TASK DEADLINES AND REMINDERS

Introduction

The project titled “Task Deadlines and Reminders” is designed to improve productivity and task management in any individual or team-based environment. It focuses on scheduling tasks, assigning deadlines, and providing automated reminders through real-time notifications and emails. The main goal is to help users stay organized, manage their workload efficiently, and ensure timely completion of responsibilities. In today’s fast-paced work and academic environments, missing important deadlines can cause significant disruptions. The Task Deadlines and Reminders system ensures that users receive timely alerts and reminders so that no task goes unnoticed or incomplete. The project integrates Flask (Python) as the backend framework, PostgreSQL for database management, and HTML, CSS, and JavaScript for the user interface.

Problem Identification

In most existing task management systems, users can create and manage tasks, but they often lack automated alert mechanisms for deadlines. Manual tracking of due dates becomes time-consuming and error-prone, especially when multiple projects or team members are involved. The absence of timely reminders results in missed deadlines, reduced productivity, and inefficient coordination. Common issues identified include: • Users forgetting to check pending tasks or nearing deadlines. • Lack of an automated reminder system to alert users in advance. • Managers unable to monitor overdue tasks effectively. • No synchronization between email alerts and system notifications.

Objectives

The primary objective of the Task Deadlines and Reminders project is to automate the monitoring and alerting process of tasks. It aims to reduce the need for manual follow-ups while increasing accountability among users. Key objectives include: 1. Provide automated alerts for approaching task deadlines. 2. Support both email and on-screen reminders for users. 3. Offer a user-friendly dashboard to view tasks by priority and deadline. 4. Ensure real-time synchronization of reminder notifications. 5. Allow administrators or team leads to track overdue and completed tasks.

System Requirements

Functional Requirements: • Create, edit, and delete tasks with deadline information. • Trigger automated reminders before and after due dates. • Enable both dashboard pop-up and email alerts. • Display all tasks sorted by due date. • Allow admin to assign and monitor tasks. **Non-Functional Requirements:** • Responsive and intuitive user interface. • Reliable notification delivery with minimal delay. • High scalability for multiple concurrent users. • Secure authentication and role-based access. • Cross-platform compatibility and database security.

System Design and Architecture

The architecture of the Task Deadlines and Reminders system follows a client-server model. Users interact through a web interface built using HTML, CSS, and JavaScript. The Flask backend handles business logic, database operations, and scheduling services. The system uses the following workflow: 1. A user creates a task with a defined title, description, and deadline. 2. Flask stores the task information in the PostgreSQL database. 3. APScheduler periodically checks for upcoming or missed deadlines. 4. When a deadline approaches, a reminder event is triggered. 5. Flask-Mail sends an email alert to the assigned user, while Socket.IO broadcasts a real-time notification to active users. 6. The reminder status and logs are stored in the database for auditing and history tracking. This architecture ensures efficient, reliable, and real-time communication between all components.

Tools & Technologies

The following technologies were used in the development of this project: • **Frontend:** HTML5, CSS3, JavaScript for building an interactive and responsive user interface. • **Backend:** Flask (Python) for managing routes, logic, and data flow. • **Database:** PostgreSQL for structured storage of task data, reminders, and logs. • **Scheduling:** APScheduler for managing timed events and periodic checks. • **Notifications:** Flask-Mail for sending email alerts and Socket.IO for real-time dashboard notifications. • **Version Control:** Git and GitHub for project management and collaboration.

Implementation Details

The implementation began with designing the database schema, including tables for users, tasks, and reminders. Each task entry includes attributes such as Task ID, Title, Description, Assigned User, Deadline, Reminder Time, and Status. The Flask application defines routes for task creation, editing, and listing, along with APIs for fetching reminders. APScheduler runs as a background process that scans for tasks nearing their deadlines. When a match is found, it triggers a reminder function that: • Sends an email notification using Flask-Mail. • Emits a live notification through Socket.IO. • Updates the reminder log in PostgreSQL. The front-end dashboard displays reminders in a dedicated “Upcoming Deadlines” section, with color-coded priorities (e.g., red for overdue, yellow for today, green for upcoming). This helps users visually identify urgent tasks and manage their schedule effectively.

Testing and Results

The system was thoroughly tested to ensure accuracy and reliability. Unit testing was performed on backend routes and scheduling functions to confirm that reminders triggered correctly. Integration testing verified that email notifications and dashboard alerts were delivered simultaneously without delay. Performance testing confirmed that the scheduler could handle multiple reminders efficiently. All functional and non-functional requirements were met successfully. **Results:** • 100% of test cases for task reminders passed successfully. • Average delay between scheduled time and alert delivery: less than 2 seconds. • No data loss observed during concurrent operations. • Improved user engagement and timely task completion rates.

Conclusion and Future Scope

The Task Deadlines and Reminders system efficiently automates deadline tracking and ensures users are always aware of their responsibilities. By combining database-driven task storage, real-time notifications, and scheduled alerts, it significantly enhances productivity and organization. **Future Scope:** • Integration with Google Calendar and Outlook for seamless synchronization. • Mobile app version with push notifications. • AI-based priority prediction to identify critical tasks automatically. • Analytics dashboard to track completion rates and user performance over time. This project serves as a scalable and practical solution for modern-day task management, ensuring no deadline is ever missed again.