Technical Documentation: University Events Platform SRMnHackHub

Problem Statement

University students often miss out on hackathons, workshops, and events due to a lack of streamlined communication. Critical information is scattered across multiple emails, resulting in missed deadlines and forgotten events. Moreover, students face challenges in identifying suitable technologies or resources for specific topics, particularly emerging fields.

Proposed Solution

To address these challenges, we developed a platform that centralizes university events such as hackathons, workshops, guest lectures, and seminars. Our solution allows students to filter events based on date and time, integrate their personal timetable, and receive notifications for events during their free slots. Additionally, the platform includes a project repository, skill assessment quizzes, and a chatbot for technical queries and guidance.

Implementation Details

Our platform is designed to solve the problem of scattered event communication by combining several key features:

Event Data Collection and Centralization

The foundation of the platform is its ability to gather and organize event information. We connected to the university's Gmail account using **imaplib** in Python, enabling the system to extract emails containing event details. The raw data extracted from emails is then cleaned and structured using **Google Generative AI** and **Charted**, ensuring consistency and accuracy. Once processed, the event data is stored in a **MongoDB** database. Events are displayed on the platform, categorized by type and date, to make browsing straightforward for students.

Timetable Integration and Event Notifications

Students can input their personal timetables into the platform. To ensure events align with their schedules, we implemented a backend algorithm using **Node.js** to cross-reference event timings with user timetables. If an event coincides with a user's free slot, they are notified via push notifications or email alerts. This ensures students are always informed about relevant opportunities without needing to constantly check the platform.

User-Friendly Event Filtering

The platform's frontend, created with **HTML**, **CSS**, **JavaScript**, and **Bootstrap**, provides an interactive and visually appealing interface. Users can filter events based on criteria such as date, time, and type. The filtering functionality is powered by APIs developed with **Express.js**, ensuring quick and efficient data retrieval from the database, even as the volume of events grows.

Project Repository Management

The project repository feature allows students to track their academic or personal projects. Users can log project details, including titles, descriptions, technologies used, and statuses. The backend, developed using **Node.js**, provides APIs that communicate with the **MongoDB** database to store and

retrieve this information. This centralized repository helps students monitor progress and maintain an organized record of their work.

Skill Assessment Module

To encourage skill development, we integrated a quiz module focused on coding and technical subjects. The quiz questions, along with options and correct answers, are stored in the **MongoDB** database. The frontend dynamically loads quiz content, allowing students to take assessments easily. After completing a quiz, the backend processes their responses using **Node.js**, calculates their scores, and provides immediate feedback. This gamified approach to learning makes technical skill development engaging and accessible.

Chatbot Integration for Technical Queries

To provide quick assistance, we integrated a chatbot using **Chatbase.co**. The chatbot is trained to answer frequently asked questions about technical topics, event details, and recommended tech stacks for specific themes like AI/ML. By embedding the chatbot into the platform, users can easily resolve common queries without needing external resources.

Challenges and Solutions

- **Email Data Cleaning**: Extracting relevant event information from emails was a significant challenge. We utilized **Google Generative AI** to effectively clean and structure the extracted data, making it ready for integration into the database.
- **Timetable Matching**: Ensuring accurate cross-referencing between user schedules and event timings required a robust logic. We addressed this by implementing an algorithm within the backend that efficiently matches user free slots with event times.
- **Chatbot Configuration**: Initially, configuring the chatbot to provide meaningful responses was complex. We overcame this by curating a dataset of FAQs and predefined answers, ensuring the chatbot effectively addressed user needs.

Conclusion

This platform provides a streamlined and centralized solution to the problem of managing university events. By integrating timetable management, personalized notifications, project tracking, skill assessments, and a chatbot, we created a comprehensive tool for academic and professional growth. The use of modern technologies like **Node.js**, **MongoDB**, and AI-based tools ensures scalability, efficiency, and ease of use, making it an essential platform for university students.