

## User Stories:

### Input:

Create user stories and acceptance criteria for a study buddy app for Clemson students. The app needs to be able to:

- have students create a profile with courses they are enrolled in
- search for study sessions with classmates and schedule study sessions
- add and remove availability, suggest classmate matches, and confirm study session meetings

### Output:

Perfect — I'll write clear **user stories** (with acceptance criteria) specifically covering the functionality you listed:

---

## # 📖 User Stories for Clemson Study Buddy App

---

### ## \*\*1. Profile Creation\*\*

#### **\*\*User Story:\*\***

As a Clemson student, I want to create a profile with my enrolled courses so that I can connect with classmates in the same classes.

#### **\*\*Acceptance Criteria:\*\***

- \* Students can sign up using a valid Clemson email (`@clemson.edu`).
- \* Students can enter/edit their name, major, and list of enrolled courses.
- \* The system saves and updates profile information.
- \* Other students can only view names and shared courses (not private details like email).

---

### ## \*\*2. Search & Schedule Study Sessions\*\*

#### **\*\*User Story:\*\***

As a student, I want to search for study sessions with classmates so that I can join or schedule study time.

#### **\*\*Acceptance Criteria:\*\***

- \* Students can search by course (e.g., "MATH 1080") to find classmates.
- \* The search shows available students, their study times, and locations (if shared).
- \* Students can propose a new study session by selecting a date/time.
- \* The invited student receives a notification and can accept or decline.
- \* Once accepted, the session appears in both students' confirmed study schedule.

---

### ## \*\*3. Manage Availability\*\*

#### \*\*User Story:\*\*

As a student, I want to add or remove availability so that classmates know when I'm free to study.

#### \*\*Acceptance Criteria:\*\*

- \* Students can add available time slots (day, time range, optional location).
- \* Students can edit or delete existing availability.
- \* Availability updates are immediately reflected when classmates search or request sessions.

---

### ## \*\*4. Suggested Matches\*\*

#### \*\*User Story:\*\*

As a student, I want the system to suggest study partners who share my courses and availability so that I can quickly find the best match.

#### \*\*Acceptance Criteria:\*\*

- \* The system compares enrolled courses and overlapping availability.
- \* Suggested matches appear in a list ranked by number of shared courses and overlapping time slots.
- \* If no matches exist, the system shows "No suggested study buddies right now."

---

### ## \*\*5. Confirm Study Sessions\*\*

#### \*\*User Story:\*\*

As a student, I want to confirm study session meetings so that I know when and where I'm studying with classmates.

## **\*\*Acceptance Criteria:\*\***

- \* Students can accept or decline meeting requests.
- \* Accepted sessions are saved in a “Confirmed Sessions” list with details (classmate name, course, date/time, location).
- \* Students can cancel confirmed sessions, which notifies the other student.
- \* Past sessions are automatically moved to “History” or archived.

---

✨ That covers all the functionality you listed (profiles, searching, scheduling, availability, matches, confirmations).

Would you like me to also **\*\*group these into Epics\*\*** (like “Profiles,” “Scheduling,” “Availability”) for a more Agile-style backlog?

## **Evaluation:**

Being able to create a profile and enroll in courses is the most important feature to include. Once done, students should be able to connect with each other through study sessions. Study session attributes include date, time, place, and amount of people. Students also need to be able to add availability in order to coordinate schedules with others. While these requirements are essential, we needed to add some on top of these to cover a wider range of features. This includes creating a dashboard, providing email notifications of study sessions, viewing session history, providing a way for students to message each other regarding sessions, and adding a review/feedback section.

## **Code Generation:**

### **Input:**

Can you make some recommendations as to how to start this assignment: 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

Additionally explain what a product backlog is in agile development.

Note that these are the deliverables expected and we are the agile group:

Codebase (functional program) in github with a 2-3 minute video walk through of function and features

Process Artifacts

Waterfall: SRS, design docs (UML/pseudocode), test plan & results

Agile: backlog, user stories, sprint plans, stand-up logs, retrospective notes

AI Usage Report (1 page): Describe how ChatGPT was used, what was helpful, what required correction.

Reflection Report (1-2 pages, team-authored):

Strengths & weaknesses of the process used

Comparison of actual vs. expected outcomes

How ChatGPT influenced requirements, design, coding, and testing

## Output:

I'll help you get started with your Study Buddy scheduling app and explain agile product backlogs.

## Getting Started with Your Study Buddy App

**1. Technology Stack Decision** First, choose your implementation approach:

- **Web App:** React/Vue.js frontend + Node.js/Python backend + SQLite/PostgreSQL database
- **Command-line:** Python with file storage or lightweight database
- **Desktop App:** Python with tkinter or Java with Swing

For a student project with time constraints, I'd recommend a **web app with React frontend and Node.js backend** - it's modern, demonstrates full-stack skills, and easier to showcase in your video.

## 2. Initial Setup Steps

- Set up your GitHub repository with proper structure
- Create a basic project skeleton
- Set up development environment (Node.js, package.json, etc.)
- Plan your database schema (users, courses, study\_sessions, availability)

**3. Core Features to Implement First** Start with these MVP features:

- User registration/login
- Course enrollment management
- Basic availability setting
- Simple study session creation
- Match viewing (without complex algorithms initially)

## 4. Database Schema Planning

Users: user\_id, name, email, password\_hash

Courses: course\_id, course\_name, department, number

User\_Courses: user\_id, course\_id

Availability: user\_id, day\_of\_week, start\_time, end\_time

Study\_Sessions: session\_id, course\_id, creator\_id, datetime, status

Session\_Participants: session\_id, user\_id, status

## Product Backlog in Agile Development

A **product backlog** is a prioritized list of features, requirements, enhancements, and fixes that need to be implemented in your software product. Think of it as your project's "wishlist" organized by importance and value.

### Key Characteristics:

- **Living Document:** Constantly updated and refined throughout the project
- **Prioritized:** Most valuable/important items at the top
- **User-Focused:** Written from the user's perspective (often as user stories)
- **Estimated:** Items have effort estimates (story points, hours, etc.)

### What Goes in a Product Backlog:

- User stories ("As a student, I want to...")
- Bug fixes
- Technical requirements
- Performance improvements
- New feature requests

### For Your Study Buddy App, your backlog might include:

1. User account creation and authentication
2. Course enrollment functionality
3. Availability calendar interface
4. Study session creation
5. Matching algorithm implementation
6. Notification system
7. Session confirmation workflow
8. User profile management
9. Search and filter capabilities
10. Mobile responsiveness

### Agile Process Artifacts You'll Need:

- **Product Backlog:** Your complete feature list
- **User Stories:** Written as "As a [user type], I want [goal] so that [benefit]"
- **Sprint Plans:** What you'll accomplish in each 1-2 week sprint

- **Stand-up Logs:** Daily progress updates (what you did, plan to do, blockers)
- **Sprint Retrospectives:** What went well, what didn't, what to improve

**Practical Tip for Your Team:** Start by brainstorming all possible features, then ruthlessly prioritize. Focus on getting a working MVP first, then add polish. Document everything as you go - those process artifacts are worth significant points and will help your reflection report.

Would you like me to help you create initial user stories or set up a basic project structure for any particular technology stack?

## Evaluation:

The importance of this output is that it gives us context and a starting point. We are at the stage of the project where we are trying to find out footing and our AI tool helps immensely with that. From this point we decided to create a web application, starting a new repository on github, and read up on what the assignment entailed including learning about backlogs. This is where our documentation started and our recording of the process began.

## Input:

Now keep this in mind: Deliverables

1. Codebase (functional program) in github with a 2-3 minute video walk through of function and features
2. Process Artifacts
  - \* *Waterfall:* SRS, design docs (UML/pseudocode), test plan & results
  - \* *Agile:* backlog, user stories, sprint plans, stand-up logs, retrospective notes
3. AI Usage Report (1 page): Describe how ChatGPT was used, what was helpful, what required correction.
4. Reflection Report (1-2 pages, team-authored):
  - \* Strengths & weaknesses of the process used
  - \* Comparison of actual vs. expected outcomes
  - \* How ChatGPT influenced requirements, design, coding, and testing

Grading Rubric (100 points)

Category	Criteria	Points
Process Artifacts	Completeness, clarity, and adherence to process (Waterfall: SRS/design/test plan; Agile: backlog, stories, sprints, retrospectives)	25
Working Software	Functional, implements core features, meets requirements/user stories, clean code practices	20
AI Usage Report	Clear documentation of how ChatGPT was used, critical evaluation of AI outputs, thoughtful reflection on strengths/limitations	15
Reflection Report	Depth of analysis, ability to compare process theory vs. practice, lessons learned, clarity of writing	20
Presentation/Demo	Professionalism, clarity of communication, effective demonstration of process & product	10
Team Collaboration	Evidence of balanced participation, organization, and timely deliverables (peer/self-assessment factored in)	10
<b>Total</b>		<b>100</b>

Write me code for pythonanywhere and remember this is sprint writing in an agile fashion

### Output:

```
'from flask import Flask, render_template, request, redirect, url_for, flash, session, jsonify
from werkzeug.security import generate_password_hash, check_password_hash
import sqlite3
import os
from datetime import datetime, timedelta
import json
```

```
app = Flask(__name__)
app.secret_key = 'your-secret-key-change-in-production'
```

```
# Database initialization
```

```
def init_db():
```

```
    conn = sqlite3.connect('study_buddy.db')
    c = conn.cursor()
```

```
# Users table
```

```
c.execute("""CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)""")
```

```
# Courses table
```

```
c.execute("""CREATE TABLE IF NOT EXISTS courses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    course_code TEXT NOT NULL,
    course_name TEXT NOT NULL,
    department TEXT NOT NULL
)""")
```

```
# User courses (enrollment)
```

```
c.execute("""CREATE TABLE IF NOT EXISTS user_courses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    course_id INTEGER,
    FOREIGN KEY (user_id) REFERENCES users (id),
    FOREIGN KEY (course_id) REFERENCES courses (id),
```

```
    UNIQUE(user_id, course_id)
)''')
```

# Study sessions

```
c.execute("""CREATE TABLE IF NOT EXISTS study_sessions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    creator_id INTEGER,
    course_id INTEGER,
    title TEXT NOT NULL,
    description TEXT,
    session_date DATE,
    session_time TIME,
    duration INTEGER DEFAULT 60,
    location TEXT,
    max_participants INTEGER DEFAULT 4,
    status TEXT DEFAULT 'open',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (creator_id) REFERENCES users (id),
    FOREIGN KEY (course_id) REFERENCES courses (id)
)''')
```

# Session participants

```
c.execute("""CREATE TABLE IF NOT EXISTS session_participants (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    session_id INTEGER,
    user_id INTEGER,
    status TEXT DEFAULT 'pending',
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (session_id) REFERENCES study_sessions (id),
    FOREIGN KEY (user_id) REFERENCES users (id),
    UNIQUE(session_id, user_id)
)''')
```

# User availability

```
c.execute("""CREATE TABLE IF NOT EXISTS user_availability (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    day_of_week INTEGER,
    start_time TIME,
    end_time TIME,
    FOREIGN KEY (user_id) REFERENCES users (id)
)''')
```

# Insert sample courses



```

sample_courses = [
    ('CS1010', 'Introduction to Computer Science', 'Computer Science'),
    ('MATH1060', 'Calculus I', 'Mathematics'),
    ('PHYS2070', 'University Physics I', 'Physics'),
    ('CHEM1050', 'General Chemistry', 'Chemistry'),
    ('CS2030', 'Computer Science II', 'Computer Science'),
    ('CS3240', 'Database Systems', 'Computer Science'),
]

c.executemany('INSERT OR IGNORE INTO courses (course_code, course_name,
department) VALUES (?, ?, ?)',
              sample_courses)

conn.commit()
conn.close()

# Database helper functions
def get_db_connection():
    conn = sqlite3.connect('study_buddy.db')
    conn.row_factory = sqlite3.Row
    return conn

@app.route('/')
def index():
    if 'user_id' in session:
        return redirect(url_for('dashboard'))
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        if not username or not email or not password:
            flash('All fields are required!')
            return render_template('register.html')

        conn = get_db_connection()

        # Check if user exists
        existing_user = conn.execute('SELECT id FROM users WHERE username = ? OR email =
?',

```

```

        (username, email)).fetchone()

    if existing_user:
        flash('Username or email already exists!')
        conn.close()
        return render_template('register.html')

    # Create user
    password_hash = generate_password_hash(password)
    conn.execute('INSERT INTO users (username, email, password_hash) VALUES (?, ?, ?)',
                 (username, email, password_hash))
    conn.commit()
    conn.close()

    flash('Registration successful! Please log in.')
    return redirect(url_for('login'))

return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        conn = get_db_connection()
        user = conn.execute('SELECT * FROM users WHERE username = ?',
                             (username,)).fetchone()
        conn.close()

        if user and check_password_hash(user['password_hash'], password):
            session['user_id'] = user['id']
            session['username'] = user['username']
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid username or password!')

    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))

```

```

@app.route('/dashboard')
def dashboard():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()

    # Get user's enrolled courses
    user_courses = conn.execute("""
        SELECT c.id, c.course_code, c.course_name, c.department
        FROM courses c
        JOIN user_courses uc ON c.id = uc.course_id
        WHERE uc.user_id = ?
    """, (session['user_id'],)).fetchall()

    # Get upcoming study sessions for user's courses
    upcoming_sessions = conn.execute("""
        SELECT s.*, c.course_code, c.course_name, u.username as creator_name,
            COUNT(sp.id) as participant_count
        FROM study_sessions s
        JOIN courses c ON s.course_id = c.id
        JOIN users u ON s.creator_id = u.id
        LEFT JOIN session_participants sp ON s.id = sp.session_id AND sp.status = 'confirmed'
        WHERE s.course_id IN (
            SELECT course_id FROM user_courses WHERE user_id = ?
        ) AND s.session_date >= date('now')
        GROUP BY s.id
        ORDER BY s.session_date, s.session_time
        LIMIT 5
    """, (session['user_id'],)).fetchall()

    conn.close()

    return render_template('dashboard.html',
        user_courses=user_courses,
        upcoming_sessions=upcoming_sessions)

@app.route('/courses')
def courses():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()

```

```

# Get all courses
all_courses = conn.execute('SELECT * FROM courses ORDER BY department,
course_code').fetchall()

# Get user's enrolled courses
enrolled_course_ids = [row[0] for row in conn.execute(
    'SELECT course_id FROM user_courses WHERE user_id = ?',
    (session['user_id'],)
).fetchall()]

conn.close()

return render_template('courses.html',
    all_courses=all_courses,
    enrolled_course_ids=enrolled_course_ids)

@app.route('/enroll_course/<int:course_id>')
def enroll_course(course_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    try:
        conn.execute('INSERT INTO user_courses (user_id, course_id) VALUES (?, ?)',
            (session['user_id'], course_id))
        conn.commit()
        flash('Successfully enrolled in course!')
    except sqlite3.IntegrityError:
        flash('You are already enrolled in this course!')

    conn.close()
    return redirect(url_for('courses'))

@app.route('/drop_course/<int:course_id>')
def drop_course(course_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    conn.execute('DELETE FROM user_courses WHERE user_id = ? AND course_id = ?',
        (session['user_id'], course_id))
    conn.commit()
    conn.close()

```

```
flash('Successfully dropped course!')
return redirect(url_for('courses'))
```

```
@app.route('/sessions')
```

```
def sessions():
```

```
    if 'user_id' not in session:
        return redirect(url_for('login'))
```

```
    conn = get_db_connection()
```

```
    # Get all upcoming sessions for user's courses
```

```
    sessions = conn.execute("""
```

```
        SELECT s.*, c.course_code, c.course_name, u.username as creator_name,
```

```
            COUNT(sp.id) as participant_count,
```

```
            CASE WHEN s.creator_id = ? THEN 1 ELSE 0 END as is_creator,
```

```
            CASE WHEN sp_user.user_id IS NOT NULL THEN sp_user.status ELSE 'not_joined'
```

```
END as user_status
```

```
    FROM study_sessions s
```

```
    JOIN courses c ON s.course_id = c.id
```

```
    JOIN users u ON s.creator_id = u.id
```

```
    LEFT JOIN session_participants sp ON s.id = sp.session_id AND sp.status = 'confirmed'
```

```
    LEFT JOIN session_participants sp_user ON s.id = sp_user.session_id AND
```

```
    sp_user.user_id = ?
```

```
    WHERE s.course_id IN (
```

```
        SELECT course_id FROM user_courses WHERE user_id = ?
```

```
    ) AND s.session_date >= date('now')
```

```
    GROUP BY s.id
```

```
    ORDER BY s.session_date, s.session_time
```

```
""", (session['user_id'], session['user_id'], session['user_id'])).fetchall()
```

```
    conn.close()
```

```
    return render_template('sessions.html', sessions=sessions)
```

```
@app.route('/create_session', methods=['GET', 'POST'])
```

```
def create_session():
```

```
    if 'user_id' not in session:
        return redirect(url_for('login'))
```

```
    if request.method == 'POST':
```

```
        course_id = request.form['course_id']
```

```
        title = request.form['title']
```

```
        description = request.form['description']
```

```
        session_date = request.form['session_date']
```

```

session_time = request.form['session_time']
duration = request.form['duration']
location = request.form['location']
max_participants = request.form['max_participants']

conn = get_db_connection()
conn.execute("""
    INSERT INTO study_sessions
    (creator_id, course_id, title, description, session_date, session_time,
    duration, location, max_participants)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
""", (session['user_id'], course_id, title, description, session_date,
    session_time, duration, location, max_participants))
conn.commit()
conn.close()

flash('Study session created successfully!')
return redirect(url_for('sessions'))

# GET request - show form
conn = get_db_connection()
user_courses = conn.execute("""
    SELECT c.id, c.course_code, c.course_name
    FROM courses c
    JOIN user_courses uc ON c.id = uc.course_id
    WHERE uc.user_id = ?
""", (session['user_id'],)).fetchall()
conn.close()

return render_template('create_session.html', user_courses=user_courses)

@app.route('/join_session/<int:session_id>')
def join_session(session_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    try:
        conn.execute('INSERT INTO session_participants (session_id, user_id, status) VALUES
        (?, ?, ?)',
            (session_id, session['user_id'], 'confirmed'))
        conn.commit()
        flash('Successfully joined study session!')
    except sqlite3.IntegrityError:

```

```

        flash('You have already joined this session!')

    conn.close()
    return redirect(url_for('sessions'))

@app.route('/leave_session/<int:session_id>')
def leave_session(session_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    conn.execute('DELETE FROM session_participants WHERE session_id = ? AND user_id =
?',
                (session_id, session['user_id']))
    conn.commit()
    conn.close()

    flash('You have left the study session.')
    return redirect(url_for('sessions'))

if __name__ == '__main__':
    init_db()
    app.run(debug=True)

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Study Buddy - {% block title %}{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
        }
        .navbar-brand {
            font-weight: bold;
            color: #ff6600 !important;
        }
        .card {

```

```

        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        border: none;
        margin-bottom: 20px;
    }
    .btn-primary {
        background-color: #ff6600;
        border-color: #ff6600;
    }
    .btn-primary:hover {
        background-color: #e55a00;
        border-color: #e55a00;
    }
    .study-session-card {
        border-left: 4px solid #ff6600;
    }
    .footer {
        background-color: #343a40;
        color: white;
        padding: 20px 0;
        margin-top: 50px;
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
        <div class="container">
            <a class="navbar-brand" href="{{ url_for('index') }}">
                <i class="fas fa-graduation-cap"></i> Study Buddy
            </a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    {% if session.user_id %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('dashboard') }}">
                            <i class="fas fa-tachometer-alt"></i> Dashboard
                        </a>
                    </li>
                    <li class="nav-item">

```



```

        <a class="nav-link" href="{{ url_for('courses') }}">
            <i class="fas fa-book"></i> Courses
        </a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{ url_for('sessions') }}">
            <i class="fas fa-users"></i> Study Sessions
        </a>
    </li>
{% endif %}
</ul>

<ul class="navbar-nav">
    {% if session.user_id %}
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown">
                <i class="fas fa-user"></i> {{ session.username }}
            </a>
            <ul class="dropdown-menu">
                <li><a class="dropdown-item" href="#"><i class="fas fa-user-cog"></i>
Profile</a></li>
                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item" href="{{ url_for('logout') }}"><i class="fas
fa-sign-out-alt"></i> Logout</a></li>
            </ul>
        </li>
    {% else %}
        <li class="nav-item">
            <a class="nav-link" href="{{ url_for('login') }}">
                <i class="fas fa-sign-in-alt"></i> Login
            </a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{ url_for('register') }}">
                <i class="fas fa-user-plus"></i> Register
            </a>
        </li>
    {% endif %}
</ul>
</div>
</div>
</nav>

```

```

<main class="container mt-4">
  {% with messages = get_flashed_messages() %}
    {% if messages %}
      {% for message in messages %}
        <div class="alert alert-info alert-dismissible fade show" role="alert">
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
        </div>
      {% endfor %}
    {% endif %}
  {% endwith %}

  {% block content %}{% endblock %}
</main>

<footer class="footer mt-5">
  <div class="container text-center">
    <span class="text-muted">Study Buddy © 2025 - Helping Clemson Students
Connect</span>
  </div>
</footer>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
  {% block scripts %}{% endblock %}
</body>
</html>

```

```
{% extends "base.html" %}
```

```
{% block title %}Dashboard{% endblock %}
```

```
{% block content %}
```

```
<div class="row">
```

```
  <div class="col-md-12">
```

```
    <div class="d-flex justify-content-between align-items-center mb-4">
```

```
      <h1 class="h3">Welcome back, {{ session.username }}!</h1>
```

```
      <a href="{{ url_for('create_session') }}" class="btn btn-primary">
```

```
        <i class="fas fa-plus"></i> Create Study Session
```

```
      </a>
```

```
    </div>
```

```
  </div>
```

```
</div>
```

```

<div class="row">
  <!-- Enrolled Courses -->
  <div class="col-md-6">
    <div class="card">
      <div class="card-header">
        <h5 class="card-title mb-0">
          <i class="fas fa-book text-primary"></i> Your Courses ({{ user_courses|length }})
        </h5>
      </div>
      <div class="card-body">
        {% if user_courses %}
          <div class="list-group list-group-flush">
            {% for course in user_courses %}
              <div class="list-group-item d-flex justify-content-between align-items-center">
                <div>
                  <strong>{{ course.course_code }}</strong><br>
                  <small class="text-muted">{{ course.course_name }}</small>
                </div>
                <span class="badge bg-secondary rounded-pill">{{ course.department
}}</span>
              </div>
            {% endfor %}
          </div>
          <div class="mt-3">
            <a href="{{ url_for('courses') }}" class="btn btn-outline-primary btn-sm">
              <i class="fas fa-plus"></i> Manage Courses
            </a>
          </div>
        {% else %}
          <div class="text-center py-4">
            <i class="fas fa-book-open fa-3x text-muted mb-3"></i>
            <p class="text-muted">You haven't enrolled in any courses yet.</p>
            <a href="{{ url_for('courses') }}" class="btn btn-primary">
              <i class="fas fa-plus"></i> Browse Courses
            </a>
          </div>
        {% endif %}
      </div>
    </div>
  </div>

  <!-- Upcoming Study Sessions -->
  <div class="col-md-6">

```

```

<div class="card">
  <div class="card-header">
    <h5 class="card-title mb-0">
      <i class="fas fa-calendar-alt text-success"></i> Upcoming Sessions
    </h5>
  </div>
  <div class="card-body">
    {% if upcoming_sessions %}
      {% for session in upcoming_sessions %}
        <div class="card study-session-card mb-3">
          <div class="card-body">
            <div class="d-flex justify-content-between align-items-start">
              <div>
                <h6 class="card-title">{{ session.title }}</h6>
                <p class="card-text">
                  <strong>{{ session.course_code }}</strong> - {{ session.course_name }}<br>
                  <i class="fas fa-user"></i> {{ session.creator_name }}<br>
                  <i class="fas fa-calendar"></i> {{ session.session_date }}<br>
                  <i class="fas fa-clock"></i> {{ session.session_time }}
                  ({{ session.duration }} min)<br>
                  {% if session.location %}
                    <i class="fas fa-map-marker-alt"></i> {{ session.location }}<br>
                  {% endif %}
                  <i class="fas fa-users"></i> {{ session.participant_count }}/{{
session.max_participants }} participants
                </p>
              </div>
              <span class="badge bg-success">{{ session.status|title }}</span>
            </div>
          </div>
        </div>
      {% endfor %}
    <div class="mt-3">
      <a href="{{ url_for('sessions') }}" class="btn btn-outline-success btn-sm">
        <i class="fas fa-eye"></i> View All Sessions
      </a>
    </div>
  </div>
  {% else %}
    <div class="text-center py-4">
      <i class="fas fa-users fa-3x text-muted mb-3"></i>
      <p class="text-muted">No upcoming study sessions for your courses.</p>
      <a href="{{ url_for('create_session') }}" class="btn btn-success">
        <i class="fas fa-plus"></i> Create First Session
      </a>
    </div>
  </div>

```

```

        </a>
    </div>
    {% endif %}
</div>
</div>
</div>
</div>

<!-- Quick Stats Row -->
<div class="row mt-4">
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-book fa-2x text-primary mb-2"></i>
                <h4 class="card-title">{{ user_courses|length }}</h4>
                <p class="card-text">Enrolled Courses</p>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-calendar-check fa-2x text-success mb-2"></i>
                <h4 class="card-title">{{ upcoming_sessions|length }}</h4>
                <p class="card-text">Upcoming Sessions</p>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-users fa-2x text-info mb-2"></i>
                <h4 class="card-title">0</h4>
                <p class="card-text">Study Partners</p>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-trophy fa-2x text-warning mb-2"></i>
                <h4 class="card-title">0</h4>
                <p class="card-text">Sessions Completed</p>
            </div>
        </div>
    </div>

```

```
</div>
</div>
</div>
{% endblock %}
`
```

## ‘Study Buddy App - Sprint Development Guide

### Project Structure

```
study-buddy-app/
├── app.py                # Main Flask application
├── requirements.txt      # Python dependencies
├── study_buddy.db        # SQLite database (auto-created)
├── templates/           # HTML templates
│   ├── base.html        # Base template
│   ├── index.html       # Landing page
│   ├── login.html       # Login form
│   ├── register.html    # Registration form
│   ├── dashboard.html   # User dashboard
│   ├── courses.html     # Course management
│   ├── sessions.html    # Study sessions list
│   └── create_session.html # Create session form
├── static/              # Static files (CSS, JS, images)
│   └── style.css         # Custom styles
├── agile-docs/          # Process artifacts
│   ├── product-backlog.md
│   ├── user-stories.md
│   ├── sprint-plans/
│   ├── standup-logs/
│   └── retrospectives/
└── README.md            # Project documentation
```

### PythonAnywhere Deployment Steps

#### 1. Upload Files

1. Create account on PythonAnywhere (free tier available)
2. Upload all project files to `/home/yourusername/study-buddy-app/`

Install requirements in Bash console:  
pip3.10 install --user flask werkzeug

3.

## 2. Configure Web App

1. Go to Web tab → Create new web app
2. Choose "Manual configuration" → Python 3.10
3. Set source code to `/home/yourusername/study-buddy-app`

Edit WSGI file:

```
import syspath = '/home/yourusername/study-buddy-app'if path not in sys.path:
    sys.path.append(path)from app import app as application
```

- 4.

## 3. Database Initialization

Run in console:

```
cd /home/yourusername/study-buddy-app
python3.10 app.py
```

## Sprint Planning (Agile Process)

### Sprint 1 (Week 1): Foundation & Authentication

**Sprint Goal:** Basic user management and course enrollment

**User Stories:**

- As a student, I want to register an account so I can use the app
- As a student, I want to log in so I can access my profile
- As a student, I want to enroll in courses so I can find study partners

**Sprint Backlog:**

- [ ] User registration/login system
- [ ] Course catalog and enrollment
- [ ] Basic dashboard
- [ ] Database schema implementation

**Definition of Done:**

- Users can register and log in
- Users can browse and enroll in courses
- Basic navigation works
- Code is committed to GitHub

## **Sprint 2 (Week 2): Study Sessions Core**

**Sprint Goal:** Create and view study sessions

**User Stories:**

- As a student, I want to create study sessions so others can join me
- As a student, I want to view available sessions for my courses
- As a student, I want to join study sessions to collaborate

**Sprint Backlog:**

- ☐ Study session creation form
- ☐ Session listing and filtering
- ☐ Join/leave session functionality
- ☐ Session details view

## **Sprint 3 (Week 3): Matching & Enhancement**

**Sprint Goal:** Improve user experience and add matching features

**User Stories:**

- As a student, I want suggested study matches based on my courses
- As a student, I want to see session participants
- As a student, I want notifications about my sessions

**Sprint Backlog:**

- ☐ Basic matching algorithm
- ☐ Enhanced UI/UX
- ☐ Session participant management
- ☐ Responsive design improvements



**Required Templates (Create these in templates/)**

**templates/**

,

### **Evaluation:**

From the output of this part we started assembling our app by creating the skeleton as defined by AI, set up PythonAnywhere, and copy and pasted some code into the relevant files. The generation of code didn't give us exactly everything we needed but was a really good starting point and most likely is more than what we would have achieved in a couple hours. In addition,



we learned how to host dynamic information and databases because previously this team had experience only with github pages which is static.

### Input:

give me the code files and the setup for python anywhere

### Output:

```
'from flask import Flask, render_template, request, redirect, url_for, flash, session
from werkzeug.security import generate_password_hash, check_password_hash
import sqlite3
from datetime import datetime
```

```
app = Flask(__name__)
app.secret_key = 'your-secret-key-change-in-production'
```

```
def init_db():
    conn = sqlite3.connect('study_buddy.db')
    c = conn.cursor()

    # Users table
    c.execute("""CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE NOT NULL,
        email TEXT UNIQUE NOT NULL,
        password_hash TEXT NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )""")
```

```
# Courses table
c.execute("""CREATE TABLE IF NOT EXISTS courses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    course_code TEXT NOT NULL,
    course_name TEXT NOT NULL,
    department TEXT NOT NULL
)""")
```

```
# User courses (enrollment)
c.execute("""CREATE TABLE IF NOT EXISTS user_courses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

        user_id INTEGER,
        course_id INTEGER,
        FOREIGN KEY (user_id) REFERENCES users (id),
        FOREIGN KEY (course_id) REFERENCES courses (id),
        UNIQUE(user_id, course_id)
    )")

```

# Study sessions

```

c.execute("""CREATE TABLE IF NOT EXISTS study_sessions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    creator_id INTEGER,
    course_id INTEGER,
    title TEXT NOT NULL,
    description TEXT,
    session_date DATE,
    session_time TIME,
    duration INTEGER DEFAULT 60,
    location TEXT,
    max_participants INTEGER DEFAULT 4,
    status TEXT DEFAULT 'open',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (creator_id) REFERENCES users (id),
    FOREIGN KEY (course_id) REFERENCES courses (id)
)""")

```

# Session participants

```

c.execute("""CREATE TABLE IF NOT EXISTS session_participants (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    session_id INTEGER,
    user_id INTEGER,
    status TEXT DEFAULT 'confirmed',
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (session_id) REFERENCES study_sessions (id),
    FOREIGN KEY (user_id) REFERENCES users (id),
    UNIQUE(session_id, user_id)
)""")

```

# Insert sample courses

```

sample_courses = [
    ('CS1010', 'Introduction to Computer Science', 'Computer Science'),
    ('MATH1060', 'Calculus I', 'Mathematics'),
    ('PHYS2070', 'University Physics I', 'Physics'),
    ('CHEM1050', 'General Chemistry', 'Chemistry'),
    ('CS2030', 'Computer Science II', 'Computer Science'),

```

```

        ('CS3240', 'Database Systems', 'Computer Science'),
        ('ENGL1010', 'English Composition I', 'English'),
        ('HIST1010', 'World History', 'History')
    ]

    c.executemany('INSERT OR IGNORE INTO courses (course_code, course_name,
department) VALUES (?, ?, ?)',
                sample_courses)

    conn.commit()
    conn.close()

def get_db_connection():
    conn = sqlite3.connect('study_buddy.db')
    conn.row_factory = sqlite3.Row
    return conn

@app.route('/')
def index():
    if 'user_id' in session:
        return redirect(url_for('dashboard'))
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        if not username or not email or not password:
            flash('All fields are required!')
            return render_template('register.html')

        conn = get_db_connection()
        existing_user = conn.execute('SELECT id FROM users WHERE username = ? OR email =
?',
                                    (username, email)).fetchone()

        if existing_user:
            flash('Username or email already exists!')
            conn.close()
            return render_template('register.html')

```

```

password_hash = generate_password_hash(password)
conn.execute('INSERT INTO users (username, email, password_hash) VALUES (?, ?, ?)',
             (username, email, password_hash))
conn.commit()
conn.close()

flash('Registration successful! Please log in.')
return redirect(url_for('login'))

return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        conn = get_db_connection()
        user = conn.execute('SELECT * FROM users WHERE username = ?',
                             (username,)).fetchone()
        conn.close()

        if user and check_password_hash(user['password_hash'], password):
            session['user_id'] = user['id']
            session['username'] = user['username']
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid username or password!')

    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))

@app.route('/dashboard')
def dashboard():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()

    user_courses = conn.execute("""

```

```

        SELECT c.id, c.course_code, c.course_name, c.department
        FROM courses c
        JOIN user_courses uc ON c.id = uc.course_id
        WHERE uc.user_id = ?
    """ , (session['user_id'],)).fetchall()

    upcoming_sessions = conn.execute("""
        SELECT s.*, c.course_code, c.course_name, u.username as creator_name,
            COUNT(sp.id) as participant_count
        FROM study_sessions s
        JOIN courses c ON s.course_id = c.id
        JOIN users u ON s.creator_id = u.id
        LEFT JOIN session_participants sp ON s.id = sp.session_id AND sp.status = 'confirmed'
        WHERE s.course_id IN (
            SELECT course_id FROM user_courses WHERE user_id = ?
        ) AND s.session_date >= date('now')
        GROUP BY s.id
        ORDER BY s.session_date, s.session_time
        LIMIT 5
    """ , (session['user_id'],)).fetchall()

    conn.close()

    return render_template('dashboard.html',
        user_courses=user_courses,
        upcoming_sessions=upcoming_sessions)

@app.route('/courses')
def courses():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    all_courses = conn.execute('SELECT * FROM courses ORDER BY department,
course_code').fetchall()
    enrolled_course_ids = [row[0] for row in conn.execute(
        'SELECT course_id FROM user_courses WHERE user_id = ?',
        (session['user_id'],)
    ).fetchall()]
    conn.close()

    return render_template('courses.html',
        all_courses=all_courses,
        enrolled_course_ids=enrolled_course_ids)

```

```

@app.route('/enroll_course/<int:course_id>')
def enroll_course(course_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    try:
        conn.execute('INSERT INTO user_courses (user_id, course_id) VALUES (?, ?)',
                      (session['user_id'], course_id))
        conn.commit()
        flash('Successfully enrolled in course!')
    except sqlite3.IntegrityError:
        flash('You are already enrolled in this course!')
    conn.close()
    return redirect(url_for('courses'))

@app.route('/drop_course/<int:course_id>')
def drop_course(course_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    conn.execute('DELETE FROM user_courses WHERE user_id = ? AND course_id = ?',
                  (session['user_id'], course_id))
    conn.commit()
    conn.close()

    flash('Successfully dropped course!')
    return redirect(url_for('courses'))

@app.route('/sessions')
def sessions():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    sessions = conn.execute("""
        SELECT s.*, c.course_code, c.course_name, u.username as creator_name,
               COUNT(sp.id) as participant_count,
               CASE WHEN s.creator_id = ? THEN 1 ELSE 0 END as is_creator,
               CASE WHEN sp_user.user_id IS NOT NULL THEN sp_user.status ELSE 'not_joined'
        END as user_status
        FROM study_sessions s
    """)

```

```

        JOIN courses c ON s.course_id = c.id
        JOIN users u ON s.creator_id = u.id
        LEFT JOIN session_participants sp ON s.id = sp.session_id AND sp.status = 'confirmed'
        LEFT JOIN session_participants sp_user ON s.id = sp_user.session_id AND
sp_user.user_id = ?
        WHERE s.course_id IN (
            SELECT course_id FROM user_courses WHERE user_id = ?
        ) AND s.session_date >= date('now')
        GROUP BY s.id
        ORDER BY s.session_date, s.session_time
    """, (session['user_id'], session['user_id'], session['user_id'])).fetchall()

    conn.close()
    return render_template('sessions.html', sessions=sessions)

@app.route('/create_session', methods=['GET', 'POST'])
def create_session():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        course_id = request.form['course_id']
        title = request.form['title']
        description = request.form['description']
        session_date = request.form['session_date']
        session_time = request.form['session_time']
        duration = request.form['duration']
        location = request.form['location']
        max_participants = request.form['max_participants']

        conn = get_db_connection()
        conn.execute("""
            INSERT INTO study_sessions
            (creator_id, course_id, title, description, session_date, session_time,
            duration, location, max_participants)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (session['user_id'], course_id, title, description, session_date,
            session_time, duration, location, max_participants))
        conn.commit()
        conn.close()

        flash('Study session created successfully!')
        return redirect(url_for('sessions'))

```

```

conn = get_db_connection()
user_courses = conn.execute("""
    SELECT c.id, c.course_code, c.course_name
    FROM courses c
    JOIN user_courses uc ON c.id = uc.course_id
    WHERE uc.user_id = ?
""", (session['user_id'],)).fetchall()
conn.close()

return render_template('create_session.html', user_courses=user_courses)

@app.route('/join_session/<int:session_id>')
def join_session(session_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    try:
        conn.execute('INSERT INTO session_participants (session_id, user_id) VALUES (?, ?)',
            (session_id, session['user_id']))
        conn.commit()
        flash('Successfully joined study session!')
    except sqlite3.IntegrityError:
        flash('You have already joined this session!')
    conn.close()
    return redirect(url_for('sessions'))

@app.route('/leave_session/<int:session_id>')
def leave_session(session_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    conn.execute('DELETE FROM session_participants WHERE session_id = ? AND user_id = ?',
        (session_id, session['user_id']))
    conn.commit()
    conn.close()

    flash('You have left the study session.')
    return redirect(url_for('sessions'))

if __name__ == '__main__':
    init_db()

```



```
app.run(debug=True)
```

```
'<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Study Buddy - {% block title %}{% endblock %}</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/font-awesome@6.0.0/css/all.min.css"
rel="stylesheet">
  <style>
    :root {
      --clemson-orange: #ff6600;
    }
    .navbar-brand {
      font-weight: bold;
      color: var(--clemson-orange) !important;
    }
    .btn-primary {
      background-color: var(--clemson-orange);
      border-color: var(--clemson-orange);
    }
    .btn-primary:hover {
      background-color: #e55a00;
      border-color: #e55a00;
    }
    .text-primary {
      color: var(--clemson-orange) !important;
    }
    .study-session-card {
      border-left: 4px solid var(--clemson-orange);
    }
    .hero-section {
      background: linear-gradient(135deg, #f8f9fa 0%, #e9ecef 100%);
      border-radius: 10px;
    }
    .card {
      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
      margin-bottom: 20px;
    }
  </style>
```

```

</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
    <div class="container">
      <a class="navbar-brand" href="{{ url_for('index') }}">
        <i class="fas fa-graduation-cap"></i> Study Buddy
      </a>

      <div class="navbar-nav ms-auto">
        {% if session.user_id %}
          <a class="nav-link" href="{{ url_for('dashboard') }}">
            <i class="fas fa-tachometer-alt"></i> Dashboard
          </a>
          <a class="nav-link" href="{{ url_for('courses') }}">
            <i class="fas fa-book"></i> Courses
          </a>
          <a class="nav-link" href="{{ url_for('sessions') }}">
            <i class="fas fa-users"></i> Sessions
          </a>
          <a class="nav-link" href="{{ url_for('logout') }}">
            <i class="fas fa-sign-out-alt"></i> Logout ({{ session.username }})
          </a>
        {% else %}
          <a class="nav-link" href="{{ url_for('login') }}">
            <i class="fas fa-sign-in-alt"></i> Login
          </a>
          <a class="nav-link" href="{{ url_for('register') }}">
            <i class="fas fa-user-plus"></i> Register
          </a>
        {% endif %}
      </div>
    </div>
  </nav>

  <main class="container mt-4">
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        {% for message in messages %}
          <div class="alert alert-info alert-dismissible fade show" role="alert">
            {{ message }}
            <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
          </div>
        {% endfor %}
      {% endif %}
    </main>

```

```
{% endwith %}

{% block content %}{% endblock %}
</main>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
`
```

## PythonAnywhere Deployment Guide

### Step 1: Upload Files

1. **Go to PythonAnywhere.com** and create a free account
2. **Go to Files tab** and upload these files:

#### Required Files:

/home/yourusername/Study-Buddy-Scheduling-App/

```
|— app.py
|— requirements.txt
|— templates/
    |— base.html
    |— index.html
    |— login.html
    |— register.html
    |— dashboard.html
    |— courses.html
    |— sessions.html
    |— create_session.html
```

#### requirements.txt:

```
Flask==2.3.3
Werkzeug==2.3.7
```

### Step 2: Install Dependencies

1. **Open Bash Console** (from Dashboard)

### Navigate to your project:

cd Study-Buddy-Scheduling-App

2.

### Install Flask:

pip3.10 install --user Flask==2.3.3 Werkzeug==2.3.7

3.

## Step 3: Create Web App

1. Go to Web tab → Create a new web app
2. Choose "Manual configuration"
3. Select Python 3.10

## Step 4: Configure WSGI File

1. Click on WSGI configuration file link
2. Delete all content and replace with:

```
import sys
```

```
# Add your project directory to Python path
```

```
path = '/home/yourusername/Study-Buddy-Scheduling-App'
```

```
if path not in sys.path:
```

```
    sys.path.insert(0, path)
```

```
from app import app as application
```

```
if __name__ == "__main__":
```

```
    application.run()
```

Replace **yourusername** with your actual PythonAnywhere username!

## Step 5: Initialize Database

In Bash console, run:

cd Study-Buddy-Scheduling-Apppython3.10 app.py

- 1.
2. Press **Ctrl+C** to stop after it says "Running on..."

## Step 6: Test Your App

1. Go to Web tab
2. Click "Reload" button
3. Visit your app at: <https://yourusername.pythonanywhere.com>

## Troubleshooting

### Common Issues:

#### "Import Error":

- Make sure Flask is installed: `pip3.10 install --user Flask`
- Check WSGI file path matches your directory

#### "Template Not Found":

- Make sure templates folder exists
- Check all HTML files are uploaded

#### "Database Error":

- Run `python3.10 app.py` once to initialize database
- Check file permissions

### Check Logs:

- Web tab → Log files → Error log

## File Structure Checklist

Make sure you have:

- ☒ `app.py` (main Flask app)
- ☒ `requirements.txt`
- ☒ `templates/base.html`
- ☒ `templates/index.html`
- ☒ `templates/login.html`
- ☒ `templates/register.html`
- ☒ `templates/dashboard.html`
- ☒ `templates/courses.html`
- ☒ `templates/sessions.html`

- ☒ `templates/create_session.html`

## Test Features

Once deployed, test:

1. ☒ Homepage loads
2. ☒ User registration
3. ☒ User login
4. ☒ Course enrollment
5. ☒ Session creation
6. ☒ Session joining

Your app should be live at: <https://yourusername.pythonanywhere.com>

```
'<!-- templates/index.html -->
{% extends "base.html" %}
{% block title %}Home{% endblock %}
{% block content %}
<div class="row">
  <div class="col-md-8 mx-auto text-center">
    <div class="hero-section py-5">
      <h1 class="display-4 mb-4">
        <i class="fas fa-graduation-cap text-primary"></i>
        Study Buddy
      </h1>
      <p class="lead mb-4">Connect with Clemson classmates and organize study
sessions.</p>

    <div class="row mt-5">
      <div class="col-md-4">
        <div class="card h-100">
          <div class="card-body text-center">
            <i class="fas fa-users fa-3x text-primary mb-3"></i>
            <h5>Find Study Partners</h5>
            <p>Connect with classmates in your courses.</p>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card h-100">
          <div class="card-body text-center">
```

```

        <i class="fas fa-calendar-alt fa-3x text-success mb-3"></i>
        <h5>Schedule Sessions</h5>
        <p>Organize study sessions with flexible scheduling.</p>
    </div>
</div>
<div>
<div class="col-md-4">
    <div class="card h-100">
        <div class="card-body text-center">
            <i class="fas fa-book fa-3x text-info mb-3"></i>
            <h5>Course Integration</h5>
            <p>Manage all your courses in one place.</p>
        </div>
    </div>
</div>
</div>
</div>

<div class="mt-5">
    <a href="{{ url_for('register') }}" class="btn btn-primary btn-lg me-3">
        <i class="fas fa-user-plus"></i> Get Started
    </a>
    <a href="{{ url_for('login') }}" class="btn btn-outline-primary btn-lg">
        <i class="fas fa-sign-in-alt"></i> Sign In
    </a>
</div>
</div>
</div>
{% endblock %}

<!-- templates/login.html -->
{% extends "base.html" %}
{% block title %}Login{% endblock %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header">
                <h4><i class="fas fa-sign-in-alt"></i> Login to Study Buddy</h4>
            </div>
            <div class="card-body">
                <form method="POST">
                    <div class="mb-3">
                        <label for="username" class="form-label">Username</label>

```

```

        <input type="text" class="form-control" id="username" name="username"
required>
    </div>
    <div class="mb-3">
        <label for="password" class="form-label">Password</label>
        <input type="password" class="form-control" id="password" name="password"
required>
    </div>
    <div class="d-grid">
        <button type="submit" class="btn btn-primary">
            <i class="fas fa-sign-in-alt"></i> Login
        </button>
    </div>
</form>
<div class="text-center mt-3">
    <p>Don't have an account? <a href="{{ url_for('register') }}">Sign up here</a></p>
</div>
</div>
</div>
</div>
</div>
{% endblock %}

```

```

<!-- templates/register.html -->
{% extends "base.html" %}
{% block title %}Register{% endblock %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header">
                <h4><i class="fas fa-user-plus"></i> Join Study Buddy</h4>
            </div>
            <div class="card-body">
                <form method="POST">
                    <div class="mb-3">
                        <label for="username" class="form-label">Username</label>
                        <input type="text" class="form-control" id="username" name="username"
required>
                    </div>
                    <div class="mb-3">
                        <label for="email" class="form-label">Email Address</label>
                        <input type="email" class="form-control" id="email" name="email" required>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}

```



```

        <div class="mb-3">
            <label for="password" class="form-label">Password</label>
            <input type="password" class="form-control" id="password" name="password"
required>
        </div>
        <div class="d-grid">
            <button type="submit" class="btn btn-primary">
                <i class="fas fa-user-plus"></i> Create Account
            </button>
        </div>
    </form>
    <div class="text-center mt-3">
        <p>Already have an account? <a href="{{ url_for('login') }}">Sign in here</a></p>
    </div>
</div>
</div>
</div>
</div>
{% endblock %}

<!-- templates/dashboard.html -->
{% extends "base.html" %}
{% block title %}Dashboard{% endblock %}
{% block content %}
<div class="d-flex justify-content-between align-items-center mb-4">
    <h1>Welcome back, {{ session.username }}!</h1>
    <a href="{{ url_for('create_session') }}" class="btn btn-primary">
        <i class="fas fa-plus"></i> Create Study Session
    </a>
</div>

<div class="row">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header">
                <h5><i class="fas fa-book text-primary"></i> Your Courses ({{ user_courses|length
}})</h5>
            </div>
            <div class="card-body">
                {% if user_courses %}
                {% for course in user_courses %}
                <div class="d-flex justify-content-between align-items-center border-bottom py-2">
                    <div>
                        <strong>{{ course.course_code }}</strong><br>

```

```

        <small class="text-muted">{{ course.course_name }}</small>
    </div>
    <span class="badge bg-secondary">{{ course.department }}</span>
</div>
{% endfor %}
<div class="mt-3">
    <a href="{{ url_for('courses') }}" class="btn btn-outline-primary btn-sm">
        <i class="fas fa-plus"></i> Manage Courses
    </a>
</div>
{% else %}
<div class="text-center py-4">
    <i class="fas fa-users fa-3x text-muted mb-3"></i>
    <p class="text-muted">No upcoming study sessions.</p>
    <a href="{{ url_for('create_session') }}" class="btn btn-success">
        <i class="fas fa-plus"></i> Create First Session
    </a>
</div>
{% endif %}
</div>
</div>
</div>
</div>

<!-- Quick Stats -->
<div class="row mt-4">
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-book fa-2x text-primary mb-2"></i>
                <h4>{{ user_courses|length }}</h4>
                <p>Enrolled Courses</p>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card text-center">
            <div class="card-body">
                <i class="fas fa-calendar-check fa-2x text-success mb-2"></i>
                <h4>{{ upcoming_sessions|length }}</h4>
                <p>Upcoming Sessions</p>
            </div>
        </div>
    </div>
</div>

```

```

<div class="col-md-3">
  <div class="card text-center">
    <div class="card-body">
      <i class="fas fa-users fa-2x text-info mb-2"></i>
      <h4>0</h4>
      <p>Study Partners</p>
    </div>
  </div>
</div>
<div class="col-md-3">
  <div class="card text-center">
    <div class="card-body">
      <i class="fas fa-trophy fa-2x text-warning mb-2"></i>
      <h4>0</h4>
      <p>Sessions Completed</p>
    </div>
  </div>
</div>
</div>
{% endblock %}

<!-- templates/courses.html -->
{% extends "base.html" %}
{% block title %}Courses{% endblock %}
{% block content %}
<div class="d-flex justify-content-between align-items-center mb-4">
  <h1>Course Management</h1>
  <span class="badge bg-primary">{{ enrolled_course_ids|length }} Enrolled</span>
</div>

<div class="card">
  <div class="card-header">
    <h5><i class="fas fa-book"></i> Available Courses</h5>
  </div>
  <div class="card-body">
    <div class="row">
      {% for course in all_courses %}
        <div class="col-md-6 mb-3">
          <div class="card {% if course.id in enrolled_course_ids %}border-success{% endif
%}">
            <div class="card-body">
              <div class="d-flex justify-content-between align-items-start">
                <div>
                  <h6>

```

```

        {{ course.course_code }}
        {% if course.id in enrolled_course_ids %}
            <span class="badge bg-success ms-2">Enrolled</span>
        {% endif %}
    </h6>
    <p>
        <strong>{{ course.course_name }}</strong><br>
        <small class="text-muted">{{ course.department }}</small>
    </p>
</div>
<div>
    {% if course.id in enrolled_course_ids %}
        <a href="{{ url_for('drop_course', course_id=course.id) }}"
            class="btn btn-outline-danger btn-sm"
            onclick="return confirm('Are you sure?')">
            <i class="fas fa-minus"></i> Drop
        </a>
    {% else %}
        <a href="{{ url_for('enroll_course', course_id=course.id) }}"
            class="btn btn-primary btn-sm">
            <i class="fas fa-plus"></i> Enroll
        </a>
    {% endif %}
</div>
</div>
</div>
</div>
    {% endfor %}
</div>
</div>
{% endblock %}

<!-- templates/sessions.html -->
{% extends "base.html" %}
{% block title %}Study Sessions{% endblock %}
{% block content %}
<div class="d-flex justify-content-between align-items-center mb-4">
    <h1>Study Sessions</h1>
    <a href="{{ url_for('create_session') }}" class="btn btn-primary">
        <i class="fas fa-plus"></i> Create Session
    </a>
</div>

```

```

{% if sessions %}
  {% for session in sessions %}
    <div class="card study-session-card mb-3">
      <div class="card-body">
        <div class="row">
          <div class="col-md-8">
            <h5>{{ session.title }}</h5>
            <h6 class="text-muted">{{ session.course_code }} - {{ session.course_name
}}</h6>
            <p>
              {% if session.description %}{{ session.description }}<br>{% endif %}
              <i class="fas fa-user"></i> Created by {{ session.creator_name }}<br>
              <i class="fas fa-calendar"></i> {{ session.session_date }} at {{
session.session_time }}<br>
              <i class="fas fa-clock"></i> {{ session.duration }} minutes<br>
              {% if session.location %}<i class="fas fa-map-marker-alt"></i> {{ session.location
}}<br>{% endif %}
              <i class="fas fa-users"></i> {{ session.participant_count }}/{{
session.max_participants }} participants
            </p>
          </div>
          <div class="col-md-4 text-end">
            <span class="badge bg-success mb-2">{{ session.status|title }}</span><br>

            {% if session.is_creator %}
              <span class="badge bg-warning mb-2">Your Session</span><br>
              <button class="btn btn-outline-secondary btn-sm" disabled>
                <i class="fas fa-cog"></i> Manage
              </button>
            {% else %}
              {% if session.user_status == 'confirmed' %}
                <a href="{{ url_for('leave_session', session_id=session.id) }}"
                  class="btn btn-outline-danger btn-sm"
                  onclick="return confirm('Leave this session?')">
                  <i class="fas fa-sign-out-alt"></i> Leave
                </a>
              {% elif session.participant_count < session.max_participants %}
                <a href="{{ url_for('join_session', session_id=session.id) }}"
                  class="btn btn-success btn-sm">
                  <i class="fas fa-plus"></i> Join
                </a>
              {% else %}
                <button class="btn btn-secondary btn-sm" disabled>

```

```

                <i class="fas fa-users"></i> Full
            </button>
        {% endif %}
    {% endif %}
</div>
</div>
</div>
</div>
{% endfor %}
{% else %}
<div class="card">
    <div class="card-body text-center py-5">
        <i class="fas fa-users fa-3x text-muted mb-3"></i>
        <h5>No Study Sessions Available</h5>
        <p>There are no upcoming study sessions for your enrolled courses.</p>
        <a href="{{ url_for('create_session') }}" class="btn btn-primary">
            <i class="fas fa-plus"></i> Create First Session
        </a>
    </div>
</div>
</div>
{% endif %}
{% endblock %}

<!-- templates/create_session.html -->
{% extends "base.html" %}
{% block title %}Create Study Session{% endblock %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card">
            <div class="card-header">
                <h4><i class="fas fa-plus"></i> Create New Study Session</h4>
            </div>
            <div class="card-body">
                <form method="POST">
                    <div class="row">
                        <div class="col-md-6 mb-3">
                            <label for="course_id" class="form-label">Course *</label>
                            <select class="form-select" id="course_id" name="course_id" required>
                                <option value="">Select a course</option>
                                {% for course in user_courses %}
                                    <option value="{{ course.id }}">{{ course.course_code }} - {{
course.course_name }}</option>
                                {% endfor %}
                            </select>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}

```

```

        </select>
    </div>
    <div class="col-md-6 mb-3">
        <label for="title" class="form-label">Session Title *</label>
        <input type="text" class="form-control" id="title" name="title"
            placeholder="e.g., Midterm Prep Session" required>
    </div>
</div>

<div class="mb-3">
    <label for="description" class="form-label">Description</label>
    <textarea class="form-control" id="description" name="description" rows="3"
        placeholder="What will you study? Any specific topics?"></textarea>
</div>

<div class="row">
    <div class="col-md-4 mb-3">
        <label for="session_date" class="form-label">Date *</label>
        <input type="date" class="form-control" id="session_date"
name="session_date" required>
    </div>
    <div class="col-md-4 mb-3">
        <label for="session_time" class="form-label">Time *</label>
        <input type="time" class="form-control" id="session_time"
name="session_time" required>
    </div>
    <div class="col-md-4 mb-3">
        <label for="duration" class="form-label">Duration (minutes) *</label>
        <select class="form-select" id="duration" name="duration" required>
            <option value="60">1 hour</option>
            <option value="90">1.5 hours</option>
            <option value="120">2 hours</option>
            <option value="180">3 hours</option>
        </select>
    </div>
</div>

<div class="row">
    <div class="col-md-8 mb-3">
        <label for="location" class="form-label">Location</label>
        <input type="text" class="form-control" id="location" name="location"
            placeholder="e.g., Cooper Library Study Room 203">
    </div>
    <div class="col-md-4 mb-3">

```

```

        <label for="max_participants" class="form-label">Max Participants *</label>
        <select class="form-select" id="max_participants" name="max_participants"
required>
            <option value="2">2 people</option>
            <option value="3">3 people</option>
            <option value="4" selected>4 people</option>
            <option value="5">5 people</option>
            <option value="6">6 people</option>
            <option value="8">8 people</option>
        </select>
    </div>
</div>

<div class="d-flex justify-content-end gap-2">
    <a href="{{ url_for('sessions') }}" class="btn btn-secondary">
        <i class="fas fa-times"></i> Cancel
    </a>
    <button type="submit" class="btn btn-primary">
        <i class="fas fa-plus"></i> Create Session
    </button>
</div>
</form>
</div>
</div>
</div>
</div>
</div>

<script>
    // Set minimum date to today
    document.getElementById('session_date').min = new Date().toISOString().split('T')[0];
</script>
{% endblock %}"text-center py-4">
    <i class="fas fa-book-open fa-3x text-muted mb-3"></i>
    <p class="text-muted">You haven't enrolled in any courses yet.</p>
    <a href="{{ url_for('courses') }}" class="btn btn-primary">
        <i class="fas fa-plus"></i> Browse Courses
    </a>
</div>
{% endif %}
</div>
</div>
</div>

<div class="col-md-6">

```



```

<div class="card">
  <div class="card-header">
    <h5><i class="fas fa-calendar-alt text-success"></i> Upcoming Sessions</h5>
  </div>
  <div class="card-body">
    {% if upcoming_sessions %}
    {% for session in upcoming_sessions %}
    <div class="card study-session-card mb-3">
      <div class="card-body">
        <h6>{{ session.title }}</h6>
        <p class="card-text">
          <strong>{{ session.course_code }}</strong> - {{ session.course_name
}}<br>
          <i class="fas fa-user"></i> {{ session.creator_name }}<br>
          <i class="fas fa-calendar"></i> {{ session.session_date }} at {{
session.session_time }}<br>
          <i class="fas fa-users"></i> {{ session.participant_count }}/{{
session.max_participants }} participants
        </p>
      </div>
    </div>
    {% endfor %}
    <a href="{{ url_for('sessions') }}" class="btn btn-outline-success btn-sm">
      <i class="fas fa-eye"></i> View All Sessions
    </a>
    {% else %}
    <div class

```

```

# Simple commands to set up your Study Buddy project
# Run these commands one by one in your repository root

```

```

# 1. Create directory structure

```

```

mkdir -p templates
mkdir -p static
mkdir -p agile-docs

```

```

# 2. Create requirements.txt

```

```

echo "Flask==2.3.3
Werkzeug==2.3.7" > requirements.txt

```

```

# 3. Copy app.py from the artifact above into your project root

```

```

# 4. Copy base.html template from the artifact above into templates/base.html

```

# 5. Create all other templates by copying from the "All HTML Templates" artifact

# 6. Create basic agile documentation

echo "# Product Backlog

## Sprint 1 (Week 1) - Foundation

\*\*Goal\*\*: Basic user authentication and course management

### User Stories:

- US-001: User Registration (3 pts) - DONE
- US-002: User Login/Logout (2 pts) - DONE
- US-003: Course Browsing (3 pts) - DONE
- US-004: Course Enrollment (2 pts) - DONE
- US-005: Basic Dashboard (3 pts) - DONE

### Sprint 2 (Week 2) - Study Sessions

\*\*Goal\*\*: Study session creation and management

### User Stories:

- US-006: Create Study Session (5 pts)
- US-007: View Available Sessions (3 pts)
- US-008: Join/Leave Sessions (3 pts)
- US-009: Session Details (2 pts)

### Sprint 3 (Week 3) - Polish & Demo

\*\*Goal\*\*: UI improvements and demo preparation

### User Stories:

- US-010: Responsive Design (3 pts)
- US-011: Error Handling (2 pts)
- US-012: Demo Preparation (3 pts)

## Definition of Done

- Feature works as described
- Code is committed to GitHub
- Manual testing completed
- No critical bugs" > agile-docs/product-backlog.md

echo "# Daily Standups - Sprint 1

## Day 1 - \$(date '+%Y-%m-%d')

\*\*Team Member 1:\*\*

- Yesterday: Project setup

- Today: Implement user authentication
- Blockers: None

**\*\*Team Member 2:\*\***

- Yesterday: Sprint planning
- Today: Create HTML templates
- Blockers: None

**## Day 2 - [Date]**

**\*\*Team Member 1:\*\***

- Yesterday:
- Today:
- Blockers:

**\*\*Team Member 2:\*\***

- Yesterday:
- Today:
- Blockers: " > agile-docs/standup-log.md

**# 7. Create .gitignore**

echo "# Python

\_\_pycache\_\_/

\*.pyc

\*.pyo

\*.pyd

.Python

\*.so

pip-log.txt

pip-delete-this-directory.txt

.tox

.coverage

.coverage.\*

.cache

nosetests.xml

coverage.xml

\*.cover

\*.log

.git

.mypy\_cache

**# Database**

\*.db

\*.sqlite3

```

# Environment
.env
.venv
env/
venv/

# IDE
.vscode/
.idea/
*.swp
*.swo

# OS
.DS_Store
Thumbs.db" > .gitignore

echo "✅ Basic project structure created!"
echo "📝 Next steps:"
echo "1. Copy app.py from the Flask artifact"
echo "2. Copy all HTML templates from the templates artifact"
echo "3. Test locally: python app.py"
echo "4. Deploy to PythonAnywhere using the setup guide"

```

### Evaluation:

This iteration gave us all of the code needed to complete the project including the setup guide again, that also had stuff about the github. Additionally, it gave us the foundations for testing and a spring board for our next sprint as by this point we had planned to do a sprint dedicated to coding, while the last two were focused on testing and if needed debugging. We also looked back on dependencies such as flask needed for setup.

### Input:

give me styles css content

### Output:

```
/* Study Buddy App - Custom Styles */
```

```
:root {
```

```

--clemson-orange: #ff6600;
--clemson-purple: #522d80;
--clemson-light: #fff5f0;
--dark-orange: #e55a00;
--light-gray: #f8f9fa;
--medium-gray: #6c757d;
--success-green: #28a745;
--info-blue: #17a2b8;
--warning-yellow: #ffc107;
}

/* General Styles */
body {
  background-color: var(--light-gray);
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
}

/* Navigation */
.navbar-brand {
  font-weight: bold;
  color: var(--clemson-orange) !important;
  font-size: 1.5rem;
  text-decoration: none;
}

.navbar-brand:hover {
  color: var(--dark-orange) !important;
}

.navbar {
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.nav-link {
  font-weight: 500;
  transition: color 0.3s ease;
}

.nav-link:hover {
  color: var(--clemson-orange) !important;
}

/* Buttons */

```

```

.btn-primary {
  background-color: var(--clemson-orange);
  border-color: var(--clemson-orange);
  font-weight: 500;
  transition: all 0.3s ease;
}

.btn-primary:hover,
.btn-primary:focus {
  background-color: var(--dark-orange);
  border-color: var(--dark-orange);
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(255, 102, 0, 0.3);
}

.btn-outline-primary {
  color: var(--clemson-orange);
  border-color: var(--clemson-orange);
  font-weight: 500;
}

.btn-outline-primary:hover {
  background-color: var(--clemson-orange);
  border-color: var(--clemson-orange);
  transform: translateY(-2px);
}

/* Cards */
.card {
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  border: none;
  border-radius: 10px;
  margin-bottom: 20px;
  transition: all 0.3s ease;
  overflow: hidden;
}

.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
}

.card-header {
  background: linear-gradient(135deg, var(--clemson-orange), var(--dark-orange));

```

```

    color: white;
    border-bottom: none;
    font-weight: 600;
}

.card-header h4,
.card-header h5 {
    margin: 0;
    color: white;
}

/* Study Session Cards */
.study-session-card {
    border-left: 5px solid var(--clemson-orange);
    background: linear-gradient(135deg, #ffffff 0%, #fafafa 100%);
}

.study-session-card:hover {
    border-left-color: var(--dark-orange);
}

/* Hero Section */
.hero-section {
    background: linear-gradient(135deg, var(--clemson-light) 0%, #ffffff 50%, var(--light-gray) 100%);
    border-radius: 15px;
    padding: 3rem 2rem;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
    position: relative;
    overflow: hidden;
}

.hero-section::before {
    content: "";
    position: absolute;
    top: -50%;
    right: -50%;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(255, 102, 0, 0.1) 0%, transparent 70%);
    animation: float 6s ease-in-out infinite;
}

@keyframes float {

```

```
    0%, 100% { transform: translateY(0px); }
    50% { transform: translateY(-20px); }
}

.display-4 {
    font-weight: 700;
    margin-bottom: 1.5rem;
}

.lead {
    font-size: 1.3rem;
    font-weight: 400;
    color: var(--medium-gray);
}

/* Text Colors */
.text-primary {
    color: var(--clemson-orange) !important;
}

.text-success {
    color: var(--success-green) !important;
}

.text-info {
    color: var(--info-blue) !important;
}

.text-warning {
    color: var(--warning-yellow) !important;
}

/* Badges */
.badge {
    font-weight: 500;
    padding: 0.5em 0.75em;
    border-radius: 6px;
}

.bg-primary {
    background-color: var(--clemson-orange) !important;
}

/* Forms */
```



```

.form-control:focus {
  border-color: var(--clemson-orange);
  box-shadow: 0 0 0 0.2rem rgba(255, 102, 0, 0.25);
}

.form-select:focus {
  border-color: var(--clemson-orange);
  box-shadow: 0 0 0 0.2rem rgba(255, 102, 0, 0.25);
}

.form-label {
  font-weight: 500;
  color: var(--medium-gray);
}

/* Alerts */
.alert {
  border-radius: 8px;
  border: none;
  font-weight: 500;
}

.alert-info {
  background: linear-gradient(135deg, var(--clemson-light), #ffffff);
  color: var(--dark-orange);
  border-left: 4px solid var(--clemson-orange);
}

/* Stats Cards */
.stats-card {
  text-align: center;
  padding: 1.5rem;
  background: linear-gradient(135deg, #ffffff 0%, #fafafa 100%);
  border-radius: 10px;
  transition: all 0.3s ease;
}

.stats-card:hover {
  background: linear-gradient(135deg, var(--clemson-light), #ffffff);
  transform: scale(1.05);
}

.stats-card i {
  margin-bottom: 1rem;
}

```

```
}
```

```
.stats-card h4 {  
  font-weight: 700;  
  color: var(--clemson-orange);  
  font-size: 2rem;  
  margin-bottom: 0.5rem;  
}
```

```
/* Course Cards */
```

```
.course-card {  
  transition: all 0.3s ease;  
  border-radius: 10px;  
  overflow: hidden;  
}
```

```
.course-card.border-success {  
  border-color: var(--success-green) !important;  
  border-width: 2px !important;  
  background: linear-gradient(135deg, #f8fff9 0%, #ffffff 100%);  
}
```

```
.course-card:hover {  
  transform: translateY(-3px);  
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);  
}
```

```
/* List Groups */
```

```
.list-group-item {  
  border: none;  
  border-bottom: 1px solid #eee;  
  padding: 1rem;  
  transition: background-color 0.3s ease;  
}
```

```
.list-group-item:hover {  
  background-color: var(--clemson-light);  
}
```

```
.list-group-item:last-child {  
  border-bottom: none;  
}
```

```
/* Icons */
```

```

.fas, .far {
    margin-right: 0.5rem;
}

.fa-3x {
    margin-bottom: 1rem;
    opacity: 0.7;
}

/* Loading Animation */
.loading {
    position: relative;
}

.loading::after {
    content: "";
    position: absolute;
    top: 50%;
    left: 50%;
    width: 20px;
    height: 20px;
    margin: -10px 0 0 -10px;
    border: 2px solid transparent;
    border-top: 2px solid var(--clemson-orange);
    border-radius: 50%;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

/* Footer */
.footer {
    background: linear-gradient(135deg, #343a40 0%, #495057 100%);
    color: white;
    padding: 2rem 0;
    margin-top: 3rem;
    text-align: center;
}

/* Responsive Design */
@media (max-width: 768px) {

```

```
.hero-section {
  padding: 2rem 1rem;
}

.display-4 {
  font-size: 2rem;
}

.lead {
  font-size: 1.1rem;
}

.card {
  margin-bottom: 1rem;
}

.btn-lg {
  width: 100%;
  margin-bottom: 1rem;
}
}

@media (max-width: 576px) {
  .container {
    padding-left: 10px;
    padding-right: 10px;
  }

  .hero-section {
    margin: 0 -10px;
    border-radius: 0;
  }

  .navbar-brand {
    font-size: 1.2rem;
  }

  .card-body {
    padding: 1rem;
  }
}

/* Utility Classes */
.shadow-sm {
```

```

    box-shadow: 0 2px 4px rgba(0,0,0,0.1) !important;
}

.shadow {
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1) !important;
}

.shadow-lg {
    box-shadow: 0 10px 15px rgba(0, 0, 0, 0.1) !important;
}

.border-orange {
    border-color: var(--clemson-orange) !important;
}

.bg-orange {
    background-color: var(--clemson-orange) !important;
}

.text-orange {
    color: var(--clemson-orange) !important;
}

/* Animation Classes */
.fade-in {
    animation: fadeIn 0.6s ease-in;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

.slide-up {
    animation: slideUp 0.6s ease-out;
}

@keyframes slideUp {
    from { transform: translateY(30px); opacity: 0; }
    to { transform: translateY(0); opacity: 1; }
}

/* Focus States for Accessibility */
.btn:focus,

```

```

.form-control:focus,
.form-select:focus {
    outline: 2px solid var(--clemson-orange);
    outline-offset: 2px;
}

/* Print Styles */
@media print {
    .navbar, .footer, .btn {
        display: none !important;
    }

    .card {
        box-shadow: none !important;
        border: 1px solid #ddd !important;
    }

    body {
        background: white !important;
    }
}

```

### Evaluation:

The very blunt input called for style sheets because we had none at the time and the app looked very bare without any sort of beautification. This is to make the interface more attractive to a user and more accessible by conveying information through colors and the like. For instance, green meaning enrolled in a course allowed the website to convey information quickly.

## Tests:

### Input:

I have a Study Buddy Scheduling App. I need to run some test to help me know if this thing works correctly. I was told this about distinct tests: Distinct tests are tests that consider my routine, boundary, and challenging cases. We're not going to have too many challenging cases for this project, so I wouldn't be concerned about identifying those. We do, however, have plenty of boundaries we must consider, and an even larger number of routine cases to consider. A good test plan considers all potential boundaries, and a few routines. Here is the code: (inserted [app.py](#) here)

### Output:

Input: import pytest

```

from app import app

@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.test_client() as client:
        yield client

def test_home_route(client):
    """Test the home page loads successfully"""
    response = client.get('/')
    assert response.status_code == 200
    assert b"Welcome" in response.data or b"<html" in response.data

def test_dashboard_route(client):
    """Test the dashboard page is reachable"""
    response = client.get('/dashboard')
    print("Status code:", response.status_code)
    print("Redirect target (if any):", response.headers.get('Location'))

    # Accept either 200 (OK) or 302 (redirect to login)
    assert response.status_code in (200, 302)

def test_nonexistent_route(client):
    """Test a route that doesn't exist returns 404"""
    response = client.get('/not-a-real-page')
    assert response.status_code == 404

```

### Evaluation:

Checking that the page loads successfully and returns html content as expected is the most important part of the test cases. After the page loads, if authenticated correctly, the page should show either the log in page for non-authenticated users, or the dashboard for returning users. Non-existent routes should give a 404 error.