Chat – Bot

Machine – Learning assignment

W.A.S.L.WIJESINGHE

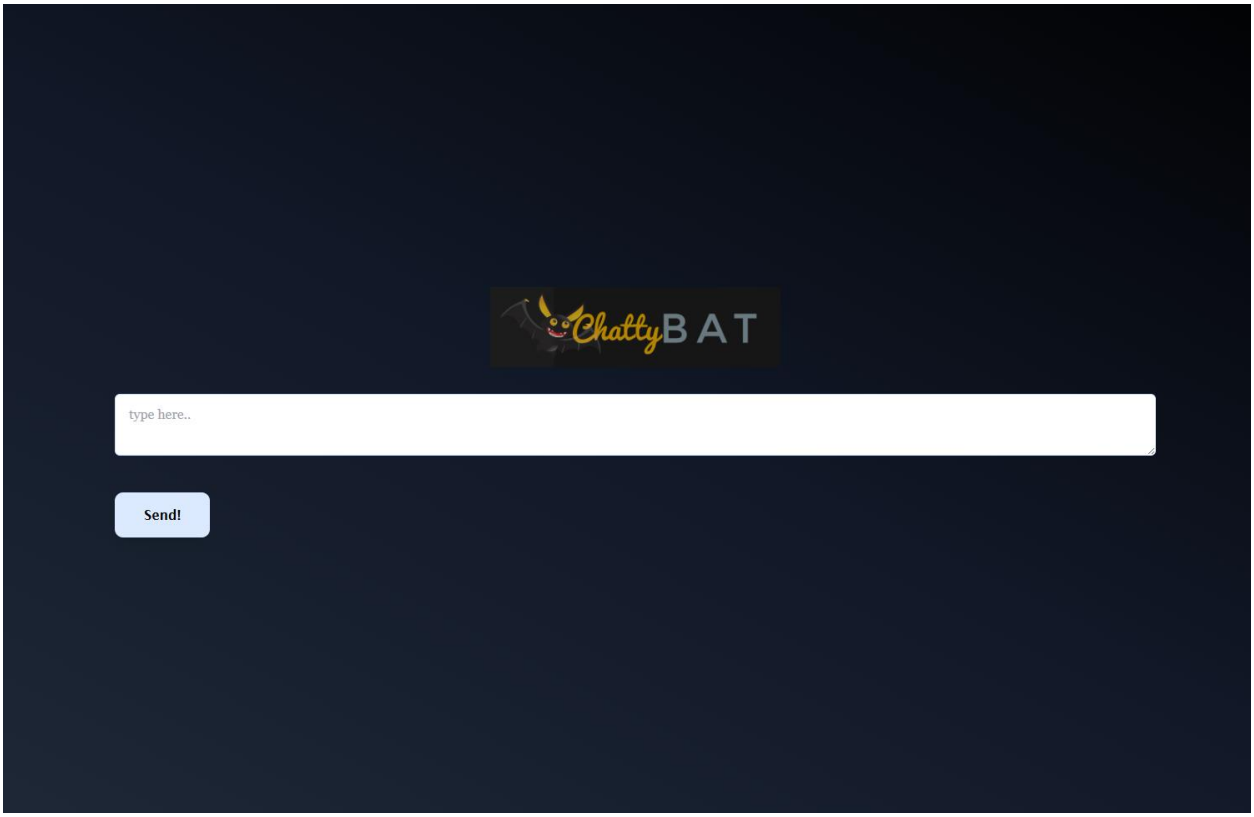
GAM - IT-2021-F-0101

Introduction

The AI ChatBot project aims to leverage Google's Gemini AI within a MERN stack environment to provide users with conversational interactions. Targeting individuals seeking quick and accurate responses, this chatbot integrates advanced natural language processing for seamless communication.

Design and Development Process

The design process began with thorough research on conversational AI frameworks and platforms. Leveraging the MERN stack - MongoDB, Express.js, React.js, and Node.js - provided a robust foundation for both frontend and backend development. Iterative prototyping and user testing informed the implementation of features for intuitive user interactions.



Features and Functionality

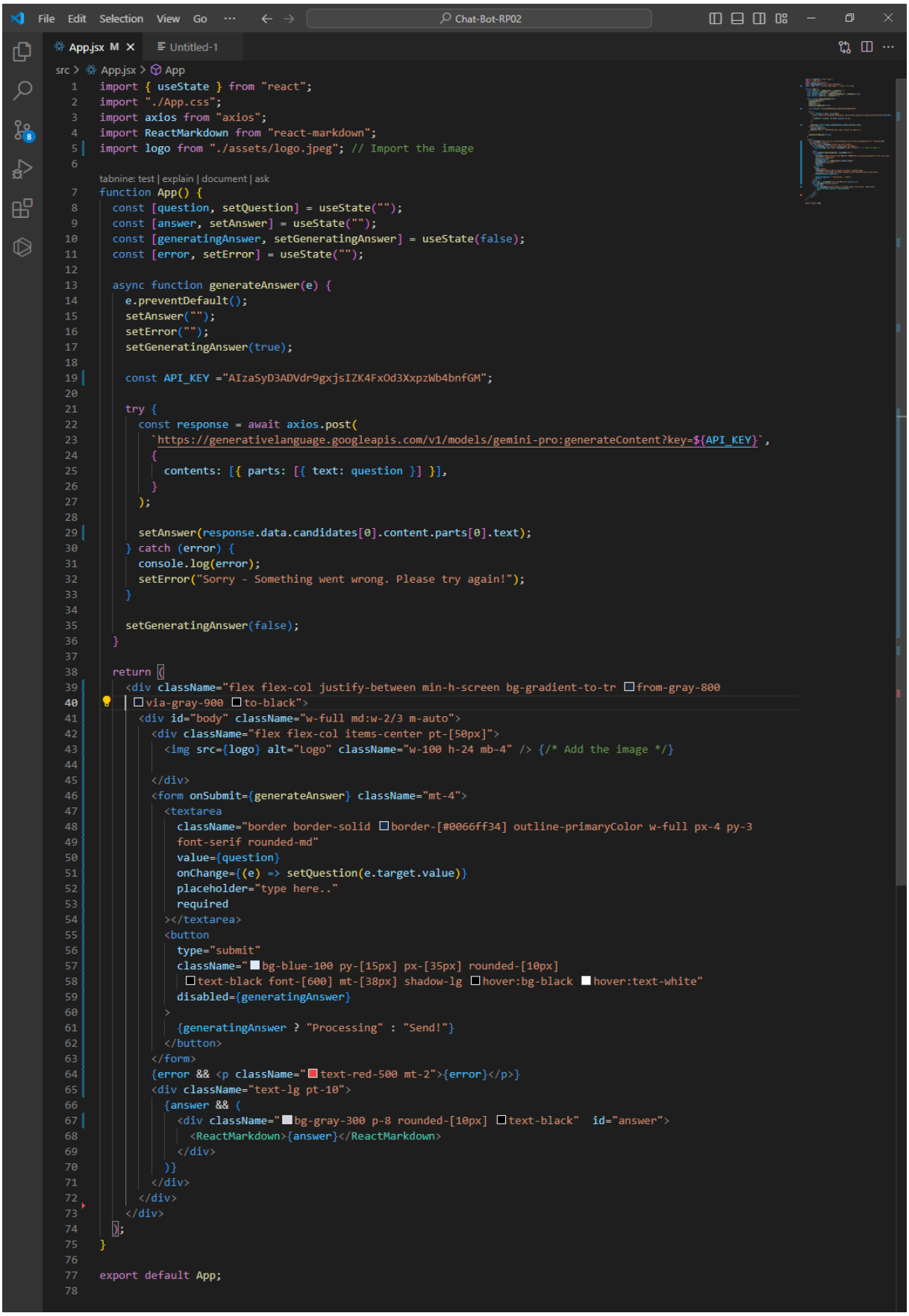
The AI ChatBot offers users a simple interface to input queries and receive responses generated by Google's Gemini AI. It leverages React for dynamic frontend rendering, Axios for seamless HTTP requests, and FontAwesome for visual elements. Key functionalities include real-time question generation and display of AI-generated responses.

User Experience

Designed with user-centric principles, the AI ChatBot ensures a seamless conversational experience. Its intuitive interface guides users through the query process, while feedback mechanisms and loading indicators enhance engagement. Conversational flow is prioritized to mimic human interactions, promoting user satisfaction.

Technical Details

Utilizing a MERN stack architecture, the AI ChatBot integrates Google's Gemini AI API for natural language processing. React.js facilitates responsive frontend design, while Express.js handles server-side logic. MongoDB serves as the database for storing user queries and responses, ensuring scalability and reliability.


File Edit Selection View Go ···
Chat-Bot-RP02
App.jsx M
Untitled-1
src > App.jsx > App
import { useState } from "react";
import "./App.css";
import axios from "axios";
import ReactMarkdown from "react-markdown";
import logo from "./assets/logo.jpeg"; // Import the image
tabnine: test | explain | document | ask
function App() {
const [question, setQuestion] = useState("");
const [answer, setAnswer] = useState("");
const [generatingAnswer, setGeneratingAnswer] = useState(false);
const [error, setError] = useState("");
async function generateAnswer(e) {
e.preventDefault();
setAnswer("");
setError("");
setGeneratingAnswer(true);
const API_KEY ="AIzaSyD3ADVdr9gxjsIZK4FxOd3XxpzWb4bnfGM";
try {
const response = await axios.post(
`https://generativelanguage.googleapis.com/v1/models/gemini-pro:generateContent?key=${API_KEY}`,
{
contents: [{ parts: [{ text: question }] }],
}
);
setAnswer(response.data.candidates[0].content.parts[0].text);
} catch (error) {
console.log(error);
setError("Sorry - Something went wrong. Please try again!");
}
setGeneratingAnswer(false);
}
return (
<div className="flex flex-col justify-between min-h-screen bg-gradient-to-tr from-gray-800
via-gray-900 to-black">

 {/* Add the image */}

<form onSubmit={generateAnswer} className="mt-4">
<textarea
className="border border-solid border-[#0066ff34] outline-primaryColor w-full px-4 py-3
font-serif rounded-md"
value={question}
onChange={(e) => setQuestion(e.target.value)}
placeholder="type here.."
required
></textarea>
<button
type="submit"
className="bg-blue-100 py-[15px] px-[35px] rounded-[10px]
text-black font-[600] mt-[38px] shadow-lg hover:bg-black hover:text-white"
disabled={generatingAnswer}
>
{generatingAnswer ? "Processing" : "Send!"}
</button>
</form>
{error && {error} }

{answer && (

<ReactMarkdown>{answer}</ReactMarkdown>

)}

);
}
export default App;

This code is a React component that implements a simple question-answering application using an external AI service for generating answers.

Here's a brief overview of its functionality:

1. Imports and Setup :

Essential React hooks (`useState`) and components are imported.

`axios` is used for making HTTP requests.

`ReactMarkdown` is used to render markdown content.

An image is imported for the logo.

2. State Management :

The component manages several pieces of state using `useState`:

`question` to store the user's input.

`answer` to store the response from the AI service.

`generatingAnswer` to indicate whether an answer is being generated.

`error` to handle any error messages.

3. Generating Answers :

`generateAnswer` is an asynchronous function that sends the user's question to an AI service using a POST request.

It handles the response and sets the `answer` state with the returned data.

It also manages error handling and sets the `error` state if something goes wrong.

4. Rendering the UI :

The component's JSX structure includes:

A container with a gradient background.

A section for displaying the logo.

A form with a `textarea` for the user to type their question.

A submit button that triggers the `generateAnswer` function.

Conditional rendering for showing processing status, errors, and the generated answer.

5. Styling :

Tailwind CSS classes are used extensively for styling elements such as the container, form, button, and text display areas.

Overall, this component provides a user interface for inputting a question, submitting it to an AI service,

and displaying the response, with appropriate error handling and loading indicators.

Future Enhancements

1. **User Authentication**:
    - Implement user authentication to allow personalized experiences and manage usage quotas for the AI service.
2. **Enhanced Error Handling**:
    - Improve error handling by providing more detailed error messages and recovery options (e.g., retrying the request automatically).
3. **Input Validation**:
    - Add input validation to ensure that questions meet certain criteria before sending them to the AI service (e.g., minimum length, no offensive language).
4.

Conclusion

This React component demonstrates a simple yet powerful way to integrate AI-driven question answering into a web application. By leveraging state management, asynchronous API calls, and conditional rendering, the component provides a responsive and user-friendly interface for interacting with an AI service.