Advanced Higher Computing Science

Software design and development project workshop materials

This edition: December 2024 (version 1.0)

Introduction

This document is for teachers and lecturers and/or Advanced Higher Computing Science candidates.

This document contains workshop activities originally devised for an Understanding Standards event held in 2023. The workshop activities focused on the problem description, requirements specification, design, implementation, testing and evaluation of an example Advanced Higher software design and development project.

Workshop activity 1: problem descriptions

Read the following problem description.

Example 1

My project aims to program a multiple-choice general knowledge quiz for the Clydeside Quiz League. The 36 possible quiz questions, potential answers and the correct answers will all be stored in an SQL database table. 10 questions will be selected and the users will be shown the questions one at a time and given 5 seconds to answer each question. The answer will be validated and checked and 2 points added to the score if a correct answer was selected within the time allowed, but 4 points will be removed if the answer was entered too late. When the user has completed the quiz, they will be given their total score. This score and the user's username can be added to another database table that will be used to store a list of scores of the Quiz League players. Users will be shown the complete list of scores in descending order of score and they can then choose to try the quiz again or to quit the program.

Things to consider:

- Is it clear from example 1 which technologies are integrated in the solution?
- Advanced Higher projects must validate all inputs. Consider whether this requirement has been considered in example 1.
- Example 1 doesn’t provide sufficient detail of the mandatory Advanced Higher concepts for a software development project. Consider its shortfalls.
- Consider whether the intended use of a database meets the requirements of an Advanced Higher project.

Example 2

Example 2 meets the standard required for the Advanced Higher problem description.

My project aims to create a general knowledge quiz for the Clydeside Quiz League. The quiz will be developed as a procedural program that integrates with a database.

- the program will connect to the database and use a SQL query to select the quiz questions and answers stored in a database table and store them in an array of records so that they can be processed by the program
- the program will select and display random questions from those available and display them one at a time
- the answers entered will be validated
- the program will check whether the answer is correct
- once the quiz is finished, the program will display the user's score and the user can choose to submit their score to the league
- the user will be asked to enter their username, which will be validated
- the program will connect to the database and use a SQL query to add the username and score to the database table
- the program will connect to the database and use a SQL query to select the stored usernames and scores and store them in an array of records
- the bubble sort algorithm will be used to arrange the usernames and scores into descending order of score
- all inputs to the program will be validated and appropriate error messages will be displayed when invalid values are entered

Workshop activity 2: requirements specification

Example 3

Example 3 goes beyond the standard required for the Advanced Higher problem description.

My project aims to create a general knowledge quiz for the Clydeside Quiz League. The quiz will be developed as a procedural program that integrates with a database.

- the program will connect to the database and use a SQL query to select the 36 quiz questions, possible answers and correct answers stored in a database table and store them in an array of records so that they can be processed by the program
- the program will select and display 10 random questions from the 36 available and display them one at a time
- a timer will be used to give users 5 seconds to respond to each question
- the answers entered will be validated to make sure that only a, b, c, d, A, B, C and D are entered
- the program will check whether the answer is correct and award 2 points for a correct response received within the time allowed but will remove 4 points for taking too long to respond to a question
- once the quiz is finished, the program will display the user's score and the user can choose to submit their score to the league
- the user will be asked to enter their username so that the username and score can be saved to a separate database table
- the username will be validated and must contain between 5 and 12 characters
- the program will connect to the database and use a SQL query to add the username and score to the database table
- the program will connect to the database and use a SQL query to select the usernames and scores of the Clydeside Quiz League players and store them in a second array of records
- the bubble sort algorithm will be used to arrange the usernames and scores into descending order of score
- the usernames and scores will be displayed in descending order
- users can then try the quiz again or quit the program
- the program will terminate only when the user chooses to quit the program
- all inputs to the program will be validated and appropriate error messages will be displayed when invalid values are entered

Use example 3 to identify the essential end-user and functional requirements for this project.

Completed requirements specification for example 3

Since example 3 exceeds the requirements of the Advanced Higher problem description, the number of end-user requirements below also exceeds the Advanced Higher project requirements.

End-user requirements

Requirement number	The end users of the solution should be able to:
EU 1	Select to complete the quiz or quit program
EU 2	View one question at a time
EU 3	Enter answer to each individual question
EU 4	View final quiz score
EU 5	Choose whether to submit final score to the league
EU 6	Enter username
EU 7	View names and scores of Clydeside Quiz League players
EU 8	Select to complete the quiz again or quit program

Functional requirements

Requirement number	The solution is required to:
FR 1	Allow user to either play quiz or quit program
FR 2	Store details of 36 quiz questions and answers in one database table with details of users stored in a separate database table
FR 3	Connect to database to execute SQL queries
FR 4	Execute SQL query to select 36 questions and answers stored in the database
FR 5	Store questions and answers in an array of records
FR 6	Select 10 random questions from the 36 available
FR 7	Display one question at a time
FR 8	Use a timer to give user 5 seconds to enter their answer
FR 9	Validate response received – this will be a required value and only a, b, c, d, A, B, C and D are acceptable
FR 10	Check answer entered and update score
FR 11	Display user's final score and ask whether user wants to save score to league
FR 12	Enter username and validate to make sure it contains between 5 and 12 characters
FR 13	Execute SQL query to add username and score to existing database table

Requirement number	The solution is required to:
FR 14	Execute SQL query to select all usernames and scores stored in the database
FR 15	Store usernames and scores in an array of records
FR 16	Use the bubble sort algorithm to arrange usernames and scores into descending order of score
FR 17	Display top ten usernames and scores of Clydeside Quiz League
FR 18	Allow user to either play quiz again or quit program
FR 19	Validate all inputs to the program and display appropriate error messages when invalid input values are entered

The requirements specification forms the ‘golden threads’ that run through the project development.

Consider the importance of having clearly defined end-user and functional requirements in the requirements specification:

- when creating the project plan
- at the design stage of the development
- during the implementation of the solution
- when creating the final test plan
- at the evaluation stage of the development

Workshop activity 3: discussion points

Design

Consider the design tasks that must be completed for this project.

Complete list of design tasks

Design task 1: design of Advanced Higher concepts

- design the data structure that will be used to store the quiz questions and answers within the program
- design the data structure that will be used to store the usernames and scores within the program
- create the top-level design and data flow
- refine top-level design to show details of any programmed requirements — including the sort algorithm needed to arrange the usernames and scores in descending order of score and the intended input validation

Design task 2: design of integration

- create a data dictionary for the table that will be used to store the usernames and scores
- create a data dictionary for the table that will be used to store the quiz questions and answers
- create an ERD to show that the tables are unrelated
- design the connection between the program and the database (if not already indicated in the top-level design or refinements)
- design each query required for the solution

Design task 3: user-interface design

- complete the user-interface design for each input screen, showing input validation and underlying processes behind any buttons or menu options
- complete the user-interface design for each output screen

Design task 4: design matches requirements

- check that a design that meets all requirements listed in the requirements specification has been submitted

Implementation

Consider the solution being developed.

- What evidence should be included to demonstrate that the implemented sort algorithm works correctly.

Testing: comprehensive test plan and comprehensive testing

Read the complete requirements specification.

- Discuss what tests the candidate should include in the test plan to demonstrate that all planned input validation has been implemented correctly.
- Discuss what evidence the candidate should include to demonstrate that the implemented input validation traps errors as planned.
- Discuss how evidence used for implementation of Advanced Higher concepts and implementation of integration could be reused at the testing stage.

Evaluation: evaluation of maintainability and robustness

Consider the solution being developed.

- Advanced Higher candidates are expected to refer to specific types of future maintenance. Consider features that could possibly aid or hinder any such maintenance.
- Consider whether it would be appropriate to be critical of the limited input validation that had been included in their solution.