App Plan

Exported on: 27/11/2025, 2:06:23 pm

This is a highly structured and rigorous project plan for your VCE Software Development School-assessed Task (SAT). Focusing on managing exam data, grading questions, and setting study goals ensures you cover all the mandatory technical requirements across Unit 3 and Unit 4.

The project structure must follow the four stages of the Problem-Solving Methodology (PSM): Analysis, Design, Development, and Evaluation.

Final App Plan: Multi-Subject Exam Study Bot and Goal Tracker

1. Project Management (Unit 3 Outcome 2 & Unit 4 Outcome 1)

The SAT requires you to document and manage the entire process using project management techniques.

• **Project Plan (Gantt Chart):** You must create a Project Plan that outlines tasks, sequencing, time allocation, dependencies, milestones, and the critical path, covering both Unit 3 Analysis/Design and Unit 4 Development/Evaluation.

• **Monitoring:** The plan must be monitored, modified, and annotated throughout Unit 3 and Unit 4 to explain progress or changes.

---

2. Analysis Stage (Unit 3 Outcome 2)

This stage involves determining what the application must do and defining its limitations. The findings are compiled into a **Software Requirements Specification (SRS)**.

SRS Component	Application Focus	VCE Requirement Support
Functional Requirements	The ability to read question data from external files, generate quizzes, accept user answers, grade answers (the complex process), store graded results, and track goals against actual progress.	Describe what the software should do, including specific details like data manipulation and validation.
Non-functional Requirements	**Usability** (clear interface for multiple subjects/goals), **Portability** (working across different desktop environments), and **Maintainability** (clear code for future changes).	Describe the quality attributes the solution must possess.
Constraints	**Economic/Time:** Must be completed by the SAT deadlines. **Technical:** Specific programming language features (must be OOP), file type limitations (TXT, CSV, XML for question data). **Legal:** Protecting the privacy of user study data (e.g., student results).	These are limitations that must be considered when designing the solution.
Scope	Define exactly which subjects are included, which question formats are supported (e.g., multiple choice, simple numerical input), and explicitly state what the app *will not* do (e.g., it will	Identify what will and will not be addressed by the solution.

	not automatically download exam papers or mark essay questions).	

Analytical Tools: You must use diagrams to depict the interactions.

• **Context Diagram (Level 0):** Shows the system boundary (The Study Bot) and external entities (e.g., The Student, The Question Data File, The Goals File).

• **Data Flow Diagram (DFD) (Level 1):** Maps the major processes (e.g., Grade Submission, Store Goal Progress) and data flows within the system.

• **Use Case Diagram:** Shows the Actor (The Student/User) interacting with key functions (Use Cases) such as 'Log In', 'Generate Quiz', 'Set Weekly Goal', and 'View Progress'.

---

3. Design Stage (Unit 3 Outcome 2)

This stage determines *how* the solution will function and appear, resulting in detailed designs.

1. **Ideation and Evaluation Criteria:** Generate and document two or three design ideas (e.g., using sketches or annotations) for the interface layout and core functionality. Develop **evaluation criteria** to measure the efficiency and effectiveness of these ideas against your SRS requirements (e.g., usability, clarity of goal tracking display).

2. **Detailed Designs (Preferred Solution):** Fully develop the chosen design using appropriate tools.

◦ **Data Dictionary:** Defines all variables, constants, and complex structures (like records or arrays) used to store questions, student details, and goal tracking data, applying **naming conventions** (e.g., Hungarian notation or camel casing).

◦ **Data Structures:** Must incorporate complex data structures, typically **records** (or classes/objects), to store related data, such as `QuestionID`, `Subject`, `Answer`, and `Difficulty`. Records are better than a series of one-dimensional arrays for related fields.

◦ **Mock-ups:** Detailed visual designs of key screens (e.g., the Quiz Generation screen, the Weekly Goal Tracker dashboard, the Login screen), annotated to show functionality and adherence to **design principles** (alignment, contrast, usability).

◦ **Pseudocode:** Detailed, English-like instructions for complex processes. This must include logic for:

▪ **Grading Functionality:** Comparison and calculation logic (using selection statements like `IF...ELSEIF`).

▪ **Sorting/Searching:** Algorithms for managing data (e.g., sorting past scores by mark, or searching the question bank by subject).

---

4. Development and Testing (Unit 4 Outcome 1)

This phase involves coding the solution and proving it works as intended.

Activity	Application Focus	VCE Requirement Support
Programming	Develop the multi-subject application using an **Object-Oriented Programming (OOP)** language. You must use control structures (sequence, selection, iteration).	Develop the solution to meet specifications using appropriate

		features of an OOP language and algorithms.
Data Handling	Store question data and student goal progress using **data sources** (TXT, CSV, or XML files) and appropriate **data structures** (records/objects).	Use and apply appropriate data types, data structures, and data sources.
Validation	Implement validation techniques for inputs (e.g., the answer field, the weekly goal number). Must include: **Existence checking** (an answer was entered), **Type checking** (numeric input is numeric), and **Range checking** (a goal quantity is within reasonable limits).	Use validation techniques to check data entry for reasonableness and completeness.
Internal Documentation	Add detailed **comments** within the code to explain variables (`fltQuizScore`), functions (the grading algorithm), and class structures, adhering to naming conventions.	Write internal documentation to support the functioning, maintenance, and upgrading of the solution.
Security	Implement **User Authentication** (logins) and define a procedure for **Backups** (full, incremental, or differential) to protect goal/score history.	Security of the solution and of the data/information is a required element.
Alpha Testing (Debugging)	Systematically test all modules during coding, especially the grading logic and goal calculation. Document results using **testing tables** that compare expected versus actual output, explicitly checking **boundary values** (e.g., testing the lowest and highest score thresholds).	Conduct debugging techniques to ensure solutions meet requirements. Test cases must compare expected and actual output.
Beta Testing	Prepare and conduct a testing plan with **at least two potential users** (e.g., classmates acting as students). Record observations of user interaction with the quiz generator and goal tracker. Based on results, **recommend modifications** to address identified appearance or behavior issues.	Design, conduct, and document beta tests with potential users, capture results, and recommend modifications.

---

5. Evaluation Stage (Unit 4 Outcome 1)

This final stage assesses the success of the solution and the process used.

• **Solution Evaluation:** Use the criteria developed in Unit 3 to evaluate the final working solution's **efficiency** (e.g., calculation speed, code complexity/clarity) and **effectiveness** (how well it met the functional requirements, such as successfully tracking goals and generating quizzes).

• **Project Plan Assessment:** Assess the effectiveness of the **Project Plan (Gantt chart)** used throughout the project, referencing the annotations and modifications made. Discuss the impact these changes had on the completion of the project. This is typically presented as a written report or annotated visual plan.