Software Requirements Specification (SRS)

Study Buddy Scheduling App for Clemson Students

---

1. Introduction

1.1 Purpose

The Study Buddy app is designed to help Clemson University students connect with classmates to schedule study sessions. It enables students to create profiles with their enrolled courses, share availability, find study partners, and coordinate meetings.

1.2 Scope

This application will be developed in Java and support two user interfaces:

- **Command-Line Interface (CLI)** for quick interaction and basic use
- **Graphical User Interface (GUI)** for a more user-friendly experience

Users will be able to:

- Create and edit personal profiles (name, email, courses)
- Add or remove their available time slots for studying
- Search for and view classmates enrolled in the same courses
- Schedule study sessions by matching shared courses and overlapping availability
- Save and load data locally for persistent use

---

2. Overall Description

2.1 User Characteristics

- Clemson students familiar with basic computer use.
- No user authentication is required in this version.

2.2 Operating Environment

- Java 8 or later installed on Windows, macOS, or Linux
- CLI runs in a terminal or command prompt
- GUI runs as a desktop application (using JavaFX or Swing)

2.3 Design Constraints

- Data storage will be via local JSON files using a Java JSON library such as Gson or Jackson.
- Time slots will be stored as simple strings without complex date-time parsing.

---

3. Functional Requirements

3.1 Profile Management

- Users can create or update a profile including:
    - Name
    - Email
    - List of enrolled courses (e.g., “CPSC1010”, “MATH1060”)

3.2 Availability Management

- Users can add or remove availability time slots (e.g., “Mon 14:00-16:00”).

3.3 Study Buddy Matching

- The system will display classmates who share at least one enrolled course.
- The system will suggest study partners based on overlapping availability.

3.4 Session Scheduling

- Users can schedule study sessions by selecting a study buddy, course, and mutually available time slot.
- Sessions are confirmed and saved for both participants.

3.5 Data Persistence

- User profiles, availability, and sessions are saved to and loaded from JSON files.
- Data is updated whenever changes occur or when the application closes.

---

4. Non-Functional Requirements

4.1 Usability

- CLI menu should be intuitive and straightforward.

- GUI should provide easy navigation between profile, availability, matches, and scheduling.

4.2 Performance

- Data loading and saving operations should complete within 2 seconds.
- Searching matches should be efficient for up to a few hundred users.

4.3 Portability

- The app will run on any system with Java installed.

4.4 Maintainability

- The code should follow a modular design with separation between data (model), user interface (view), and application logic (controller).

---

5. System Features and UI

5.1 CLI Features

- Text-based menus for each function (profile management, availability, matches, scheduling).
- Inputs read via Scanner or similar class.

5.2 GUI Features

- A main window with tabs or panels for:
  - Profile Editing
  - Availability Management
  - Viewing Matches
  - Scheduling Sessions
- Use JavaFX or Swing for GUI components.
- Forms for data entry and tables/lists for displaying matches and sessions.

---

6. Data Model

6.1 User Class

- Attributes: name (String), email (String), courses (List<String>), availability (List<String>), sessions (List<StudySession>)

6.2 StudySession Class

- Attributes: withUserEmail (String), course (String), timeSlot (String)

---

7. Development and Implementation Plan

7.1 Step 1: Core CLI

- Implement profile creation and editing
- Implement availability management
- Implement matching and scheduling logic
- Implement data persistence to JSON

7.2 Step 2: GUI Implementation

- Design basic GUI layout with forms and lists
- Connect GUI actions to backend logic from CLI implementation
- Implement save/load functionality in GUI

---

8. Assumptions and Limitations

- User authentication and security are out of scope.
- No integration with Clemson's official course registration system; users manually input courses.
- Time inputs are simple strings without validation beyond basic format.
- Notifications or messaging features are not included.