First, we used ChatGPT to create an SRS (Software Requirements Specification):  
Prompt:  
“Project Theme A “Study Buddy” scheduling app for Clemson students: Students can create a profile with courses they are enrolled in. They can search for and schedule study sessions with classmates. Features: add/remove availability, suggest matches, and confirm meetings. Implementation: A command-line or web app in a chosen programming language Requirements Phase: Use ChatGPT to help write a formal Software Requirements Specification (SRS).”

ChatGPT then gave an SRS that seemed to include unnecessary requirements for the given project, so we prompted ChatGPT to simplify the SRS and from there, we edited the output until it fit what the project required.

Finished GPT output:  
**Software Requirements Specification (SRS)**

**Project:** Study Buddy – Clemson Student Scheduling App  
 **Author:** Tristin Franklin & Elise James  
 **Date:** August 28, 2025

## **1. Introduction**

**Purpose:**Study Buddy helps Clemson students find and schedule study sessions with classmates. Students can create profiles, share courses, manage availability, and schedule meetings.

**Scope:**

* Students create profiles with course enrollment
* Search for classmates in the same courses
* Schedule study sessions based on availability
* Suggest and confirm study partners

**Users:** Clemson students who want to study with peers

**Platform:**

* command-line app

## **2. Features**

1. **Profile Management:**
   * Create, delete profile
   * Add/remove courses
2. **Availability Management:**
   * Add, remove available times
   * Avoid scheduling conflicts
3. **Study Session Scheduling:**
   * Suggest study partners with overlapping courses and times
   * Confirm, reschedule, or cancel sessions
4. **Search Functionality:**
   * Find classmates by course
   * Filter by available times

## **3. Requirements**

**Functional Requirements:**

* Users can create and manage profiles
* Users can add/remove courses and availability
* System suggests study partners
* Users can schedule study sessions
* Users can search for classmates by course

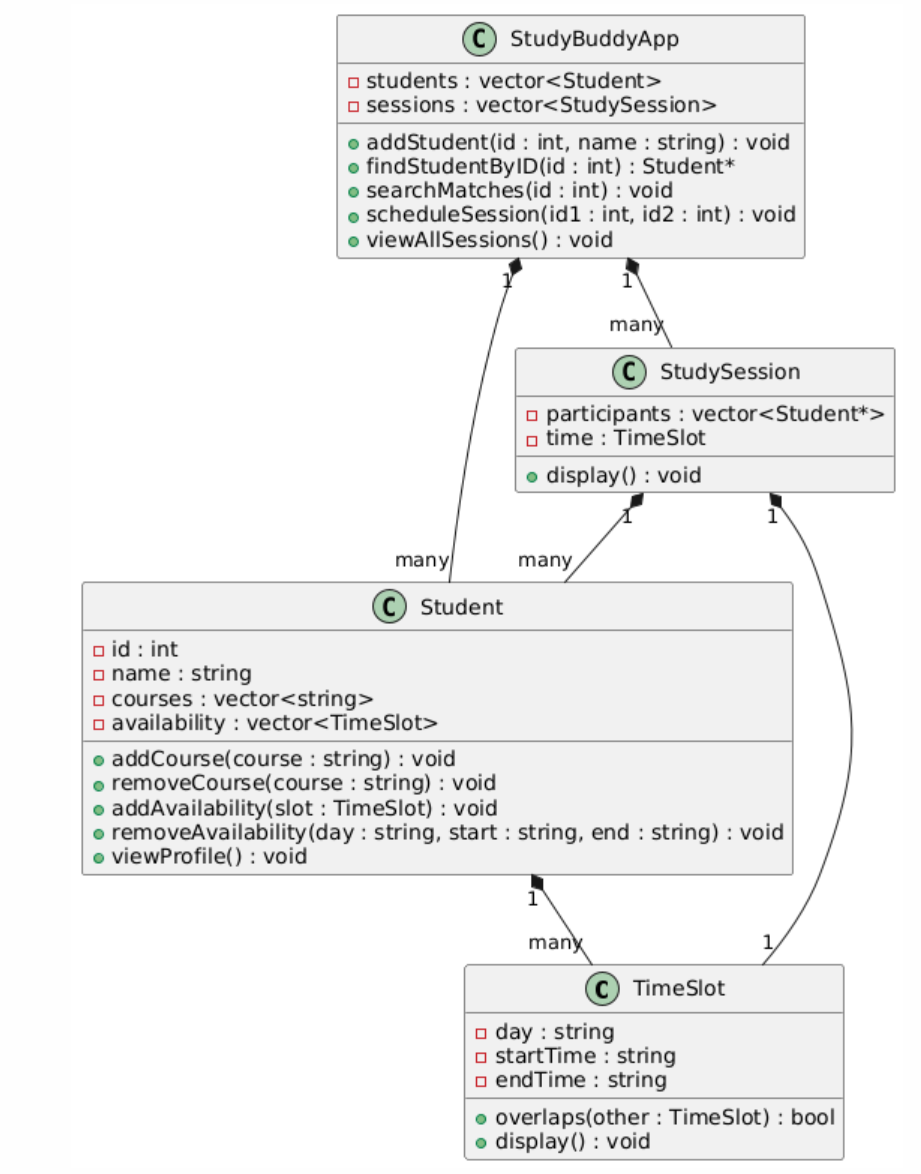
**Non-Functional Requirements:**

* Easy to use interface
* Works on command-line

## **4. Assumptions**

* Students enter accurate course and availability info
* Only Clemson students use the system

After ChatGPT gave us our SRS and we finished editing it to best fit the project, we then prompted ChatGPT to provide a UML for this project.  
ChatGPT prompt:  
“this will be in c++, create UML diagrams, pseudocode, or architectural sketches for this project.”

ChatGPT then output the UML we needed, so after this, Tristin prompted ChatGPT to create PlantUML code to easily create our UML. Here is the finished UML from the PlantUML code:  


After ChatGPT gave us our SRS, UMLs, and classes and functions we will be using, we gave it the prompt to create the code based on the project’s requirements:  
“Write the full system in one pass, strictly based on the design.”  
ChatGPT then output somewhat-functional code, and after some testing, we found many errors. We edited the code ourselves and prompted ChatGPT to create better error handling in the program with user input.

ChatGPT gave us a list of possible test cases after being prompted to do so, and then created an actual executable using the cassert library to accurately test our program. These test cases would have to be tweaked to be compatible with our program, as ChatGPT gave us test cases where integers were passed into the time function instead of strings.

ChatGPT prompting was very useful, especially using the waterfall development process. This process allows ChatGPT to “think” about what functions and classes will be used from beginning to end and allows for faster development. The code wasn’t perfect and required lots of editing as well as extra prompts for error handling.