**GENgo AI BOT Documentation**

**My Profile**

****

**Name:** Manoharan M,

**Role of this project :** Web developer,

**From :** DESIGN HEAD in Google developer groups at IIE

(INFO institute of engineering)

## About Us

**I am Manoharan**, currently pursuing a B.Tech in Information Technology, and I am in my second year of studies. As the Design Head of the Google Developer Group (GDG) at IEE, I am passionate about creating engaging and user-friendly designs that enhance the digital experience.

With a strong foundation in UI/UX design, I have successfully completed various projects, including brochures, social media posts, business card designs, and banners. My goal is to combine my technical knowledge with creative design to produce impactful and innovative solutions that meet user needs.

I believe that effective design is not just about aesthetics; it’s about understanding the user and providing them with intuitive experiences. I am excited to continue developing my skills and contributing to projects that push the boundaries of design and technology.

**GENgo AI BOT Documentation**

**Table of Contents**

1. [Project Overview](#project-overview)
2. [Features](#features)
3. [Technologies Used](#technologies-used)
4. [Installation](#installation)
5. [Usage](#usage)
6. [Components](#components)
   * [NavBar](#navbar)
   * [ChatBox](#chatbox)
   * [ButtonGrid](#buttongrid)
   * [About](#about)
   * [Home](#home)
7. [API Integration](#api-integration)
8. [Animations](#animations)
9. [Additional Libraries](#additional-libraries)
10. [Future Enhancements](#future-enhancements)
11. [Contributing](#contributing)
12. Link: <https://ge-ngo-ai-bot.vercel.app/>

**Project Overview**

GENgo AI BOT is an advanced chatbot developed by the GDG Team at IEE, led by Mano ,With help of GDG Mentors . It assists users by understanding and responding to their queries with high accuracy and relevance.

**Features**

* Interactive chatbot interface
* Predefined message buttons for quick actions
* Integration with Google Generative AI for dynamic responses
* Responsive design for various screen sizes
* Background video for an engaging user experience
* Smooth animations for enhanced user interaction

**Technologies Used**

* **Frontend**: React, Tailwind CSS
* **Backend**: Google Generative AI
* **Routing**: React Router
* **Icons**: React Icons
  + **React Icons** is a library that provides a collection of popular icons as React components. It allows for easy integration of icons throughout the application, enhancing the UI with visual elements without needing to manually import SVG files. This helps in maintaining consistency and reducing the overall size of the codebase.
* **React Markdown** for output text format
  + **react-markdown** is a library used for rendering Markdown content as React components. It allows for easy formatting of text, enabling the chatbot to respond with rich text that can include headings, lists, links, and other elements.

**Installation**

**Install Tailwind CSS with Vite**

Setting up Tailwind CSS in a Vite project.

1. **Create your project**

Start by creating a new Vite project if you don’t have one set up already. The most common approach is to use [**Create Vite**](https://vitejs.dev/guide/#scaffolding-your-first-vite-project).

Terminal

**npm create vite@latest my-project -- --template reactcd my-project**

1. **Install Tailwind CSS**

Install tailwindcss and its peer dependencies, then generate your tailwind.config.js and postcss.config.js files.

Terminal

**npm install -D tailwindcss postcss autoprefixernpx tailwindcss init -p**

1. **Configure your template paths**

Add the paths to all of your template files in your tailwind.config.js file.

tailwind.config.js

/\*\* @type {import('tailwindcss').Config} \*/

export default {

**content: [**

**"./index.html",**

**"./src/\*\*/\*.{js,ts,jsx,tsx}",**

**],**

theme: {

extend: {},

},

plugins: [],

}

1. **Add the Tailwind directives to your CSS**

Add the @tailwind directives for each of Tailwind’s layers to your ./src/index.css file.

**index.css**

**@tailwind base;**

**@tailwind components;**

**@tailwind utilities;**

1. **Start your build process**

Run your build process with npm run dev.

Terminal

**npm run dev**

1. **Start using Tailwind in your project**

Start using Tailwind’s utility classes to style your content.

**App.jsx**

**export default function App() {**

**return (**

**<h1 className="text-3xl font-bold underline">**

**Hello world!**

**</h1>**

**)**

**}**

**Usage**

1. Start the development server:

bash

Copy code

npm start

1. Open your browser and navigate to http://localhost:3000.

**Components**

**NavBar**

The NavBar component provides navigation links and branding.

**Props**

* None

**Example**

jsx

Copy code

<NavBar />

**ChatBox**

The ChatBox component allows users to interact with the AI by sending messages.

**Props**

* sendMessage: Function to handle sending messages.

**Example**

jsx

Copy code

<ChatBox sendMessage={sendMessage} />

**ButtonGrid**

The ButtonGrid component displays buttons for predefined actions.

**Props**

* sendMessage: Function to handle button clicks.

**Example**

jsx

Copy code

<ButtonGrid sendMessage={sendMessage} />

**About**

The About component provides information about the GENgo AI BOT.

**Example**

jsx

Copy code

<About />

**Home**

The Home component serves as the main interface for interacting with the chatbot.

**Example**

jsx

Copy code

<Home />

**API Integration**

The application integrates with Google Generative AI to fetch responses based on user input. The API key is stored securely and is used to initialize the AI model.

**Animations**

**Keyframe Animations**

* **Typing Dots Animation**: Creates an effect for loading messages, enhancing user experience during response times.

css

Copy code

@keyframes typingDots {

0%, 20% { opacity: 0; }

25% { opacity: 1; }

50% { opacity: 0; }

}

* **Glow Animation**: Applied to the input field when the bot is processing a response, indicating activity.

css

Copy code

@keyframes glow {

0% { box-shadow: 0 0 5px rgba(0, 123, 255, 0.6); }

50% { box-shadow: 0 0 15px rgba(0, 123, 255, 0.8); }

100% { box-shadow: 0 0 5px rgba(0, 123, 255, 0.6); }

}

**Usage in Components**

* The isLoading state in the ChatBox component triggers the loading animation when the bot is processing a response.
* The button components have hover effects to enhance interactivity.

**Additional Libraries**

**React Markdown**

react-markdown is a library used for rendering Markdown content as React components. It allows for easy formatting of text, enabling the chatbot to respond with rich text that can include headings, lists, links, and other elements.

**Installation**

To install react-markdown, run:

bash

Copy code

npm install react-markdown

**Usage**

In the ChatBox component, react-markdown is used to display the AI's responses:

jsx

Copy code

import ReactMarkdown from 'react-markdown';

<ReactMarkdown>{msg.text}</ReactMarkdown>

This allows users to see formatted responses, enhancing readability and user engagement.

**Other Libraries**

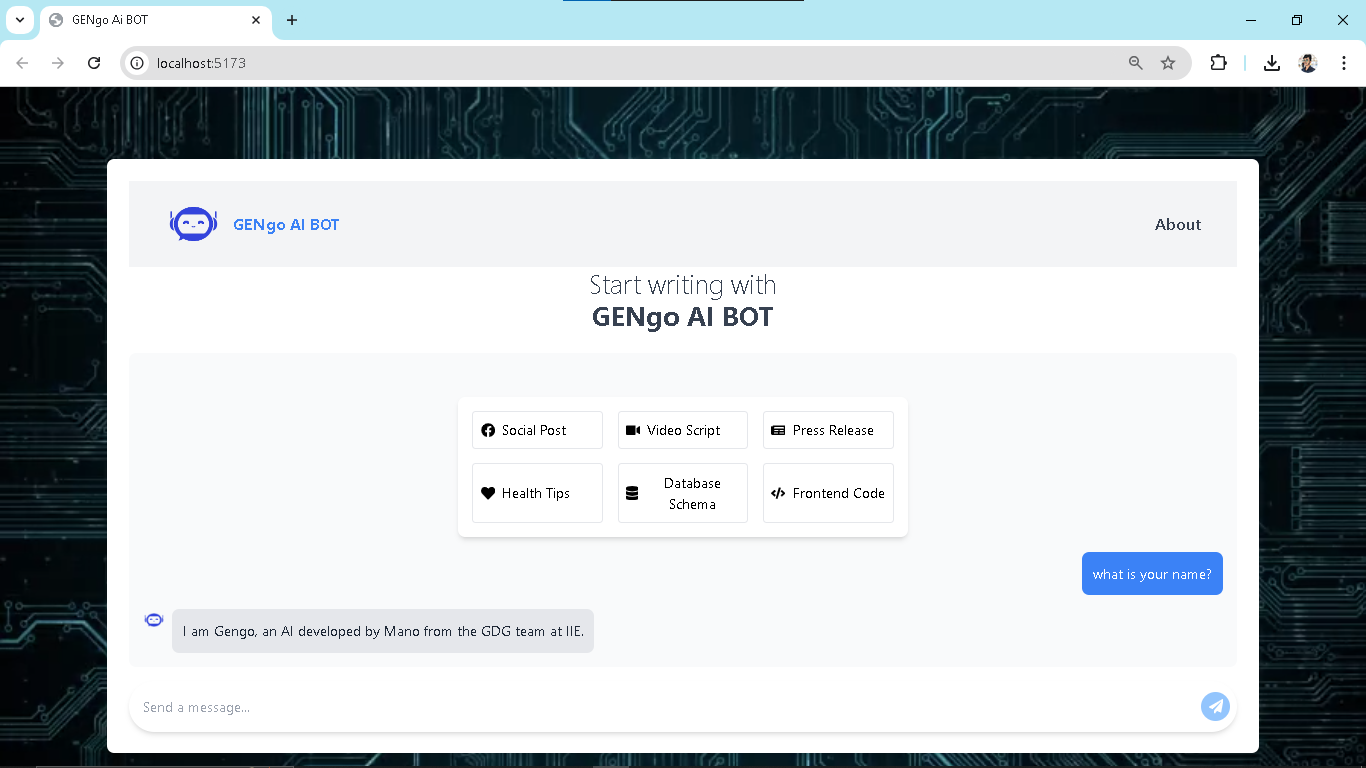
* **React Icons**: Used for icon representation within buttons and components.
* **Tailwind CSS**: For styling and responsive design.
* **React Router**: For managing navigation between components.

**Future Enhancements**

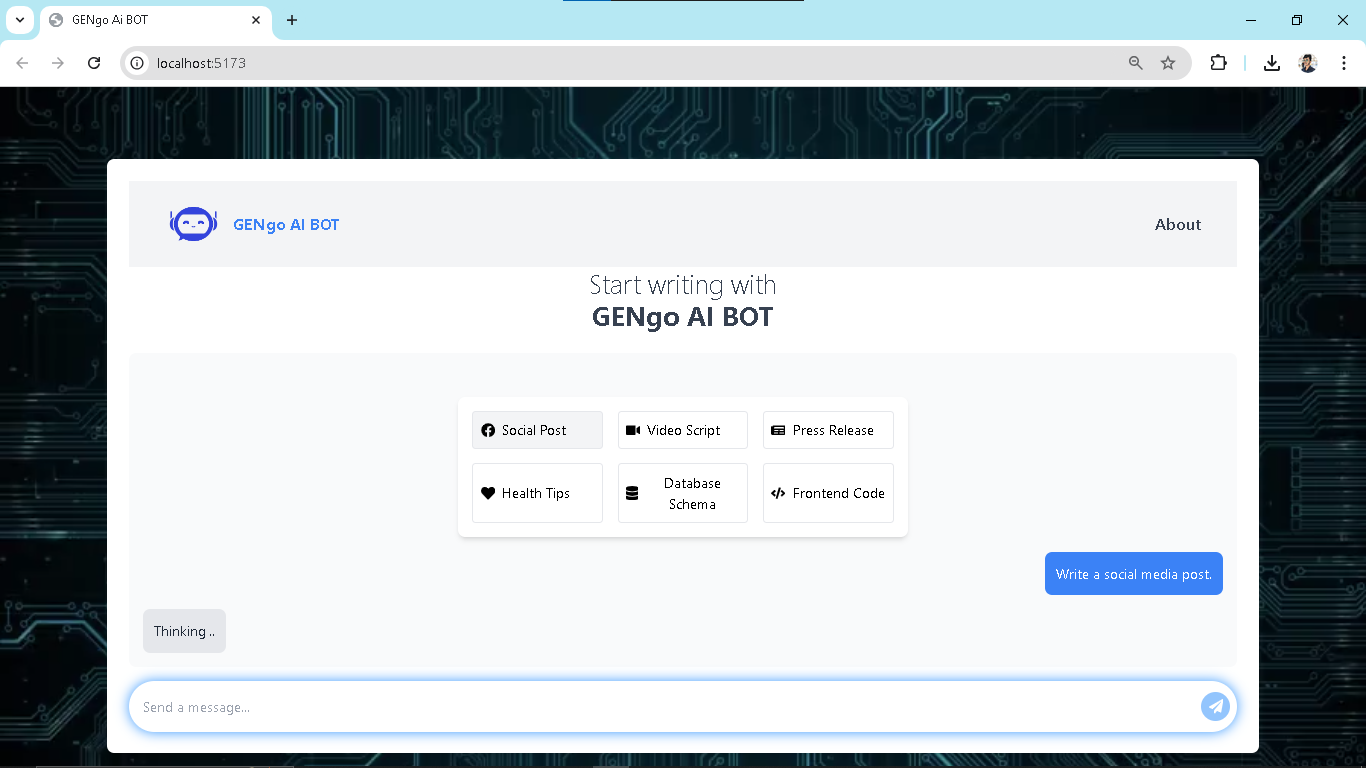
* Expand the bot's capabilities with more predefined actions.
* Improve error handling and user feedback.
* Implement user authentication for personalized experiences.

**OUTPUT**

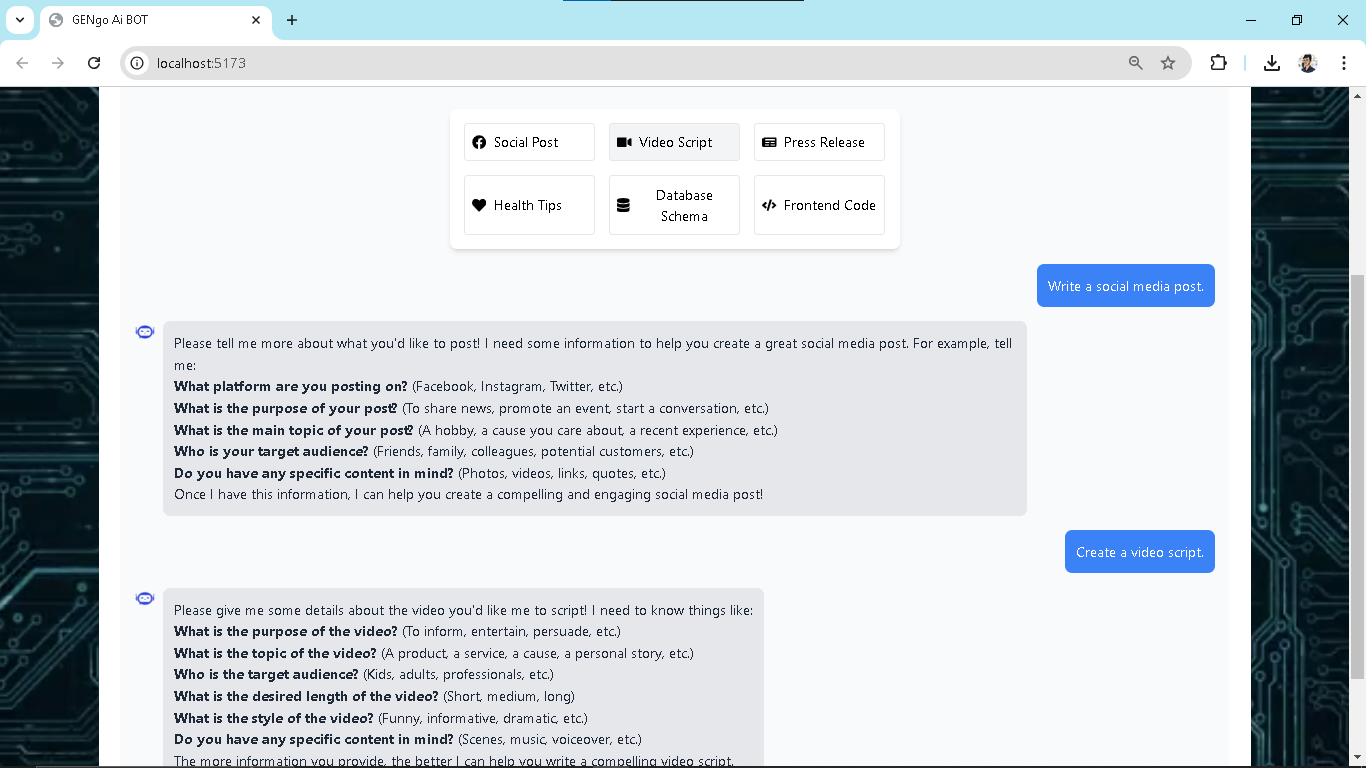
UI

****

**Animation**

****

**Button function**

****

**About us for app**

****

**Contributing**

Contributions are welcome! Please open an issue or submit a pull request for any improvements or features.

Thank you