CSE327: Term Project [Summer 2025]

Intelligent Task Planner for Students

Project Description

The Intelligent Task Planner for Students is a web-based productivity application designed to help university students effectively manage their academic workload and daily tasks. By inputting their assignments, classes, exams, deadlines, and available time, students receive AI-powered or rule-based scheduling recommendations to optimize their productivity.

The system will support priority-based task sorting, smart time slot recommendations, reminder notifications, and optionally, a Pomodoro timer and visual analytics to track time management efficiency. The system should also allow students to make manual changes to the schedule if necessary, while adapting intelligently over time.

Students can take ideas from the existing tools, such as TickTick.
https://ticktick.com/?language=en_US

This project must adhere to the Software Development Life Cycle (SDLC), guiding students through Requirement Analysis, System Design, Implementation, Testing, and Reporting.

Project Objectives

- **Develop** a user-friendly web application for students to input, track, and visualize tasks.
- **Implement** a rule-based or AI-assisted scheduling engine to recommend optimal task plans.
- **Design and implement** a relational database to store user data, tasks, and schedules.
- Integrate time-based reminders
- Google calendar syncing (optional).
- Add visualization features, such as daily and weekly charts, and time usage trends (optional).
- Ensure proper security, scalability, and performance for a multi-user environment.

Project Deliverables

1. Software Requirement Specification (SRS) Document

a. Introduction and Objectives

- Purpose and scope of the intelligent task planner.
- Stakeholders and expected benefits.

b. Functional and Non-functional Requirements

- Add/edit/delete tasks.
- Input available time blocks.

- Priority and deadline-based planning.
- Notifications and edits.
- System response time, usability, data privacy, etc.

c. System Use Case Diagram

- Actors: Student (User), System Scheduler (AI), Admin (Optional).
- Use cases: Create task, Get schedule, Update preferences, Track progress.

d. User Interface Mockups

- Task input form, calendar/schedule view, analytics dashboard.

e. Technology Stack

- Frontend: HTML, CSS, JS, React/Vue (optional)
- Backend: Django/Flask/Node.js
- DBMS: MySQL/PostgreSQL
- AI Scheduler (Optional): Python + Scheduling logic/ML

2. System Modeling and Design Diagrams

a. ER Diagram: Entities - User, Task, TimeSlot, Schedule, Settings.
b. Class Diagram: Classes - TaskManager, SchedulerEngine, UserProfile, NotificationService.
c. Sequence Diagram: Scenario - User submits task → System generates optimized schedule.
d. Activity Diagram: Task addition to schedule generation and visualization.
e. State Machine Diagram: Task lifecycle - Created → Scheduled → In-progress → Completed/Deleted.

3. Final Report and Project Presentation

a. Project Overview and Features: Summary of goals, user stories, and implemented functionalities.
b. Database Schema and System Architectures: Final database structure, overall system diagram (MVC/MVT pattern).
c. Implementation Challenges and Solutions: Resolving time-slot overlap, handling user edits, and ensuring UI responsiveness.
d. Implementation Architecture: Explanation of MVC/MVT and component integration.
e. Implementation Testing: Unit testing, UI testing, edge-case handling.
f. Project Scheduling: Utilize a Gantt chart or weekly progress tracking.
g. Project Budgeting: (If applicable) Assumptions about costs in a real-world scenario.
h. Future Enhancements: Sync with Google Calendar or Outlook, ML-based personalized time recommendations, Mobile app version.