**Software Requirements Specification (SRS)**

**Study Buddy App**

**1. Introduction**

**1.1 Purpose**  
The Study Buddy App helps students collaborate by connecting them based on courses and availability. This version is targeted for a C++ implementation, either as a command-line interface or with a lightweight GUI using a C++ framework.

**1.2 Scope**  
A standalone C++ application where students create profiles, manage availability, find matches, schedule, and confirm study sessions.

**2. Overall Description**

**2.1 Product Perspective**  
A C++ desktop application (CLI or simple GUI) that stores data locally (e.g., file system or lightweight database) and handles user interaction through text menus or GUI elements.

**2.2 Product Functions**

* Create and update student profiles.
* Manage enrolled courses and availability time slots.
* Search classmates by course.
* Suggest study partners based on overlapping availability.
* Schedule and confirm study sessions.

**2.3 User Classes and Characteristics**

* Students who are comfortable with desktop apps or CLI.

**2.4 Operating Environment**

* Cross-platform (Windows, macOS, Linux).
* Requires a C++ compiler (e.g., g++, MSVC).
* Optional GUI libraries: Qt, wxWidgets, or ncurses for CLI enhancements.

**2.5 Design and Implementation Constraints**

* Use standard C++17 or newer for modern features (e.g., std::vector, std::optional, smart pointers).
* Data persistence via file I/O (JSON/XML) or embedded DB (SQLite with C++ wrapper).
* Efficient memory management and exception safety.
* Modular codebase with clear separation of concerns (e.g., classes for Student, Session, Course).

**3. Specific Requirements**

**3.1 Functional Requirements**

| **ID** | **Requirement Description** | **Priority** |
| --- | --- | --- |
| FR1 | Provide a Student class with attributes: name, email, courses (vector), availability (vector). | High |
| FR2 | Provide methods to add/remove availability slots (e.g., addAvailability(), removeAvailability()). | High |
| FR3 | Implement search functionality to find students enrolled in the same course. | High |
| FR4 | Suggest study partners based on shared courses and intersecting availability using appropriate data structures. | Medium |
| FR5 | Allow scheduling of sessions by creating Session objects involving multiple Student instances. | High |
| FR6 | Implement confirmation mechanism via session status updates (e.g., Pending, Confirmed). | High |
| FR7 | Handle conflicts and overlapping sessions gracefully. | High |
| FR8 | Input/output operations to save/load student data and sessions using file streams or SQLite. | High |