How to run this chatbot?

- If node modules are missing - open terminal and enter
  npm i
- Npm start to run the chatbot {added dynamic port for depoying in AWS }
- Add openAi - key in .env file
- Add mongodb key in .env file

Deployed link (AWS)-
http://chatbot-1-env.eba-g5pyqtqr.ap-south-1.elasticbeanstalk.com/

Github Link -
https://github.com/SangamPrasad2000/gpt4-chatbot

App.Js

1. **Configuration Setup:**
   a. I have used the dotenv module to load environment variables from a .env file.
   b. Express, HTTP, Socket.IO, and the OpenAI API wrapper are imported.

CODE -

```
require("dotenv").config();
const express = require("express");
const http = require("http");
const socketIO = require("socket.io");
const { Configuration, OpenAIApi } = require("openai");
const { connectDB, ChatModel } = require("./mongoose");
const app = express();
const server = http.createServer(app);
const io = socketIO(server);
const port = process.env.PORT || 3000;
```

2. **OpenAI API Configuration:**
   a. The OpenAI API key and model are used to configured using the openai package.

CODE -

```
const configuration = new Configuration({
 apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);
```

3. **Static File Hosting and Database Connection:**
   a. Express middleware is used to serve static files from the "public" directory.
   b. The connectDB function is called to establish a connection to the MongoDB database.
   c.

CODE -

```
app.use(express.static("public"));
connectDB();
```

4. **API Endpoint to Retrieve Messages from MongoDB:**
   a. I have defined an API endpoint (/api/messages) to retrieve all messages from the MongoDB database.

CODE-

```
app.get("/api/messages", async (req, res) => {
 // ... (retrieve and send messages)
});
```

5. **Socket.IO Connection Handling:**
    a. The server listens for socket connections and logs when a new user connects or disconnects.

CODE-

```
io.on("connection", async (socket) => {
 // ... (handle user connections and disconnections)
});
```

6. **Handling User Messages:**
    a. The server listens for the "sendMessage" event from clients, processes the message using the OpenAI API, and sends back the assistant's response.

CODE -

```
socket.on("sendMessage", async (message, callback) => {
 // ... (handle user messages and send responses)
});
```

7. **Storing Messages in MongoDB:**
    a. User and assistant messages are stored in the MongoDB database using the Mongoose model ChatModel.

CODE -

```
const chat = new ChatModel({
 message: message,
 sender: "user",
 timestamp: new Date().toLocaleString(),
});

await chat.save();

const chat2 = new ChatModel({
 message: response,
 sender: "assistant",
 timestamp: new Date().toLocaleString(),
});

await chat2.save();
```

8. **Updating Conversation History:**
    a. The conversation history is updated with user and assistant messages.
    b.

CODE -

```
conversationHistory.push({ role: "user", content: message });
conversationHistory.push({ role: "assistant", content: response, timestamp: new
Date().toLocaleString() });
```

9. **Server Listening:**
    a. The server is set to listen on the specified port, and a message is logged when the server starts.

CODE -

```
server.listen(port, () => {
 console.log(`Server is running on port ${port}`);
});
```

--

Mongoose.js

1. **Database Connection:**
    a. The connectDB function uses Mongoose to connect to the MongoDB database. If the connection is successful, it logs a message indicating the successful connection. If there is an error, it logs the error message and exits the process.

2. **Chat Schema:**

   a. The chatSchema is defined using Mongoose's Schema class. It represents the structure of documents in the "Chat" collection, with fields for the message content, sender, and timestamp.
3. **Chat Model:**
   a. The ChatModel is created using Mongoose's model method. It binds the "Chat" model to the previously defined schema. This model is used to interact with the "Chat" collection in the MongoDB database.
4. **Exporting Modules:**
   - The connectDB and ChatModel are exported as part of an object to be used in other parts of the application.

Index.html

I have create the HTML file for the front-end part of my chat application. Html

1. **HTML Structure:**
   a. The HTML file follows the standard structure with <!DOCTYPE html> declaration.
   b. The <head> section includes meta tags for character set and viewport settings, a title, and a link to an external stylesheet (styles.css).
   c. The <body> section contains a <div> with the ID "chat-container" that wraps the chat interface.
2. **Chat Container:**
   a. Within the "chat-container" <div>, there is a child <div> with the ID "messages" to display the chat messages.
   b. An <input> element with the ID "message-input" is provided for users to type their messages, and a <button> element triggers the submitClick() function when clicked.
3. **Scripts:**
   a. The HTML file includes two <script> tags.

b. The first one includes the Socket.IO JavaScript library from the server (/socket.io/socket.io.js).
c. The second one includes a custom script (script.js), likely containing client-side JavaScript for handling chat interactions.
d.

Style.css

The provided CSS code is for styling the chat application's user interface. Here's an explanation of the key styles:

Body Styling:

- The body style sets the font family, background color, margin, padding, and uses flexbox to center the chat container both horizontally and vertically (justify-content: center; align-items: center;).
- height: 100vh; ensures that the body takes up the full height of the viewport.

Chat Container Styling:

- The #chat-container style defines the appearance of the main chat container. It has a white background, a box shadow for a subtle elevation effect, and rounded corners.
- overflow: hidden; ensures that any content exceeding the container's dimensions is hidden.
- The width is set to a maximum of 600 pixels but can scale down to 100% of the parent container's width (max-width: 100%;).
- display: flex; flex-direction: column; sets up a flex container with a column layout, and max-height: 700px; limits the height.

Messages Styling:

- The #messages style configures the appearance of the area where chat messages are displayed.
- It has padding, flex property set to 1 (allowing it to take up remaining space), and is set to scroll vertically when the content overflows.
- The height is set to 400 pixels (height: 400px;), and min-height: 400px; ensures it won't collapse to a smaller height.

Message Input Styling:

- The #message-input style defines the appearance of the input field for typing messages.
- It spans the full width, has padding, no border, and a light background color.
- border-top: 1px solid #ddd; adds a subtle separation between the input field and the messages.
- The input has a rounded bottom border (border-radius: 0 0 15px 15px;).

Button Styling:

- The button style configures the appearance of the send button.
- It has padding, a green background color (#4caf50;), white text, no border, and a cursor that indicates it's clickable.
- border-radius: 0 0 15px 15px; gives the button a rounded bottom border.
- On hover, the background color changes for a visual feedback effect.

User and Assistant Styles:

- The .user and .assistant styles define the text color for messages from the user and the assistant, respectively.
- The user's messages are in black (#000000), and the assistant's messages are in a shade of green (#2f6c32).

--

Script.js

WebSocket Connection

The application establishes a WebSocket connection using Socket.io. This connection enables real-time, bidirectional communication between the client (browser) and the server.

Event Listeners

Handling Incoming Messages

The application listens for a "message" event from the server. Upon receiving a message, it updates the chat interface by removing any existing typing indicator and displaying the received message, attributed to the assistant.

Handling Typing Indicator

The application listens for a "typing" event from the server. When this event occurs, it displays a visual indicator in the chat, informing the user that the assistant is currently typing.

User Input Handling

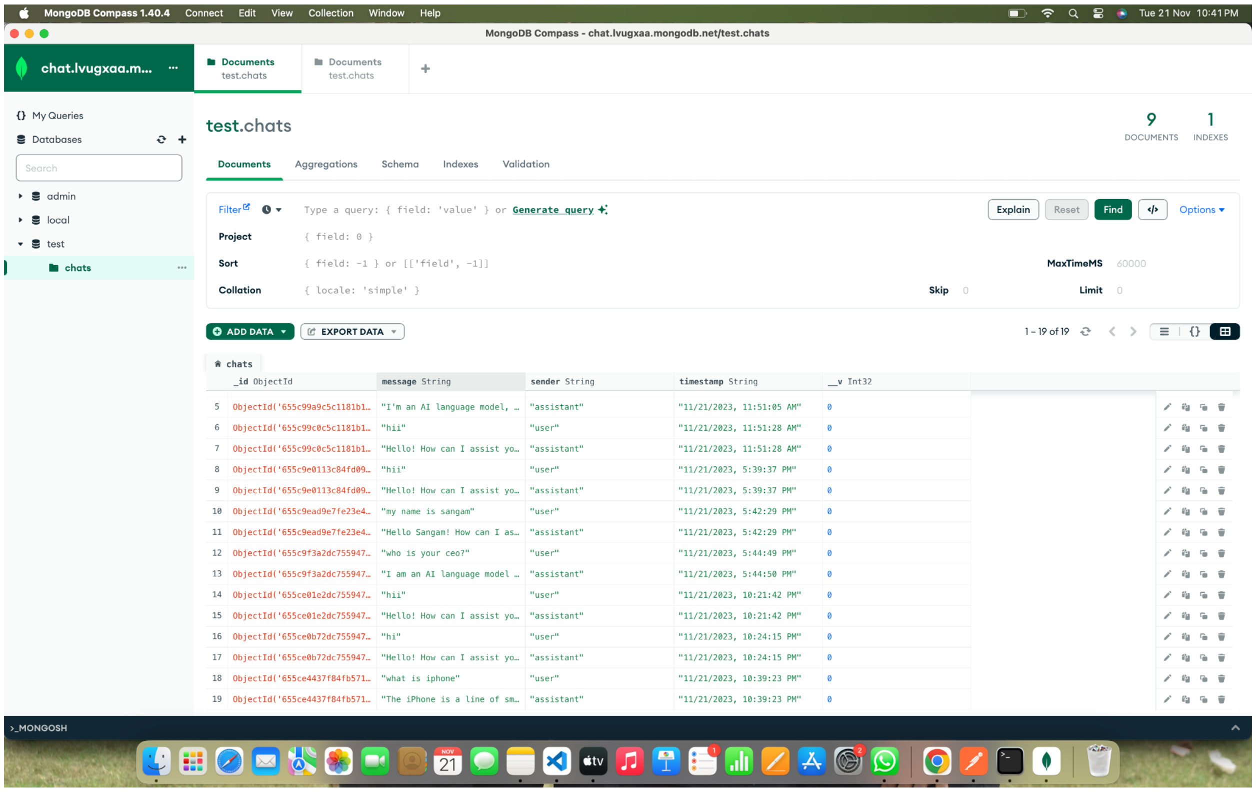
The application captures user input in the message input field. An event listener is set up to detect when the Enter key is pressed. If the Enter key is detected, it triggers the submission of the user's message.

Fetching Initial Messages

Upon loading the page, the application fetches initial messages from the server. This ensures that the chat interface displays any existing conversation history, attributing each message to its respective sender (user or assistant).

Test Images

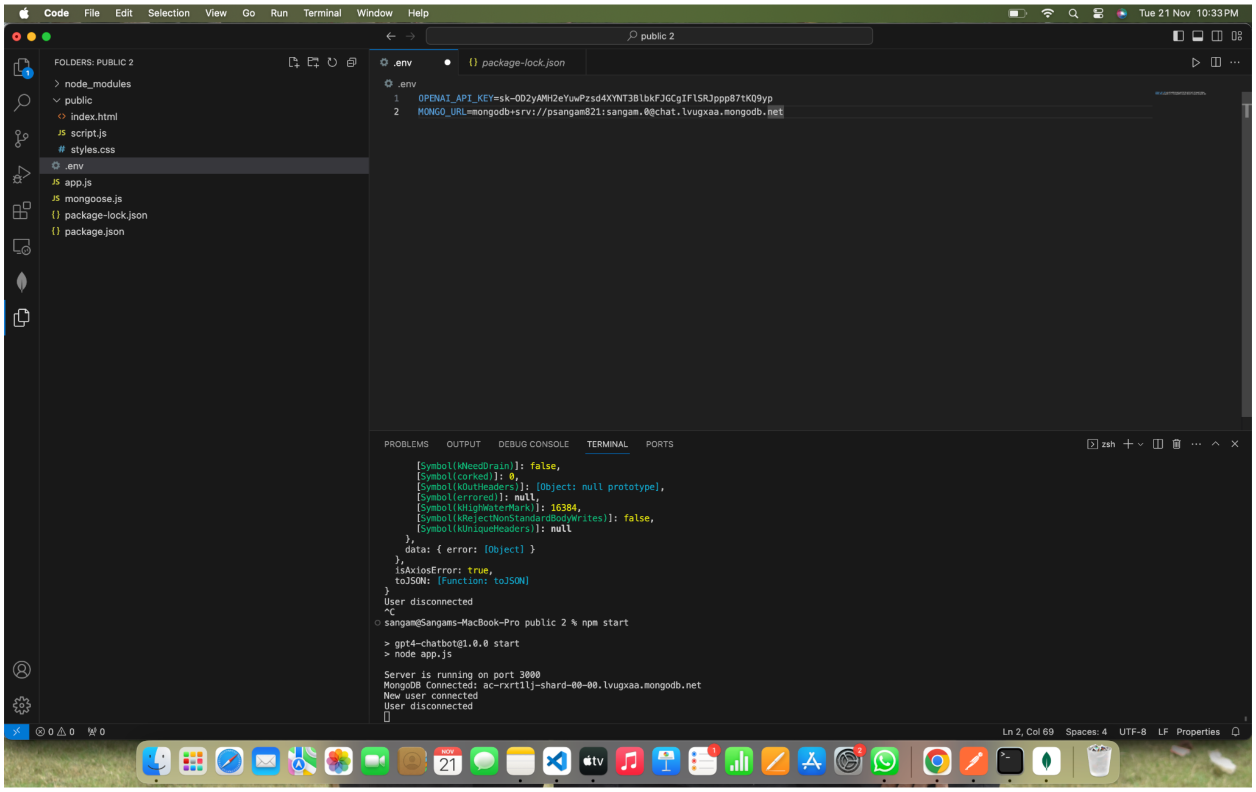
Stored Chats in Mongo db



chats

	_id ObjectId	message String	sender String	timestamp String	__v Int32
5	ObjectId('655c99a9c5c1181b1…	"I'm an AI language model, …	"assistant"	"11/21/2023, 11:51:05 AM"	0
6	ObjectId('655c99c0c5c1181b1…	"hii"	"user"	"11/21/2023, 11:51:28 AM"	0
7	ObjectId('655c99c0c5c1181b1…	"Hello! How can I assist yo…	"assistant"	"11/21/2023, 11:51:28 AM"	0
8	ObjectId('655c9e0113c84fd09…	"hii"	"user"	"11/21/2023, 5:39:37 PM"	0
9	ObjectId('655c9e0113c84fd09…	"Hello! How can I assist yo…	"assistant"	"11/21/2023, 5:39:37 PM"	0
10	ObjectId('655c9ead9e7fe23e4…	"my name is sangam"	"user"	"11/21/2023, 5:42:29 PM"	0
11	ObjectId('655c9ead9e7fe23e4…	"Hello Sangam! How can I as…	"assistant"	"11/21/2023, 5:42:29 PM"	0
12	ObjectId('655c9f3a2dc755947…	"who is your ceo?"	"user"	"11/21/2023, 5:44:49 PM"	0
13	ObjectId('655c9f3a2dc755947…	"I am an AI language model …	"assistant"	"11/21/2023, 5:44:50 PM"	0
14	ObjectId('655ce01e2dc755947…	"hii"	"user"	"11/21/2023, 10:21:42 PM"	0
15	ObjectId('655ce01e2dc755947…	"Hello! How can I assist yo…	"assistant"	"11/21/2023, 10:21:42 PM"	0
16	ObjectId('655ce0b72dc755947…	"hi"	"user"	"11/21/2023, 10:24:15 PM"	0
17	ObjectId('655ce0b72dc755947…	"Hello! How can I assist yo…	"assistant"	"11/21/2023, 10:24:15 PM"	0
18	ObjectId('655ce4437f84fb571…	"what is iphone"	"user"	"11/21/2023, 10:39:23 PM"	0
19	ObjectId('655ce4437f84fb571…	"The iPhone is a line of sm…	"assistant"	"11/21/2023, 10:39:23 PM"	0

.env file



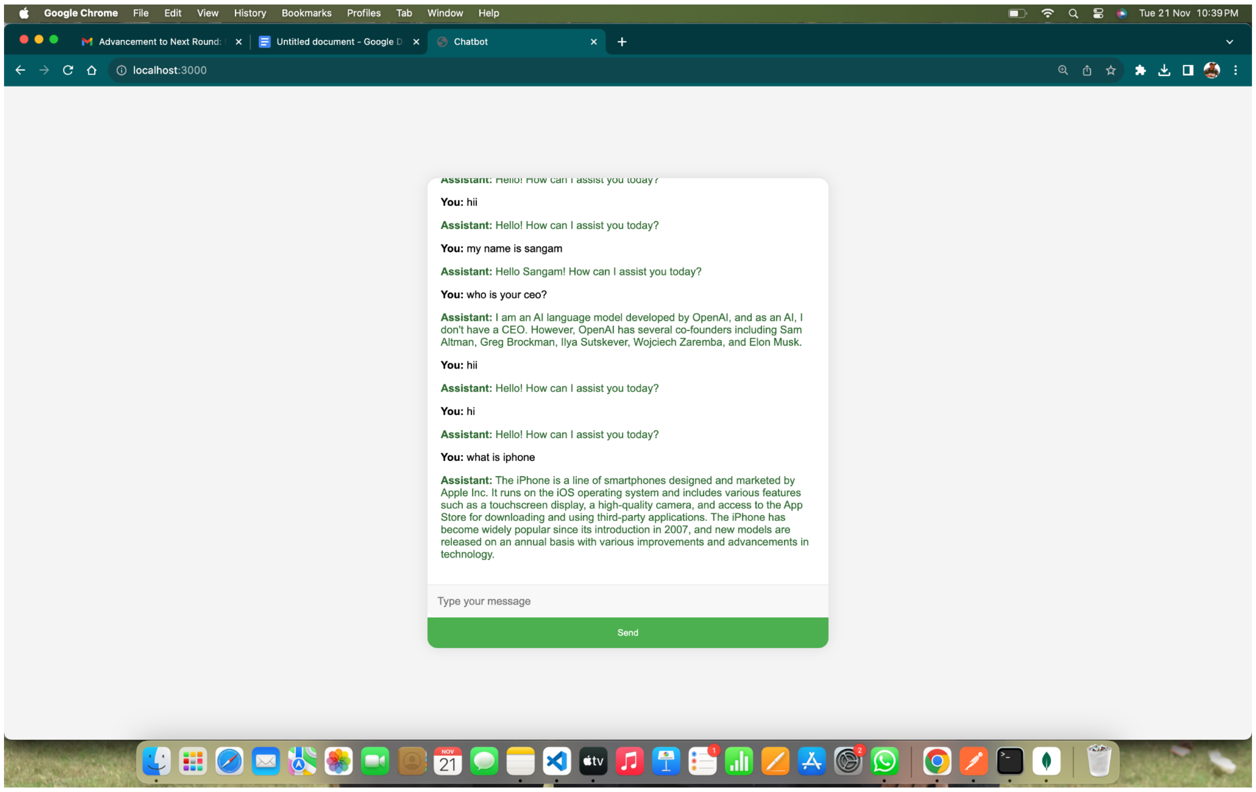
```
OPENAI_API_KEY=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
MONGO_URL=mongodb+srv://psangam821:sangam.0@chat.lvugxaa.mongodb.net
```

```
      [Symbol(kNeedDrain)]: false,
      [Symbol(corked)]: 0,
      [Symbol(kOutHeaders)]: [Object: null prototype],
      [Symbol(errored)]: null,
      [Symbol(kHighWaterMark)]: 16384,
      [Symbol(kRejectNonStandardBodyWrites)]: false,
      [Symbol(kUniqueHeaders)]: null
    },
    data: { error: [Object] }
  },
  isAxiosError: true,
  toJSON: [Function: toJSON]
}
User disconnected
^C
sangam@Sangams-MacBook-Pro public 2 % npm start

> gpt4-chatbot@1.0.0 start
> node app.js

Server is running on port 3000
MongoDB Connected: ac-rxrt1lj-shard-00-00.lvugxaa.mongodb.net
New user connected
User disconnected
```

Application test -



Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue 21 Nov 10:39 PM
Advancement to Next Round
Untitled document - Google D
Chatbot
localhost:3000

Assistant: Hello! How can I assist you today?

You: hii

Assistant: Hello! How can I assist you today?

You: my name is sangam

Assistant: Hello Sangam! How can I assist you today?

You: who is your ceo?

Assistant: I am an AI language model developed by OpenAI, and as an AI, I don't have a CEO. However, OpenAI has several co-founders including Sam Altman, Greg Brockman, Ilya Sutskever, Wojciech Zaremba, and Elon Musk.

You: hii

Assistant: Hello! How can I assist you today?

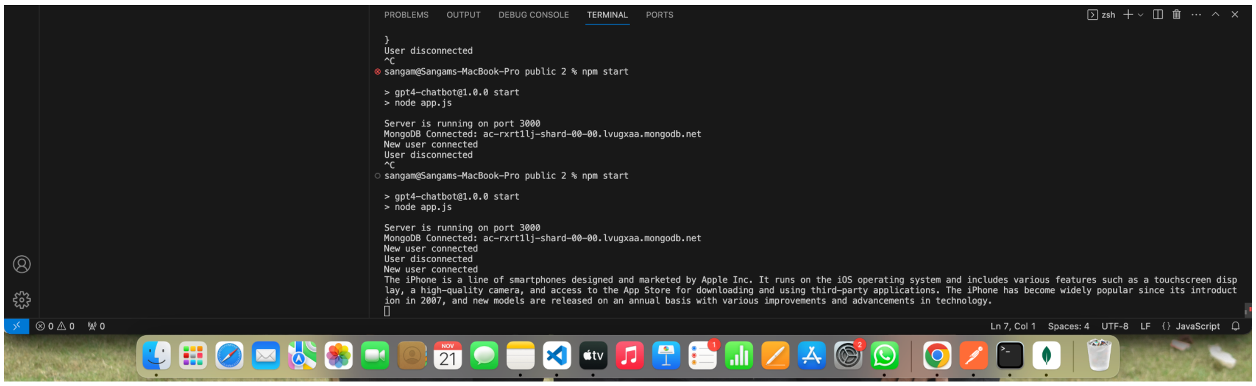
You: hi

Assistant: Hello! How can I assist you today?

You: what is iphone

Assistant: The iPhone is a line of smartphones designed and marketed by Apple Inc. It runs on the iOS operating system and includes various features such as a touchscreen display, a high-quality camera, and access to the App Store for downloading and using third-party applications. The iPhone has become widely popular since its introduction in 2007, and new models are released on an annual basis with various improvements and advancements in technology.

Response validation -



```
}
User disconnected
^C
sangam@Sangams-MacBook-Pro public 2 % npm start

> gpt4-chatbot@1.0.0 start
> node app.js

Server is running on port 3000
MongoDB Connected: ac-rxrt1lj-shard-00-00.lvugxaa.mongodb.net
New user connected
User disconnected
^C
sangam@Sangams-MacBook-Pro public 2 % npm start

> gpt4-chatbot@1.0.0 start
> node app.js

Server is running on port 3000
MongoDB Connected: ac-rxrt1lj-shard-00-00.lvugxaa.mongodb.net
New user connected
User disconnected
New user connected
The iPhone is a line of smartphones designed and marketed by Apple Inc. It runs on the iOS operating system and includes various features such as a touchscreen disp
lay, a high-quality camera, and access to the App Store for downloading and using third-party applications. The iPhone has become widely popular since its introduct
ion in 2007, and new models are released on an annual basis with various improvements and advancements in technology.
```

Potential Improvements :-

1. **We can add user login and sighnup for privacy in search.**
2. **We can add word by word result and removing bite by bite result for better and faster response.**
3. **Display timestamps for each message to indicate when a message was sent.**
4. **Enable basic message formatting options such as bold, italics, and links. This can be especially useful for users who want to emphasize certain parts of their messages.**
5. **Allow users to customize the behavior or appearance of the chatbot. This could include selecting a chatbot persona or adjusting its language style.**

ChatBot By Sangam Prasad