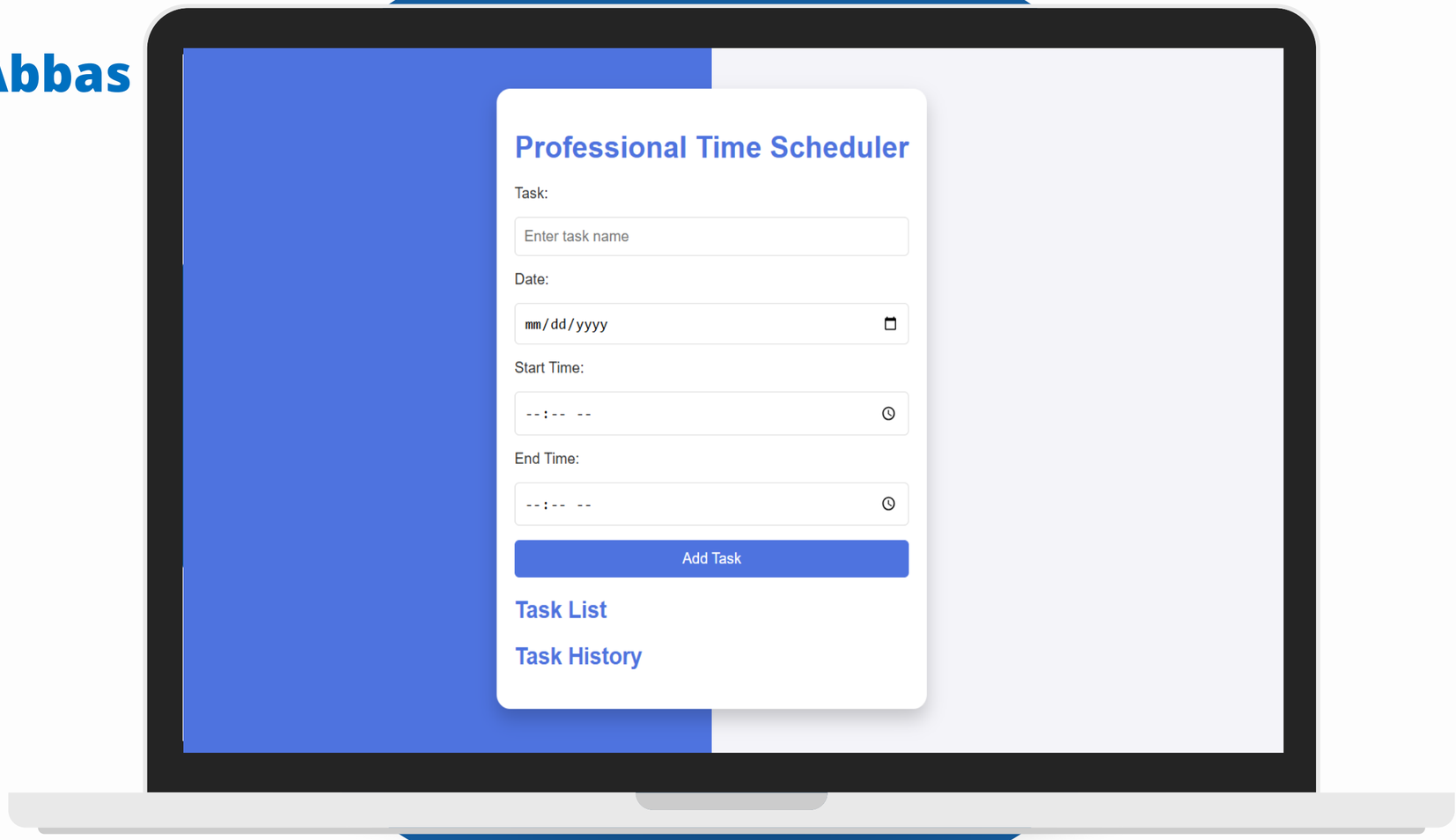
Title: Professional Time Scheduler

Presented by: Mufti Abbas

Date: 11/12/2024

Roll NO: 14725

Class: 3rd D (BScs)



Overview of the Project

The Professional Time Scheduler is a web-based task management application designed to help users manage their daily tasks efficiently. It supports task scheduling, reminders, and notifications using Flask for the backend and JavaScript for dynamic client-side functionality.

Introduction

The Professional Time Scheduler is a web-based application designed to help users organize and manage their daily tasks effectively. In today’s fast-paced world, managing time efficiently is crucial for personal and professional productivity. This application addresses this need by providing a platform where users can schedule tasks, set reminders, and receive notifications.

The project leverages modern web technologies, combining Flask for backend logic and JavaScript for dynamic frontend interactions. With features like task creation, alarm management, and real-time updates, the system ensures users stay on top of their responsibilities.

Objective:

- The primary goal is to create an intuitive and efficient task management system that:
- Simplifies task scheduling.
- Automates reminders and alarms.
- Provides real-time task updates.
- Ensures task persistence through database integration.

Technologies Used

HTML,css

Define the user interface layout and styling for responsiveness.

JS

Manages UI interactivity (e.g., adding tasks, setting alarms).Sends AJAX requests to the backend for CRUD operations.

Python Framework

Manages API routes for creating, reading, updating, and deleting tasks.

Handles alarm scheduling logic and task state management.

Code of project

```python
from flask import Flask, render_template, request, jsonify
import heapq
from datetime import datetime

app = Flask(__name__)

# In-memory priority queue for tasks
tasks = []
task_history = []

# Task counter for unique IDs
task_counter = 1

# Task class to store task details and its priority
class Task:
    def __init__(self, task_id, name, date, start_time, end_time, priority):
        self.id = task_id
        self.name = name
        self.date = date
        self.start_time = start_time
        self.end_time = end_time
        self.priority = priority  # Lower number means higher priority (earlier start time)
```

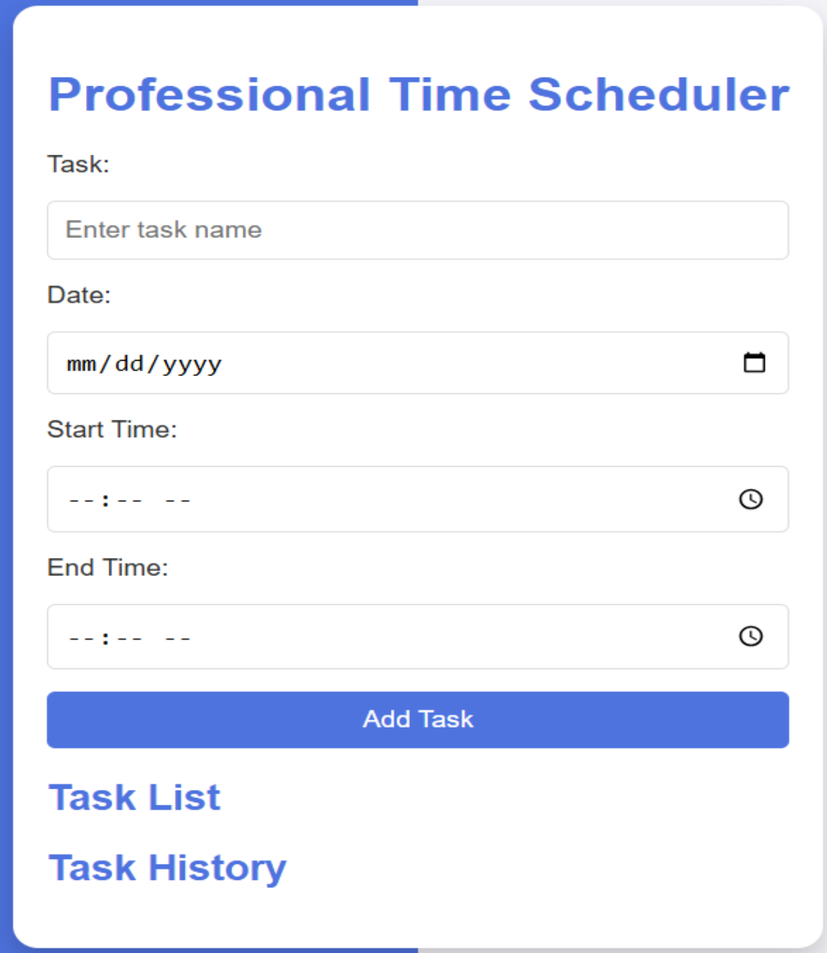
Code of project

```python
    def __lt__(self, other):
        return self.priority < other.priority

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/tasks', methods=['GET', 'POST'])
def tasks_route():
    global task_counter
    if request.method == 'GET':
        return jsonify([task.__dict__ for task in tasks])

    if request.method == 'POST':
        data = request.json
        task = Task(
            task_counter, data['taskName'], data['taskDate'], data['startTime'],
        )
        heapq.heappush(tasks, task)
        task_counter += 1
        return jsonify(task.__dict__)
```

Code of project

```python
@app.route('/tasks/<int:task_id>', methods=['GET', 'DELETE'])
def task_details(task_id):
    global tasks, task_history
    task = None
    for t in tasks:
        if t.id == task_id:
            task = t
            break

    if not task:
        return jsonify({'message': 'Task not found'}), 404

    if request.method == 'GET':
        task_history.append(task)
        tasks.remove(task)
        return jsonify({'message': 'Task completed', 'task': task.__dict__})

    if request.method == 'DELETE':
        tasks = [t for t in tasks if t.id != task_id]
        task_history = [t for t in task_history if t.id != task_id]
        return jsonify({'message': 'Task deleted'})

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

UI Design


Professional Time Scheduler
Task:
Enter task name
Date:
mm/dd/yyyy
Start Time:
--:-- --
End Time:
--:-- --
Add Task
Task List
Task History

THANK YOU!