# **Software Requirements Specification (SRS)**

**Project:** Study Buddy Scheduling App  
 **Course:** Software Development – Clemson University  
 **Group A (Waterfall)** **Language:** Java (Command-line Application)

## **1. Introduction**

### **1.1 Purpose**

The purpose of this SRS is to formally define the requirements for the Study Buddy Scheduling App, a command-line prototype system designed for Clemson students. The app enables students to create a study profile, register for courses, set availability, and schedule study sessions with classmates. This SRS provides the foundation for design, implementation, and testing under the Waterfall development process. Once approved, no requirements may be changed.

### **1.2 Scope**

The Study Buddy Scheduling App will:

* Allow students to create a profile linked to their enrolled courses.
* Support study session scheduling for **one-on-one, small group (3–5 students), or large group (6+ students)**.
* Provide functionality to **add/remove availability in 15-minute intervals**.
* Recommend study sessions based on students’ **course registrations and overlapping availability**.
* Allow students to **register for sessions** and view who else is attending.
* Enable students to **cancel their registration** if needed.
* When exiting, prompt students to confirm their registrations before closing the application.

The app will be a **Java command-line prototype** with **local file-based storage**. It is not intended for production use but will demonstrate scheduling functionality and process learning.

## **2. Overall Description**

### **2.1 Product Perspective**

The Study Buddy Scheduling App is a stand-alone system. It has no external interfaces or dependencies other than file-based storage. The application will be deployed on local machines and accessed via the Java runtime environment.

### **2.2 Product Functions**

* **Profile Management:** Create and update a profile with enrolled courses and study preferences.
* **Course Entry:** Select courses using Clemson course prefix + course number.
* **Availability Management:** Add and remove availability slots in 15-minute increments.
* **Session Recommendations:** Suggest study sessions based on overlapping availability and enrolled courses.
* **Session Scheduling:** Join or leave study sessions with classmates.
* **Session Viewing:** Display attendees for each scheduled session.
* **Exit Confirmation:** On exit, display the user’s registered sessions and prompt for confirmation.

### **2.3 User Characteristics**

Users are Clemson University students with basic computer literacy. No prior programming or technical expertise is required.

### **2.4 Constraints**

* Command-line interface only (no GUI or web interface).
* Storage is limited to local text files.
* Single-user login at a time (prototype scope).

### **2.5 Assumptions and Dependencies**

* Students provide the correct course prefix and number when entering courses.
* Time slots are entered in valid 24-hour format with 15-minute granularity.
* No real-time notifications; students must open the app to view schedules.

## **3. Specific Requirements**

### **3.1 Functional Requirements**

**FR1. Profile Management**

* FR1.1: The system shall allow a student to create a profile with their name and enrolled courses.
* FR1.2: The system shall allow the student to specify a study preference: one-on-one, small group (3–5), or large group (6+).
* FR1.3: The system shall store profiles persistently in a file.

**FR2. Course Selection**

* FR2.1: The system shall allow students to enter courses by prefix (e.g., CPSC) and number (e.g., 1010).
* FR2.2: The system shall validate prefixes against a predefined list.

**FR3. Availability Management**

* FR3.1: The system shall allow students to add available time slots in 15-minute intervals.
* FR3.2: The system shall allow students to remove previously added availability.

**FR4. Session Scheduling**

* FR4.1: The system shall allow students to view available study sessions by course.
* FR4.2: The system shall allow students to register for one-on-one, small group, or large group sessions.
* FR4.3: The system shall allow students to cancel their registration.
* FR4.4: The system shall display the list of attendees for each session.

**FR5. Match & Recommendation System**

* FR5.1: The system shall suggest study matches based on overlapping availability.
* FR5.2: The system shall recommend study sessions based specifically on the courses the student is registered in.

**FR6. Exit Confirmation**

* FR6.1: When a student attempts to exit the program, the system shall display all of their current session registrations.
* FR6.2: The system shall prompt the student to confirm their registrations before the application closes.

### **3.2 Non-Functional Requirements**

* **NFR1 (Usability):** The system shall provide a text-based command-line menu with clear instructions.
* **NFR2 (Performance):** The system shall process profile creation and session scheduling within 2 seconds.
* **NFR3 (Storage):** The system shall store data in text files readable by the system for future sessions.
* **NFR4 (Reliability):** The system shall not crash during normal use and handle invalid input gracefully.
* **NFR5 (Security):** The system shall require a local login (username/password) for profile access.

## **4. Use Case Scenarios**

**Use Case 1: Create Profile**

* Actor: Student
* Precondition: Students have access to the app.
* Steps:  
  1. The student chooses “Create Profile.”
  2. The student enters name, login credentials, courses, and study preference.
  3. Profile is stored locally.
* Postcondition: Profile is saved in file storage.

**Use Case 2: Add Availability**

* Actor: Student
* Precondition: The student is logged in.
* Steps:  
  1. The student selects “Add Availability.”
  2. The student inputs a time slot (e.g., 14:30–15:30).
  3. The system saves availability.
* Postcondition: Availability is added to profile.

**Use Case 3: Join Study Session**

* Actor: Student
* Precondition: Student is logged in with at least one course and availability entered.
* Steps:  
  1. Student views recommended study sessions.
  2. The student selects a session to join.
  3. The system adds the student to the session attendee list.
* Postcondition: Student is registered for the session.

**Use Case 4: Exit Confirmation**

* Actor: Student
* Precondition: Student has registered for one or more study sessions.
* Steps:  
  1. The student selects “Exit.”
  2. The system displays a list of current session registrations.
  3. The student confirms or cancels.
  4. If confirmed, the system exits; if canceled, the student is returned to the main menu.
* Postcondition: Session registrations are confirmed and the program exits.

## **5. Testing Requirements**

* **Unit Testing:** Validate individual methods for profile creation, course validation, availability management, and session scheduling.
* **System Testing:** Simulate full workflows such as “Student creates profile → adds availability → joins a study session → confirms sessions on exit.”

## **6. Appendices**

* **A1: Course Prefix List (sample)**: CPSC, BIOL, MATH, ENGL, HIST, PHYS, CHEM.
* **A2: File Formats**: Profiles and sessions will be stored in text files with structured formatting (e.g., CSV or JSON-like).