each issue (if appropriate)	

Testing

Comprehensive test plan that indicates:	Complete
♦ testing of:	
— all functional requirements	
— all end-user requirements	
— input validation	
— integration	
♦ the planned tests to be used	
♦ characteristics of personas	
♦ test cases to be completed by the personas	
Requirements testing	**Complete**
♦ Screenshot evidence of each test listed in the plan.	
Results of the test cases	**Complete**
♦ Description of results of testing with the persona and test cases.	

Evaluation

Fitness for purpose	Complete
♦ Description of how closely the solution matches each of the functional and end-user requirements stated in the 'requirements specification'.	
♦ Discussion of the results of testing.	
Evaluation	**Complete**
♦ Description of maintainability of the solution that refers to ease of future maintenance.	
♦ Description of robustness of the solution.	

Evidence checklist for DDD projects

Analysis

Description of the problem that includes:	Complete
♦ outline of the problem:	
— DDD Advanced Higher (AH) concepts	
— integration with SDD/WDD	
♦ constraints	
UML use case diagram that defines:	**Complete**
♦ actors	
♦ use cases	
♦ relationships	
Requirements specification that includes:	**Complete**
♦ end-user requirements	
♦ functional requirements for DDD	
♦ functional requirements for integration with SDD/WDD	
♦ validation of inputs	
Project plan for each stage that includes:	**Complete**
♦ tasks for each stage	
♦ resources required for implementation	
♦ estimate of timings	

Design

Design of Advanced Higher concepts that includes:	Complete
♦ data dictionary with appropriate SQL data types and input validation	
♦ entity-relationship diagram	
♦ design of queries	
Design of integration with either:	**Complete**
♦ SDD, including pseudocode or structure diagram showing:	
— connection with database	
— execution of queries	
— entry of data from the user	
— formatting of output	
OR	
♦ WDD, including pseudocode or structure diagram showing:	
— connection with webpage/site	
— execution of queries	
— entry of data from the user	
— formatting of output	

User interface design that includes:	Complete
♦ wireframe designs of:	
— input forms showing input validation	
— buttons and menu options showing underlying processes	
— output screens	

Implementation

Implementation of Advanced Higher concepts that includes:	Complete
♦ evidence of implemented SQL code for:	
— implemented tables	
— all queries	
♦ evidence of implemented code for:	
— all functional requirements	
— all end-user requirements	
— input validation	
♦ screenshot evidence showing:	
— initial values shown in tables	
— tables updated by INSERT, DELETE, UPDATE queries	
— results from SELECT queries	
Implementation of integration with either:	**Complete**
♦ SDD, including evidence of implemented code showing:	
— connection with database	
— execution of queries	
— input received from user and/or formatting of output	
OR	
♦ WDD, including evidence of implemented code:	
— connection with database	
— execution of queries	
— input received from user and/or formatting of output	
Implementation of user interface, including:	**Complete**
♦ screenshot evidence of:	
— input screens	
— output showing results of processing	
Description of new skills and/or knowledge, including a description of:	**Complete**
♦ each new skill and/or knowledge	
♦ research required to acquire new skills and/or knowledge	
♦ the use made of new skills and/or knowledge in implementation stage	

Log of ongoing testing that includes:	Complete
♦ description of all tests performed during implementation	
♦ description of details of non-trivial errors or problems encountered, including:	
— description of each issue	
— references used to resolve each issue (if appropriate)	

Testing

Comprehensive test plan that indicates:	Complete
♦ testing of:	
— all functional requirements	
— all end-user requirements	
— input validation	
— integration	
♦ the planned tests to be used	
♦ characteristics of personas	
♦ test cases to be completed by the personas	
Requirements testing	**Complete**
♦ Screenshot evidence of each test listed in the plan.	
Results of the test cases	**Complete**
♦ Description of results of testing with the persona and test cases.	

Evaluation

Fitness for purpose	Complete
♦ Description of how closely the solution matches each of the functional and end-user requirements stated in the 'requirements specification'.	
♦ Discussion of the results of testing.	
Evaluation	**Complete**
♦ Description of maintainability of the solution that refers to ease of future maintenance.	
♦ Description of robustness of the solution.	

Evidence checklist for WDD projects

Analysis

Description of the problem that includes:	Complete
♦ outline of the problem:	
— WDD Advanced Higher (AH) concepts	
— integration with DDD/SDD	
♦ constraints	
UML use case diagram that defines:	**Complete**
♦ actors	
♦ use cases	
♦ relationships	
Requirements specification that includes:	**Complete**
♦ end-user requirements	
♦ functional requirements for WDD	
♦ functional requirements for integration with DDD/SDD	
♦ validation of inputs	
Project plan for each stage that includes:	**Complete**
♦ tasks for each stage	
♦ resources required for implementation	
♦ estimate of timings	

Design

Design of Advanced Higher concepts that includes:	Complete
♦ pseudocode showing:	
— processing of form data	
— assignment of variables	
♦ site navigation structure	
♦ use of session variables (in pseudocode or site navigation structure)	
♦ media queries	
♦ input validation	
Design of integration with either:	**Complete**
♦ DDD, including:	
— data dictionary	
— entity-relationship diagram (if two or more tables)	
— connection to database	
— design of queries	
OR	
♦ SDD, including:	
— connection with program	
— AH data structure or algorithm	

User interface design that includes:	Complete
Wireframe designs of:	
♦ input forms showing input validation	
♦ buttons and menu options showing underlying processes	
♦ output screens	

Implementation

Implementation of Advanced Higher concepts that includes:	Complete
♦ evidence of implemented:	
— HTML code for forms	
— code for form processing and assignment of variables	
— external CSS file with link in HTML/PHP pages	
— media queries in CSS file	
— session variables	
— functional requirements	
— end-user requirements	
— input validation	
Implementation of integration with either:	**Complete**
♦ DDD, including evidence of:	
— database structure	
— initial values in tables	
— connection code	
— code for query/queries	
— results of SELECT query/queries	
— tables updated by INSERT, DELETE, UPDATE query/queries	
OR	
♦ SDD, including evidence of implemented code showing:	
— connection with program	
— AH data structure or algorithm	
— input received from user and/or format output	
Implementation of user interface, including:	**Complete**
♦ screenshot evidence of:	
— input screens	
— output showing results of processing	
Description of new skills and/or knowledge, including a description of:	**Complete**
♦ each new skill and/or knowledge	
♦ research required to acquire new skills and/or knowledge	
♦ the use made of new skills and/or knowledge in implementation stage	

Log of ongoing testing that includes:	Complete
♦ description of all tests performed during implementation	
♦ description of details of non-trivial errors or problems encountered, including:	
— description of each issue	
— references used to resolve each issue (if appropriate)	

Testing

Comprehensive test plan that indicates:	Complete
♦ testing of:	
— all functional requirements	
— all end-user requirements	
— input validation	
— integration	
♦ the planned tests to be used	
♦ characteristics of personas	
♦ test cases to be completed by the personas	
Requirements testing	**Complete**
♦ Screenshot evidence of each test listed in the plan.	
Results of the test cases	**Complete**
♦ Description of results of testing with the persona and test cases.	

Evaluation

Fitness for purpose	Complete
♦ Description of how closely the solution matches each of the functional and end-user requirements stated in the 'requirements specification'.	
♦ Discussion of the results of testing.	
Evaluation	**Complete**
♦ Description of maintainability of the solution that refers to ease of future maintenance.	
♦ Description of robustness of the solution.	

Administrative information

Published: September 2024 (version 2.0)

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

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.