Hackathon Project: Chatting Website with Gemini Integration

Introduction

This project was developed for a hackathon with the goal of creating a functional chatting website using the Gemini API. The platform supports real-time communication between user and Gemini, file uploads, and features such as document summarization and question-answering using AI. This report outlines the key tasks completed during the development of the platform.

Team
VctrlC
Members
Sahana Athota
Vishrut Gurrala
Rishit Verma

Task 1: Basic Chat Website Implementation

Objective

The goal of this task was to build a real-time chatting platform where the user can exchange messages with Gemini. This includes integrating the gemini api with the website.

Implementation Overview

To achieve this, the following components were implemented:

1. **Frontend:**
    - **User Interface (UI):** A clean and intuitive UI was designed to ensure a smooth user experience. The interface includes a message input field, a list of messages, and notifications when new messages arrive.
    - **Real-time Updates:** Messages sent by users are immediately displayed in the chat window using real-time data handling.
2. **Backend:**
    - **Gemini API Integration:** When a user sends a message, it is sent to the backend, which forwards it to the Gemini API to either generate a response or send it to the intended recipient.
    - **Message Generation:** the backend communicates with the Gemini API to generate an appropriate response based on the input.
3. **API Interaction:**
    - **Message Handling:** User messages are received by the backend, where the Gemini API is called. If the message requires a response, the Gemini API processes the input and generates a reply.

Key Features

- **Real-time Messaging:** Ensures that messages sent are instantly visible to the user.
- **AI-driven Responses:** The Gemini API allows for intelligent responses based on user inputs when needed (e.g., automated responses, chatbot functionality).
- **Efficient Communication Handling:** User messages are processed by the backend, relayed to the appropriate recipient or API, and responses are immediately sent back to the frontend.

Technologies Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js with Express.js
- **API Integration:** Gemini API for processing and generating responses
- **Database:** MongoDB

Challenges Faced

- **API Integration:** Ensuring smooth interaction with the Gemini API to generate real-time responses with low latency.
- **Message Synchronisation:** Handling the relay of messages between users and the API while maintaining real-time updates.

Results

- The chatting website successfully allows users to interact with AI-generated responses through the Gemini API. The system efficiently handles real-time communication, ensuring smooth message delivery.

Task 2: File Upload Support

Objective

The goal of this task was to enable users to upload files in the chat and utilise the Gemini API to extract insights from the uploaded files. The platform allows users to get summaries of file content and ask questions related to the document, making the chat more interactive and useful for handling documents.

Implementation Overview

To implement this functionality, the following components were developed:

1. **Frontend:**
    - **File Upload Interface:** Users can upload files through a simple, user-friendly interface. The platform allows users to type the file path of the desired file.
    - **User Interaction:** After uploading a file, users can either request a summary or ask questions related to the content of the file. These requests are sent to the backend for processing by the Gemini API.
2. **Backend:**
    - **File Handling:** When a file is uploaded, the backend extracts the necessary content from the file and sends it to the Gemini API for further analysis.
    - **Gemini API Integration:** The Gemini API is used to:
        - **Summarise file Content:** The API processes the uploaded file and generates a concise summary of the content.
        - **Answer Questions:** Users can ask specific questions related to the content of the file, and the Gemini API returns relevant answers by analysing the document's content.
    - **Real-time Processing:** The file is processed and analysed in real-time, providing users with instant feedback on their queries or requests for summaries.
3. **API Interaction:**
    - **Summarization Request:** When a user uploads a file and requests a summary, the backend extracts the text from the file and sends it to the Gemini API, which returns a brief, accurate summary of the document.
    - **Question-Answering:** Users can ask follow-up questions based on the content of the file. The backend forwards the question along with the extracted text to the Gemini API, which analyses the file content to generate relevant responses.

Key Features

- **File Upload:** Users can upload files by giving the file path of the desired file.
- **File Summarization:** The Gemini API summarises the key points of the file, allowing users to quickly understand its contents.
- **Question-Answering:** Users can ask questions based on the file, and the Gemini API provides relevant answers by analysing the document.

Technologies Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js with Express.js
- **API Integration:** Gemini API for summarising and answering questions related to uploaded file content

Challenges Faced

- **Contextual Understanding:** Ensuring that the Gemini API provides relevant answers based on the context of the document and user queries.
- **Real-time Performance:** Maintaining low latency for real-time file summarization and question-answering to ensure a smooth user experience.

Results

- The chatting website now supports file uploads, with advanced functionality powered by the Gemini API. Users can quickly summarise documents and ask questions about the content, enhancing the productivity and interactivity of the platform.

Task 3: User Authentication

Objective

To ensure secure access to the chat platform, user authentication was implemented. This guarantees that only authorised users can interact with the system, upload files, or access chat features.

Implementation

1. **Authentication Mechanism:**
    - The authentication system allows users to register by providing a username and password. Both are stored in a MongoDB database.
    - During registration, the username and password are saved directly to the database without any encryption or hashing. This implementation focuses on simplicity for the hackathon, though it may not meet best security practices for production systems.
2. **Backend:**

    User Login Flow:

    - When users attempt to log in, the backend retrieves the corresponding user record from MongoDB based on the provided username.
    - The password submitted by the user is compared directly with the stored password in the database for validation.

Key Features

- **User Authentication:** Ensures that only registered users can access the chat and file upload features.

Challenges

- Implementing password storage and authentication flow.

Results

The platform now supports secure user authentication, preventing unauthorised access to the chat features and file upload system.

Conclusion

This project successfully delivered a real-time chatting website with advanced features such as PDF file uploads, summarization, and question-answering using the Gemini API. The combination of real-time messaging, file processing with the Gemini API, and secure user authentication ensures a robust and interactive platform. Each task presented unique challenges, which were overcome to create a fully functional and engaging user experience.