SRS

Software Requirements Specification (SRS)

Project: Study Buddy – Clemson Student Scheduling App
Process Model: Waterfall (Group A)

1. Introduction
1.1 Purpose

The Study Buddy system enables Clemson students to create profiles, list enrolled courses, manage availability, and schedule study sessions with classmates. The system provides study partner matching and session confirmation features.

1.2 Scope

The application will:

Allow profile creation and course enrollment.

Manage availability (add/remove times).

Suggest compatible matches based on courses and schedules.

Enable scheduling and confirmation of meetings.

Provide both CLI and web front end, with a JavaScript/Node.js backend and React UI.

Store persistent data in a relational database.

1.3 Definitions, Acronyms, Abbreviations

CLI: Command Line Interface

DB: Database

API: Application Programming Interface

1.4 References

IEEE 830-1998 SRS standard.

2. Overall Description

2.1 Product Perspective

The system is a new application, client-server based, with REST APIs.

2.2 Product Functions

User authentication (login/register).

Profile/course management.

Availability management.

Study partner search and suggestion.

Meeting scheduling and confirmation.

2.3 User Characteristics

Primary users are Clemson students with valid university credentials. Users are assumed to have basic technical proficiency.

2.4 Constraints

Must run on Node.js backend with PostgreSQL DB.

Must provide a web React interface.

Clemson student authentication will be via name

Development must follow Waterfall methodology (no requirement changes after approval).

2.5 Assumptions and Dependencies

Reliable internet access.

University students provide accurate availability and course data.

Hosting via GitHub and supporting services.

3. Specific Requirements

3.1 Functional Requirements

FR-1: The system shall allow users to register, log in, and log out.
FR-2: The system shall allow users to add, edit, or delete courses.
FR-3: The system shall allow users to add or remove availability slots.
FR-4: The system shall suggest study partners based on common courses and overlapping availability.
FR-5: The system shall allow users to schedule and confirm meetings.
FR-6: The system shall send notifications upon meeting confirmation.
FR-7: The system shall store user data persistently in a relational DB.

3.2 Non-Functional Requirements

NFR-1: The system shall ensure secure storage of user credentials.
NFR-2: The system shall provide responses to API calls within 2 seconds.
NFR-3: The system shall be available 99% of the time during academic semesters.
NFR-4: The system shall support up to 1,000 concurrent users.
NFR-5: The system shall provide a usable interface on the web.

3.3 External Interface Requirements

User Interface: Web interface via React;

API Interface: RESTful JSON endpoints.

Database Interface: PostgreSQL.

Hardware: Runs on standard student laptops or smartphones.

4. Appendices

Glossary:

Study Buddy: Partner matched for a study session.

Availability Slot: Time range user sets as free for studying.

Pseudocode

Profile page

```
FUNCTION ProfilePage

  IMPORT useState
  IMPORT useApp (provides user state and actions)

  INITIALIZE activeTab = "info"
  INITIALIZE editMode = false
  INITIALIZE formData = { name: user.name, major: user.major }
  INITIALIZE newCourse = { courseCode: "", courseName: "" }
  INITIALIZE newAvailability = { day: "", startTime: "", endTime: "" }

  FUNCTION handleUpdateProfile:
    UPDATE user profile with formData (name + major)
    EXIT editMode

  FUNCTION handleAddCourse:
    IF courseCode does not match pattern (e.g., "CS 1010")
      SHOW alert "Invalid format"
      RETURN
    IF course already exists
      SHOW alert "Course already added"
      RETURN
    ADD newCourse to user.courses
    RESET newCourse inputs

  FUNCTION handleAddAvailability:
    IF startTime OR endTime is missing
      SHOW alert "Please select both start and end times"
      RETURN
    IF startTime >= endTime
      SHOW alert "End time must be after start time"
      RETURN
    ADD newAvailability to user.availability
    RESET newAvailability inputs

  RENDER UI:
    DISPLAY header "My Profile"

    DISPLAY Tabs: [Info, Courses, Availability]
```

```
   WHEN clicked, set activeTab

  IF activeTab = "info":
   IF NOT editMode:
     SHOW user.name (or "Not set")
     SHOW user.major (or "Not set")
     SHOW button "Edit Profile"
   ELSE:
     SHOW input for name
     SHOW input for major
     SHOW button "Save" (calls handleUpdateProfile)
     SHOW button "Cancel" (exits editMode)

  IF activeTab = "courses":
   DISPLAY form to add new course:
     input for courseCode
     input for courseName
     "Add Course" button (calls handleAddCourse)
   DISPLAY user.courses list
     For each course: show course info + "Remove" button

  IF activeTab = "availability":
   DISPLAY form to add availability:
     dropdown for day
     input for start time
     input for end time
     "Add Availability" button (calls handleAddAvailability)
   DISPLAY user.availability list
     For each slot: show day + times + "Remove" button

END FUNCTION
```

Matches page

```
FUNCTION findMatches(user, sessions):
    CREATE empty list called matches
    CREATE empty set called matchedUsers

    FOR EACH session IN sessions:
        IF user is enrolled in session.courseCode:

            // Check the host
            IF session.hostName is not the current user AND host not in matchedUsers:
                ADD host to matchedUsers
            ADD object {name: host, course: session.courseCode, sessionCount: number of
sessions hosted by host} TO matches

         // Check participants
         FOR EACH participant IN session.participants:
            IF participant is not the current user AND participant not in matchedUsers:
               ADD participant to matchedUsers
               ADD object {name: participant, course: session.courseCode, sessionCount:
number of sessions participant is in} TO matches

   RETURN matches


FUNCTION renderMatchesPage(user, sessions):
   matches = findMatches(user, sessions)

   DISPLAY "Study Buddy Matches"
   DISPLAY "People in your courses who are looking for study partners:"

   IF matches is empty:
      DISPLAY "No matches found. Join or create sessions to find study buddies!"
   ELSE:
      FOR EACH match IN matches:
         DISPLAY match.name
         DISPLAY "Common course: " + match.course
         DISPLAY "Active in " + match.sessionCount + " session(s)"
```

Sessions page

```
FUNCTION handleCreateSession():
   IF user.name is empty:
      ALERT "Set your name in profile before creating a session"
      RETURN

   IF any of newSession.courseCode, newSession.date, newSession.time, newSession.location
is empty:
      ALERT "Fill in all required fields"
      RETURN

   course = FIND user.courses WHERE course.courseCode == newSession.courseCode
   IF course is not found:
      ALERT "Must be enrolled in course to create a session"
```

```
        RETURN

    CALL createSession WITH newSession + courseName from course
    RESET newSession fields
    SET showCreateForm to false


FUNCTION handleJoinSession(sessionId):
    IF user.name is empty:
        ALERT "Set your name in profile before joining a session"
        RETURN

    session = FIND sessions WHERE session.id == sessionId
    IF user not enrolled in session.courseCode:
        ALERT "Must be enrolled in course to join this session"
        RETURN

    CALL joinSession(sessionId)


FUNCTION renderSessionsPage():
    DISPLAY "Study Sessions"
    DISPLAY Button "Create New Session" OR "Cancel" based on showCreateForm

    IF showCreateForm is true:
        DISPLAY form with:
            - Dropdown for user's courses
            - Date input
            - Time input
            - Location input
            - Optional description input
            - Button to call handleCreateSession

    DISPLAY "Available Sessions"
    IF sessions list is empty:
        DISPLAY "No sessions available"
    ELSE:
        FOR EACH session IN sessions:
            DISPLAY session info (course, host, date, time, location, description)
            DISPLAY participants list or "None yet"
            IF session.hostName != user.name:
                DISPLAY Button:
                    IF user already in session.participants -> "Joined", disabled
                    ELSE -> "Join Session", onClick -> handleJoinSession(session.id)
```

App.css

```
DEFINE app container
    max width = 800px
    centered horizontally
    padding = 20px
    font = system UI font stack

DEFINE header
    margin-bottom = 30px
    h1 color = dark purple
    h1 margin-bottom = 20px

DEFINE nav panel
    display = flex
    gap between buttons = 10px
    border-bottom = 2px solid light gray
    padding-bottom = 10px

DEFINE nav button & tabs button
    no background
    no border
    padding = 8px 16px
    cursor = pointer
    font size = 16px
    color = gray
    transition = smooth

ON hover of nav button or tab button
    color = darker gray

IF button is active
    border-bottom = orange
    color = orange
    font-weight = bold

DEFINE tabs container
    display = flex
    gap = 10px
    margin-bottom = 20px
    border-bottom = 1px solid light gray

DEFINE tab content
```

```
    padding-top/bottom = 20px

DEFINE form container
    background = light gray
    padding = 20px
    border-radius = 8px
    margin-bottom = 20px

DEFINE inputs, selects, textareas
    width = 100%
    padding = 8px 12px
    margin = 10px 0
    border = 1px solid light gray
    border-radius = 4px
    font size = 16px

DEFINE button
    background = orange
    text color = white
    no border
    padding = 10px 20px
    border-radius = 4px
    cursor = pointer
    font size = 16px
    margin-right = 10px
    transition = background color

ON button hover
    background = darker orange

IF button disabled
    background = gray
    cursor = not-allowed

DEFINE cancel button
    background = dark gray
    ON hover = darker gray

DEFINE button group
    display = flex
    gap = 10px
    margin-top = 10px

DEFINE course & availability item
```

```
    background = white
    border = 1px solid light gray
    padding = 15px
    border-radius = 4px
    margin-bottom = 10px
    display = flex
    justify content = space-between
    align-items = center

DEFINE session & match card
    background = white
    border = 1px solid light gray
    padding = 15px
    border-radius = 4px
    margin-bottom = 10px

DEFINE card headers (h4)
    margin = 0 0 10px 0
    color = dark purple

DEFINE card paragraphs
    margin = 5px 0
    color = dark gray

DEFINE create session button
    background = dark purple
    margin-bottom = 20px
    hover background = darker purple

DEFINE info display
    p margin = 10px 0
    button margin-top = 15px

DEFINE lists headings
    color = dark gray
    margin = 20px 0 15px 0

DEFINE textarea
    min-height = 80px
    resize = vertical

DEFINE no-matches section
    text-align = center
    padding = 40px
    background = light gray
    border-radius = 8px

DEFINE no-matches paragraph
    margin = 10px 0
    color = gray
```

App.jsx

```
IMPORT useState from React
IMPORT AppProvider from context
IMPORT ProfilePage, SessionsPage, MatchesPage
IMPORT CSS file

DEFINE App component
    CREATE state variable currentPage, default = "profile"

    RETURN
        WRAP everything inside AppProvider
            CREATE main container div with class "app"

            DEFINE header
                DISPLAY app title "Study Buddy"
                DEFINE navigation buttons
                    BUTTON "Profile"
                        IF currentPage is "profile"
                            mark as active
                        ON CLICK
                            set currentPage to "profile"

                    BUTTON "Sessions"
                        IF currentPage is "sessions"
                            mark as active
                        ON CLICK
                            set currentPage to "sessions"

                    BUTTON "Matches"
                        IF currentPage is "matches"
                            mark as active
                        ON CLICK
                            set currentPage to "matches"
            DEFINE main section
                IF currentPage is "profile"
                    DISPLAY ProfilePage component
                ELSE IF currentPage is "sessions"
                    DISPLAY SessionsPage component
                ELSE IF currentPage is "matches"
                    DISPLAY MatchesPage component

EXPORT App as default
```

Test plan

Test	Steps	Expected	Result
Edit profile	Click edit, change name and major	Profile updated with new values	Information changed. Success
View profile info	Go to profile to view it	Name and major are displayed	Information displayed. Success
Add course	Enter valid course code and name	Course appears in course list	Course appears Success
Add duplicate course	Enter existing course code	Alert showing course already added	Course already added alert. Success
Add availability	Select day/time, click add	Availability appears	Availability appears Success
Create session	Fill information needed to create a session	Session is added to session list	Session added to list Success
Join session	Click join an available session	User is added to participants	User added to participants Success
View matches	Go to matches page	List of matched users displayed that have the same course	Success
Navigation	Click all tabs	Correct page displayed	Success