THE KING'S COLLEGE - YEAR 12 COMPUTER SCIENCE GENERAL

Task 6 Student Task Sheet: Study Session Helper Program

Task 6 Study Session Helper Program - Unit 4 - 20%

Task 6 Student Task Sheet: Study Session Helper Program

Assessment Details

Item	Details
Course	Year 12 Computer Science General
Unit	Unit 4
Assessment type	Project
Weighting	20%
Raw mark total	50 marks
Timing	Term 3 Weeks 4-8
Final submission window	Monday 7 September to Friday 11 September 2026

Scenario

Year 12 students often have several assessments to prepare for at the same time. They need a small digital tool that helps them choose a study focus, practise a small set of questions and receive a useful summary of what to do next.

You are to design, build, test and evaluate a **Study Session Helper Program** for a Year 12 student.

Task

Create a working coded program that:

1. asks the user for study session information
2. checks at least one input using validation or error handling
3. uses selection to give different advice or feedback
4. uses iteration so the user can repeat questions, topics or menu actions
5. records progress using variables, counters, scores or totals
6. produces a useful final output summary for the user
7. is documented, tested and evaluated using the Software Development Cycle

Your program may be built as a text-based program, web page, app, Scratch project or another teacher-approved programming environment. The interface can be simple, but it must be clear enough for the intended user to complete a study session.

Required Program Features

Your final program must include:

- **User input:** asks for the student's name or session label, study topic and available study time.
- **Confidence or priority input:** asks the user for a confidence rating, priority rating or similar value that affects the output.
- **Practice questions or prompts:** includes at least three study questions, flashcards or practice prompts.
- **Selection:** uses at least one meaningful `if`, decision, branch or equivalent structure that changes the output.
- **Iteration:** uses at least one loop or repeated action, such as repeated quiz questions, repeated menu use or repeated topic entry.
- **Variables and data types:** stores and updates values such as topic, time, confidence, score, attempts or recommendation.
- **Validation or error handling:** checks at least one important input, such as blank topic, impossible time, invalid menu choice or rating outside the accepted range.
- **Output summary:** displays a useful study recommendation, score, progress summary or next-step advice.
- **Interface:** uses clear prompts, labels, buttons, layout or menu choices that suit a Year 12 student.

Required Evidence And Mark Allocation

The task is marked out of **50 marks**.

Criterion 1: Problem Definition, User Requirements And Scope

Description	Marks
Evidence document identifies the target user as a Year 12 student preparing for assessment or a clearly related Year 12 study user.	1
Evidence document states a clear study-session problem or need linked to the supplied stimulus.	1
Evidence document describes at least four functional requirements linked to the Study Session Helper Program.	1-2
Evidence document identifies relevant constraints, such as time, device, programming environment, user ability, available data or classroom conditions.	1
Evidence document includes testable success criteria that can be used in the final evaluation.	1
Subtotal	**6**

Criterion 2: Product Design And SDC Planning

Description	Marks
Evidence document describes how the Software Development Cycle stages will be used for this project.	1
Evidence document provides a user flow or step-by-step interaction plan for the study session.	1
Evidence document provides a wireframe, prototype, screen sketch or text-interface plan showing the main user interaction.	1-2
Evidence document provides an input-process-output plan connected to the required program features.	1
Evidence document provides a data or variable plan with suitable names and intended data types.	1
Evidence document explains how user needs influenced the interface or interaction design.	1
Evidence document keeps the planned Study Session Helper Program scope realistic for the assessment window.	1
Subtotal	**8**

Criterion 3: Algorithm Design

Description	Marks
Evidence document provides pseudocode and/or a flow chart that represents the main Study Session Helper Program logic.	1-2
Algorithm evidence shows sequence clearly.	1
Algorithm evidence shows at least one meaningful selection structure that affects advice, feedback, scoring or output.	1
Algorithm evidence shows at least one meaningful iteration structure for repeated questions, menu use, topic entry or attempts.	1
Algorithm evidence includes validation or error-handling logic.	1
Evidence document provides a trace table for a meaningful part of the algorithm.	1
Algorithm evidence is clear and consistent with the final code.	1
Subtotal	**8**

Criterion 4: Program Development

Description	Marks
Submitted product runs correctly or has authenticated evidence of running.	1-2
Submitted product is recognisably a Study Session Helper Program and addresses the supplied stimulus.	1-2
Submitted product uses input, processing and output clearly.	1-2
Submitted product uses meaningful variable names and suitable data types.	1-2
Submitted product uses sequence and readable program structure.	1
Submitted product uses at least one meaningful selection structure in working code.	1-2
Submitted product uses at least one meaningful iteration structure in working code.	1-2
Submitted product includes validation, error handling or user-friendly handling of unexpected input.	1
Subtotal	**14**

Criterion 5: Testing, Debugging, Validation And Iteration

Description	Marks
Evidence document includes testing with normal data.	1
Evidence document includes testing with boundary data where relevant, such as minimum/maximum time, lowest/highest confidence rating or final quiz question.	1
Evidence document includes testing with invalid or unexpected data where relevant, such as blank topic, impossible time or rating outside the accepted range.	1
Testing evidence records expected and actual results.	1-2
Evidence document identifies at least one error, issue or design problem and explains how it was addressed.	1
Evidence document includes validation evidence that checks the product against user needs, success criteria or user feedback.	1
Evidence document shows at least one improvement or justified change.	1
Subtotal	**8**

Criterion 6: Evaluation, Documentation And Submission Quality

Description	Marks
Final evaluation judges the product against success criteria or requirements.	1-2
Final evaluation identifies specific strengths and limitations of the final product.	1
Final evaluation recommends realistic future improvements.	1
Final submission is organised and named clearly so it can be marked.	1
Final submission acknowledges external help, assets, tutorials, examples, code snippets or AI support where used.	1
Subtotal	**6**
Total	**50**

What To Submit

Submit one final evidence package through the LMS. It must include:

1. your working product file, project folder or product link
2. an evidence document with screenshots and explanations
3. planning and design evidence
4. pseudocode and/or a flow chart
5. a trace table for one important part of the algorithm
6. a testing log
7. validation and iteration evidence
8. final evaluation
9. acknowledgements for external help, assets, tutorials, code snippets or AI support

Use this file name pattern:

`CSCGT_Task6_2026_Surname_FirstName`

Conditions

- Work will be completed across supervised class time and approved homework time.
- Authentication checkpoints must be completed.
- You may use class notes and teacher-provided examples.
- You may use a teacher-approved programming environment.
- You may use Figma, Keynote, paper sketches or another approved tool for prototype evidence.
- Any external help, assets, code snippets, tutorials or AI support must be acknowledged.
- You must be able to explain your own code and design decisions.

Important Scope Rules

A strong Task 6 program is small, purposeful, working, tested and well explained. It does not need to look like a commercial app.

Avoid:

- building a full commercial app
- spending most of the time on colours or visual polish
- submitting only a prototype without working program logic

- using code you cannot explain
- leaving out selection, iteration or validation
- changing the product purpose without teacher approval

Before You Start

Begin with the user problem and required program features. Your design decisions, code, testing and evaluation should all connect back to the Study Session Helper Program scenario.
