1. Introduction

1.1 Purpose

The purpose of the Study Buddy Scheduling App is to help Clemson University students find classmates enrolled in the same courses, schedule study sessions, and manage availability. The application will simplify collaboration by allowing users to create profiles, match with peers, and confirm study sessions. This SRS defines the functional and non-functional requirements to guide the design and implementation.

1.2 Scope

The Study Buddy Scheduling App will be implemented as a command-line program. It will allow students to:

- Create and manage a personal profile with enrolled courses.
- Search for classmates taking the same courses.
- Add and update availability.
- Receive suggested study matches.
- Schedule, confirm, or cancel study sessions.

2. Overall Description

2.1 Product Perspective

The Study Buddy App is a standalone scheduling tool that integrates with Clemson's authentication system. It does not replace existing scheduling systems but complements them by focusing on student-to-student study coordination.

2.2 Product Features

- Profile management (courses, availability).
- Search and filtering by course.
- Match suggestion engine.
- Study session scheduling, confirmation, and cancellation.

2.3 User Classes and Characteristics

- **General Students**: Clemson undergraduates and graduates seeking study partners.

2.4 Operating Environment

- **Command-Line Implementation:** Runs on Linux/Unix/MacOS/Windows with C++.

3. Specific Requirements

3.1 Functional Requirements

FR1: Profile Management

- Users shall create and update their profile with enrolled courses.
- Users shall add, update, or remove availability.

FR2: Search and Matching

- Users shall search for other students by course.
- The user should receive matches based on overlapping courses and availability.

FR3: Scheduling

- Users shall send study session invitations.
- Recipients shall accept or decline invitations.
- Users shall be able to cancel sessions.
- Users shall be able to confirm sessions

3.2 Non-Functional Requirements

- Command-line version shall run on Windows, Mac, and Linux.
- The system shall be accompanied by technical documentation for future developers, detailing the build process, database schema, and API (if applicable).
- The system shall gracefully handle errors (e.g., invalid input, network failure) without crashing and provide a meaningful error message to the user.
- System should be easy to use and understand.

4. System Models

4.1 Use Case Example: Schedule a Study Session

Actors: Student A, Student B
Precondition: Both users are registered and logged in.
Steps:

1. Student A searches for classmates in “CPSC 3220.”
2. Student A selects Student B from the results.
3. Student A proposes a time slot.
4. Student B accepts.
5. The system updates both calendars and sends confirmation.

Postcondition: Session is stored in the system, and both users are notified.