**PROJECT TITLE: STUDY PLANNER WEBSITE.**

**Project Overview:**

**Introduction:**

The Smart Study Planner is a web-based application designed to assist students in managing their study routines effectively. The project integrates several tools like a task manager, note-taking system, calendar, timer, and a chatbot for real-time assistance. The objective is to help students streamline their study schedules, optimize their time, and provide helpful resources for better productivity and learning experiences.

**Objectives**

* To allow users to create and manage study sessions.
* To enable users to set and track study goals.
* To provide a weekly and monthly view of study schedules.
* To send reminders for upcoming study sessions.
* To allow users to review past study activities for better planning.

**Need of the Project:**

In today’s fast-paced educational environment, students face challenges in staying organized and managing their study time effectively. Existing solutions often require switching between multiple applications, which leads to inefficiency and wasted time. The Smart Study Planner addresses this gap by offering an all-in-one platform, enabling students to:

* Organize tasks with deadlines.
* Schedule and visualize activities using a calendar.
* Take and manage notes seamlessly.
* Use a timer to track and analyze study time.
* Resolve queries in real time with a friendly chatbot.

The project promotes a structured and stress-free approach to academics by providing essential tools in one platform.

**Problem Statement:**

The main problem is the lack of a centralized tool that integrates task management, note-taking, scheduling, time tracking, and academic support in one platform. Many students use separate tools for these functions, leading to fragmented information, reduced productivity, and difficulty in tracking progress. This project aims to provide an all-in-one solution to manage tasks, notes, schedules, study time, and queries efficiently.

**Features and Functionalities:**

This website offers a range of functionalities and features according to user requirements

* Feature of login for already registered users
* Feature of signup for new users
* Addition and deletion of tasks
* Addition and deletion of notes
* Feature of time to note the time interval
* Availability of calendar and time to note the time and date of upcoming tasks
* Feature of chatbot in order to solve the queries
* Working database for updating and retrieval of previously saved data.

**Languages and Framework Used:**

* HTML
* CSS
* JAVASCRIPT
* jQuery
* AngularJS
* Bootstrap
* FLASK in python (For Backend)
* MySQL (For Database)

**Implementation Code and Related Scenarios:**

# **HTML DOCUMENT:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Smart Study Companion</title>

    <link rel="stylesheet" href="/static/css/index.css">

    <link rel="stylesheet" href="/static/css/all.min.css">

    <script src="/static/js/form.js"></script>

</head>

<body>

    <header>

        <h1>Smart Study Companion.<br> Your Personal Study Assistant. We have what you need</h1>

        <img src="/static/images/laptop-pic.jpg" height="190px" width="300px" >

    </header>

    <!-- Add the login button to the navigation bar -->

        <nav class="navbar">

            <div class="logo">SmartStudyCompanion</div>

            <ul class="links">

                <li><a href="#tasks">Tasks</a></li>

                <li><a href="#calendar">Calendar</a></li>

                <li><a href="#chat">Chat</a></li>

                <li><a href="#notes">Notes</a></li>

                <li><a href="#timer">Timer</a></li>

                <li><button id="loginButton">Login</button></li>

            </ul>

        </nav>

<!-- Login/Signup Form -->

<div id="loginModal" class="modal">

    <div class="modal-content">

        <button id="cancelButton" onclick="cancelForm()">&times;</button>

        <h2 id="formTitle">Login</h2>

        <form id="loginForm" action="/login" method="POST" >

            <div id="part1">

                <!-- login form content -->

                <label for="username">Username:</label>

                <input type="text" id="username" name="username" required>

                <label for="password">Password:</label>

                <input type="password" id="password" name="password" required>

                <button type="submit">Login</button>

            </div>

            <div id="part2">

                <img src="/static/images/login-img.jpg" height="190px" width="300px">

            </div>

        </form>

        <form id="signupForm" action="/signup" method="POST" style="display: none;">

            <div id="part3">

            <!-- signup form content -->

            <input type="text" id="signupUsername" placeholder="Username" required>

            <input type="password" id="signupPassword" placeholder="Password" required>

            <input type="password" id="signupConfirmPassword" placeholder="Confirm Password" required>

            <button type="submit">Signup</button>

        </div>

            <div id="part4">

                <img src="/static/images/login-img.jpg" height="190px" width="300px">

            </div>

        </form>

        <button id="toggleForm" onclick="toggleForm()">Don't have an account? Signup</button>

    </div>

</div>

<div class="section">

    <section id="tasks">

        <h2>Task Manager</h2>

        <input type="text" id="taskInput" placeholder="Add a new task">

        <input type="date" id="taskDate">

        <input type="time" id="taskTime">

        <button onclick="addTask()">Add Task</button>

        <ul id="taskList"></ul>

    </section>

</div>

<div class="section">

    <section id="calendar">

        <h2>Calendar</h2>

        <div class="calendarContainer">

            <div class="calendar-header">

              <h2 id="month-year"></h2>

              <button id="prev-month">Prev</button>

              <button id="next-month">Next</button>

            </div>

            <div class="calendar-body">

              <table id="calendar-table">

                <thead>

                  <tr>

                    <th>Sun</th>

                    <th>Mon</th>

                    <th>Tue</th>

                    <th>Wed</th>

                    <th>Thu</th>

                    <th>Fri</th>

                    <th>Sat</th>

                  </tr>

                </thead>

                <tbody id="calendar-tbody">

                  <!-- calendar days will be generated here -->

                </tbody>

              </table>

            </div>

          </div>

    </section>

</div>

<div class="section">

    <section id="chat">

        <h2>Chat/Discussion Forum</h2>

        <div id="chatBox">

            <div id="messages"></div>

            <input type="text" id="chatInput" placeholder="Type a message">

            <button onclick="sendMessage()">Send</button>

        </div>

    </section>

</div>

<div class="section">

    <section id="notes">

        <h2>Notes</h2>

        <!-- Form to add a new note -->

<input type="text" id="noteTitle" placeholder="Note Title">

<textarea id="noteContent" placeholder="Note Content"></textarea>

<button onclick="addNote()">Add Note</button>

<!-- List of notes -->

<ul id="noteList"></ul>

    </section>

</div>

<div class="section">

    <section id="timer">

        <h2>Timer</h2>

        <div id="timerDisplay">00:00:00</div>

        <button onclick="startTimer()">Start</button>

        <button onclick="stopTimer()">Stop</button>

        <button onclick="resetTimer()">Reset</button>

    </section>

</div>

    <footer class="footer">

        &copy; 2024 Smart Study Planner. All rights reserved.

    </footer>

    <script src="/static/js/script.js"></script>

</body>

</html>

# **CSS**

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 60px 0 0; /\* Add padding equal to the navbar height \*/

    background-color: #f4f4f9;

    color: #333;

}

header {

    color: white;

    padding: 100px 20px;

    text-align: center;

    background-image: url(/static/images/contrats-image.avif);

    background-size: cover;

    background-position: center;

    background-repeat: no-repeat;

    height: 300px;

    display: flex;

    flex-direction: row;

    justify-content: center;

    align-items: center;

    position: relative;

}

.logo{

    color: #fff;

}

.navbar {

    background-color: #00509e;

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px 20px;

    position: fixed; /\* Fix the navbar at the top \*/

    top: 0; /\* Align it to the top of the viewport \*/

    width: 100%; /\* Ensure it spans the full width \*/

    z-index: 1000; /\* Keep it above other elements \*/

}

.navbar ul {

    list-style: none;

    display: flex;

    margin: 0;

    padding: 0;

}

.navbar ul li {

    margin: 0 10px;

}

.navbar ul li a {

    text-decoration: none;

    color: white;

    padding: 5px 10px;

    border-radius: 5px;

    transition: background 0.3s;

}

.navbar ul li a:hover {

    background-color: #003f73;

}

.navbar .logo {

    font-size: 24px;

    font-weight: bold;

}

section {

    margin: 20px;

    padding: 20px;

    background: #fff;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

    border-radius: 8px;

}

h2 {

    text-align: center;

    color: #444;

}

#taskInput, #notesInput, #chatInput {

    width: calc(100% - 30px);

    padding: 10px;

    border: 1px solid #ddd;

    border-radius: 5px;

    margin-bottom: 10px;

}

button {

    text-decoration: none;

    padding: 5px 10px;

    background-color: #00509e;

    color: white;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    transition: background 0.3s;

}

button:hover {

    background-color: #003f73;

}

#taskList, #notesList, #messages {

    list-style-type: none;

    padding: 0;

}

#taskList li, #notesList div, #messages div {

    padding: 10px;

    margin: 5px 0;

    background: #f9f9f9;

    border: 1px solid #ddd;

    border-radius: 5px;

    display: flex;

    justify-content: space-between;

    align-items: center;

}

#calendarContainer {

    display: flex;

    justify-content: center;

    align-items: center;

    padding: 20px;

    background: #f4f4f9;

    border: 1px solid #ddd;

    border-radius: 5px;

}

#timerDisplay {

    font-size: 24px;

    font-weight: bold;

    margin-bottom: 10px;

    text-align: center;

}

.footer {

    text-align: center;

    padding: 10px;

    background: #0066cc;

    color: white;

    margin-top: 20px;

}

/\* Login/Signup Form Styles \*/

.modal {

    display: none; /\* Hidden by default \*/

    position: fixed; /\* Stay in place \*/

    z-index: 1; /\* Sit on top \*/

    left: 0;

    top: 0;

    width: 100%; /\* Full width \*/

    height:100%; /\* Full height \*/

    overflow: auto; /\* Enable scroll if needed \*/

    background-color: rgb(0,0,0); /\* Fallback color \*/

    background-color: rgba(0,0,0,0.4); /\* Black w/ opacity \*/

    padding-top: 60px;

    text-align: center;

  }

  .modal-content {

    background-color: #fefefe;

    margin: 15% auto; /\* 15% from the top and centered \*/

    padding: 20px;

    border: 1px solid #888;

    width: 80%; /\* Could be more or less, depending on screen size \*/

    height: 300px;

  }

#cancelButton .fa-times {

    font-family: 'FontAwesome';

    content: "\f00d"; /\* Unicode character for "X" \*/

  }

  .loginForm{

    display: flex;

    flex-direction: column;

    justify-content: center;

  }

  .signupForm{

    display: flex;

    flex-direction: column;

    justify-content: center;

  }

  .section {

  transition: transform 0.2s ease-in-out;

}

.section:hover {

  transform: scale(1.05);

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

#tasks {

  background-color: #f7f7f7;

  padding: 20px;

  border-radius: 10px;

  cursor: pointer;

}

#tasks:hover {

  background-color: #e7e7e7;

}

#calendar {

  background-color: #f2f2f2;

  padding: 20px;

  border-radius: 10px;

  cursor: pointer;

}

#calendar:hover {

  background-color: #e2e2e2;

}

#chat {

  background-color: #f5f5f5;

  padding: 20px;

  border-radius: 10px;

  cursor: pointer;

}

#chat:hover {

  background-color: #e5e5e5;

}

#notes {

  background-color: #f7f7f7;

  padding: 20px;

  border-radius: 10px;

  cursor: pointer;

}

#notes:hover {

  background-color: #e7e7e7;

}

#timer {

  background-color: #f2f2f2;

  padding: 20px;

  border-radius: 10px;

  cursor: pointer;

}

#timer:hover {

  background-color: #e2e2e2;

}

.section {

    transition: transform 0.2s ease-in-out;

  }

  .section:hover {

    transform: scale(1.05);

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

  }

  #tasks {

    background-color: #f7f7f7;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

  }

  #tasks:hover {

    background-color: #ffad16;

  }

  #calendar {

    background-color: #f2f2f2;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

  }

  #calendar:hover {

    background-color: #ffad16;

  }

  #chat {

    background-color: #f5f5f5;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

  }

  #chat:hover {

    background-color: #ffad16;

  }

  #notes {

    background-color: #f7f7f7;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

  }

  #notes:hover {

    background-color: #ffad16;

  }

  #timer {

    background-color: #f2f2f2;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

  }

  #timer:hover {

    background-color: #ffad16;

  }

.calendar-container {

    width: 800px;

    margin: 40px auto;

    background-color: #f7f7f7;

    padding: 20px;

    border: 1px solid #ddd;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  }

  .calendar-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin-bottom: 20px;

  }

  .calendar-header h2 {

    font-size: 24px;

    font-weight: bold;

    margin: 0;

  }

  .calendar-header button {

    background-color: #4CAF50;

    color: #fff;

    border: none;

    padding: 10px 20px;

    font-size: 16px;

    cursor: pointer;

  }

  .calendar-header button:hover {

    background-color: #3e8e41;

  }

  .calendar-body {

    overflow: auto;

  }

  #calendar-table {

    width: 100%;

    border-collapse: collapse;

  }

  #calendar-table th, #calendar-table td {

    border: 1px solid #ddd;

    padding: 10px;

    text-align: center;

  }

  #calendar-table th {

    background-color: #f0f0f0;

  }

  #loginModal {

    display: none; /\* Hide the login modal by default \*/

  }

# **Flask in Python**

from flask import Flask, request, jsonify, render\_template, session, redirect

from flask\_mysqldb import MySQL

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from datetime import datetime, timedelta

import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Database Configuration

app.config.from\_pyfile('config.py')

mysql = MySQL(app)

# Home Route

@app.route('/')

def index():

    if 'user\_id' in session:

        return render\_template('index.html')  # Show dashboard for logged-in users

    return redirect('/login')  # Redirect to login page

# User Registration

@app.route('/signup', methods=['POST'])

def signup():

    username = request.json['username']

    password = request.json['password']

    hashed\_password = generate\_password\_hash(password)

    try:

        cur = mysql.connection.cursor()

        cur.execute("INSERT INTO users (username, password) VALUES (%s, %s)", (username, hashed\_password))

        mysql.connection.commit()

        return jsonify({"message": "User registered successfully!"}), 201

    except Exception as e:

        return jsonify({"error": str(e)}), 400

# User Login

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

def login():

    if request.method == 'POST':

        if request.is\_json:

            data = request.get\_json()

            username = data.get('username')

            password = data.get('password')

            if username is None or password is None:

                return jsonify({"error": "Missing username or password"}), 400

            # Check if username exists in the database

            cur = mysql.connection.cursor()

            cur.execute("SELECT id, password FROM users WHERE username = %s", (username,))

            user = cur.fetchone()

            if user:

                user\_id, hashed\_password = user

                # Verify the password

                if check\_password\_hash(hashed\_password, password):

                    # Login successful

                    session['user\_id'] = user\_id

                    return jsonify({"message": "Login successful!"}), 200

                else:

                    return jsonify({"error": "Invalid username or password"}), 401

            else:

                return jsonify({"error": "User not found"}), 404

        return jsonify({"error": "Request must be JSON"}), 400

    return render\_template('index.html')

# Save Task

@app.route('/tasks', methods=['POST'])

def save\_task():

    if 'user\_id' not in session:

        return jsonify({"error": "Unauthorized"}), 403

    task = request.json['task']

    date = request.json['date']

    time = request.json['time']

    cur = mysql.connection.cursor()

    cur.execute("INSERT INTO tasks (user\_id, task, date, time) VALUES (%s, %s, %s, %s)",

                (session['user\_id'], task, date, time))

    mysql.connection.commit()

    return jsonify({"message": "Task saved!"}), 201

# Get Tasks

@app.route('/get\_tasks', methods=['GET'])

def get\_tasks():

    user\_id = session.get('user\_id')

    if user\_id:

        cursor = mysql.connection.cursor()

        cursor.execute("SELECT id, task, date, time FROM tasks WHERE user\_id = %s", (user\_id,))

        tasks = cursor.fetchall()

        tasks\_json = []

        for task in tasks:

            task\_dict = {

                "id": task[0],

                "task": task[1],

                "date": str(task[2]),  # Convert date to string

                "time": str(task[3])  # Convert time to string

            }

            tasks\_json.append(task\_dict)

        return jsonify(tasks\_json)

    return jsonify({"error": "User not logged in."}), 401

# Add Task

@app.route('/add\_task', methods=['POST'])

def add\_task():

    data = request.json

    user\_id = session.get('user\_id')  # Assuming user is logged in

    task = data.get('task')

    date = data.get('date')

    time = data.get('time')

    if user\_id and task and date and time:

        cursor = mysql.connection.cursor()

        cursor.execute("INSERT INTO tasks (user\_id, task, date, time) VALUES (%s, %s, %s, %s)", (user\_id, task, date, time))

        mysql.connection.commit()

        return jsonify({"message": "Task added successfully!"})

    return jsonify({"error": "Invalid input or user not logged in."}), 400

# Delete Task

@app.route('/delete\_task', methods=['POST'])

def delete\_task():

    data = request.json

    task\_id = data.get('task\_id')

    user\_id = session.get('user\_id')

    if task\_id and user\_id:

        cursor = mysql.connection.cursor()

        cursor.execute("DELETE FROM tasks WHERE id = %s AND user\_id = %s", (task\_id, user\_id))

        mysql.connection.commit()

        return jsonify({"message": "Task deleted successfully!"})

    return jsonify({"error": "Invalid input or user not logged in."}), 400

# Logout

@app.route('/logout', methods=['POST'])

def logout():

    session.pop('user\_id', None)

    return jsonify({"message": "Logged out successfully!"}), 200

# Task Reminder Notifications

@app.route('/reminders', methods=['GET'])

def reminders():

    if 'user\_id' not in session:

        return jsonify({"error": "Unauthorized"}), 403

    cur = mysql.connection.cursor()

    cur.execute("SELECT id, task, date, time FROM tasks WHERE user\_id=%s", [session['user\_id']])

    tasks = cur.fetchall()

    reminders = []

    now = datetime.now()

    for task in tasks:

        task\_date = datetime.strptime(f"{task[2]} {task[3]}", '%Y-%m-%d %H:%M:%S')

        if now + timedelta(hours=1) >= task\_date > now:

            reminders.append({"id": task[0], "task": task[1]})

    return jsonify(reminders), 200

# Task Deadline Notification Background Job

@app.route('/send\_notifications', methods=['POST'])

def send\_deadline\_notifications():

    cursor = mysql.connection.cursor()

    query = '''

        SELECT id, user\_id, task, date, time

        FROM tasks

        WHERE CONCAT(date, ' ', time) <= NOW() + INTERVAL 1 HOUR

          AND CONCAT(date, ' ', time) > NOW()

          AND notification\_sent = FALSE

    '''

    cursor.execute(query)

    tasks = cursor.fetchall()

    for task in tasks:

        user\_id = task[1]

        task\_name = task[2]

        # Example: send email or app notification to user

        send\_notification(user\_id, f"Upcoming Deadline: {task\_name}")

        # Update the notification\_sent flag

        update\_query = 'UPDATE tasks SET notification\_sent = TRUE WHERE id = %s'

        cursor.execute(update\_query, (task[0],))

    mysql.connection.commit()

    cursor.close()

    return jsonify({"message": "Notifications sent successfully!"}), 200

# Utility to Send Notifications

def send\_notification(user\_id, message):

    cursor = mysql.connection.cursor()

    cursor.execute('SELECT username FROM users WHERE id = %s', (user\_id,))

    user = cursor.fetchone()

    email = user[0]  # Assuming username is an email

    # Example: Log the notification instead of sending email

    print(f"Notification to {email}: {message}")

    cursor.close()

# Add Note

@app.route('/add\_note', methods=['POST'])

def add\_note():

    note\_data = request.json

    title = note\_data.get('title')

    content = note\_data.get('content')

    user\_id = session.get('user\_id')

    if not all([title, content, user\_id]):

        return jsonify({"error": "Missing required fields"}), 400

    # Validate input

    if user\_id and title and content:

        cursor = mysql.connection.cursor()

        cursor.execute("INSERT INTO notes (user\_id, title, content, created\_at) VALUES (%s, %s, %s, NOW())", (user\_id, title, content))

        mysql.connection.commit()

        return jsonify({"message": "Note added successfully!"})

    return jsonify({"error": "Invalid input or user not logged in."}), 400

# Get Notes

@app.route('/get\_notes', methods=['GET'])

def get\_notes():

    user\_id = session.get('user\_id')  # Get user\_id from session

    if user\_id:

        cursor = mysql.connection.cursor()

        cursor.execute("SELECT id, title, content, created\_at FROM notes WHERE user\_id = %s", (user\_id,))

        notes = cursor.fetchall()

        return jsonify([{

            "id": note[0],

            "title": note[1],

            "content": note[2],

            "created\_at": note[3].strftime('%Y-%m-%d %H:%M:%S')  # Formatting datetime for better readability

        } for note in notes])

    return jsonify({"error": "User not logged in."}), 401

# Delete Note

@app.route('/delete\_note', methods=['POST'])

def delete\_note():

    data = request.json

    note\_id = data.get('note\_id')  # Get note\_id from request

    user\_id = session.get('user\_id')  # Get user\_id from session

    if note\_id and user\_id:

        cursor = mysql.connection.cursor()

        cursor.execute("DELETE FROM notes WHERE id = %s AND user\_id = %s", (note\_id, user\_id))

        mysql.connection.commit()

        return jsonify({"message": "Note deleted successfully!"})

    return jsonify({"error": "Invalid input or user not logged in."}), 400

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**NOTE:** The JavaScript code is present in (( **website 3\static\js** )) folder of assignment so you can see it form there.

Also DEMO VIDEO and database images have been added in **Demo Video folder.**

**Conclusion**

The Smart Study Planner is a comprehensive solution that integrates essential tools to simplify academic life. By combining task management, note-taking, scheduling, time tracking, and real-time assistance, it addresses the critical challenges students face in managing their time and resources.

**Key Outcomes:**

* Enhanced productivity through effective organization and tracking.
* Simplified workflows by consolidating multiple functionalities into one platform.
* Improved user engagement with features like the chatbot.

**Future Possibilities:**

* Advanced analytics to track productivity trends and offer actionable insights.
* Integration with third-party tools like Google Calendar or Microsoft To-Do.
* Enhanced chatbot capabilities with AI for personalized assistance.

The Smart Study Planner has the potential to become an indispensable tool for students, driving academic success and fostering better study habits.