1. **Agile Development Process Plan**

This project adopts the Agile methodology, specifically using a Scrum-based approach adapted for solo development. Agile is a modern software development method that emphasizes flexibility, collaboration, and iterative progress. Unlike traditional models like Waterfall, Agile does not require all requirements to be defined at the beginning. Instead, it encourages the software to evolve through continuous feedback, small deliverables, and adaptive planning.

Scrum, as an Agile framework, structures development into short cycles called sprints, each focused on delivering specific, testable features. For this project, each sprint lasts approximately 2–3 days and is designed to handle one functional module at a time.

Agile is particularly well-suited to this seat-booking system because:

* The requirements are revealed gradually: For example, new functions such as booking references and database integration appear in later stages of the task. Agile allows me to build a basic version first and progressively enhance it.
* The user interface needs refinement: Since the system uses a menu-driven interface, Agile allows me to iteratively test and adjust it based on user feedback or tutor input.
* The project can be divided into manageable stages: Key features like seat checking, booking, and reference generation can each be developed, tested, and refined in separate sprints, reducing risk and maintaining progress.

In summary, Agile enables me to develop this system incrementally and flexibly, while maintaining high software quality through early feedback, constant testing, and rapid error correction.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Goal** |  |
| **1** | Day 1–3 | Set up seat matrix and main menu | Display layout with 'F', 'R', 'X', 'S' statuses |
| **2** | Day 4–6 | Implement core booking functions | Working options: check, book, free, display seats |
| **3** | Day 7–9 | Add extended feature (e.g., search seat) | Extra menu option for advanced user task |
| **4** | Day 10–12 | Add booking reference generator | 8-char alphanumeric unique ID logic added |
| **5** | Day 13–15 | Connect SQLite database | Store/retrieve customer data on booking/cancel |
| **6** | Day 16–17 | Final test and documentation | All bugs fixed, project committed to GitHub |

Reason:

1. Early Detection and Correction of Errors: After each cycle, I will test new features. If errors are found, I can fix them immediately before progressing, ensuring software quality.
2. Adaptability to Requirement Changes: Agile allows me to integrate future requirements (e.g., database integration, booking references) smoothly without rebuilding the system from scratch.
3. Self-Managed Iterations Improve Focus: Since I am the only developer, I can break down work into manageable cycles with clear goals and outcomes, reducing overwhelm and improving time management.

Features to Implement:

* Display seat status (F: Free, R: Reserved, X: Aisle, S: Storage)
* Check the availability of a specific seat
* Book a seat (only if available)
* Cancel a booking (mark as F and remove data)
* Display current booking status (visually or in table format)
* Generate a unique 8-character alphanumeric booking reference
* Store passenger data (passport number, name, seat)
* Integrate SQLite database for data storage and retrieval
* Additional functionality (e.g., search booking reference)
* Push code to GitHub and manage version control

Every Cycle：

* **Planning Phase**：
  + Define the target functionality for the cycle
  + Identify any required changes in data structures or logic
* **Development Phase**
  + Write and comment the code for new functions
  + Follow modular design (one function per feature if possible)
* **Testing Phase**
  + Test with both valid and invalid input
  + Ensure each menu option works as expected
* **Debugging / Refactoring**
  + Fix any errors immediately, redesign logic if needed
  + Clean and organize code, use consistent naming
* **Version Control (GitHub Commit)**
  + One commit at the end of each sprint

Bug Handling and Fix Strategy:

* Every new function will be tested immediately after development
* Testing will include edge cases and invalid inputs (e.g., booking an occupied or non-existent seat)
* Bugs will be fixed as soon as discovered and documented
* No bug fixes will be postponed to prevent blocking later sprints

Responding to Changing Requirements:

If new requirements emerge such as changes to the booking reference format or the addition of a search function, I will:

* Add the new requirement to the product backlog
* Schedule its development in an upcoming sprint
* Evaluate whether any existing structures or menus need revision
* This ensures I can make enhancements without rewriting the entire system—only the relevant module is updated.

1. **Diagram**

