ChatGPT API APP

It is an OpenAI API app created with the help of React and NodeJS which enables developers to make HTTP requests to the API endpoint, send prompt/instructions in a way they receive required responses/output which is model generated responses programmatically.

How to create ChatGPT API App-

First install the following things in your system-

* Any IDE or Code Editor (like: VS Code, PyCharm, Jupiter, Eclipse)
* React
* Nodejs

> Now open the IDE and add a new folder to workspace.

>Then create the react app using following command:

* npx create-react-app app\_name

>Now install the required dependencies using the commands below:

* npm install express
* npm install cors
* npm install body-parser
* npm install openai
* npm install axios

>Now after installing all required dependencies, it will give access to all we need to create the app.

>Now creating the Nodejs server for the OpenAi API

* Import all the required dependencies -

|  |
| --- |
| const express = require("express");  const cors = require("cors");  const bodyParser = require("body-parser");  const { Configuration, OpenAIApi } = require("openai"); |

* Now, create config and it creates a new instance of the Configuration object. Inside it, you enter values for your own apiKey. As it will give us access to the chat completion functionalities.

|  |
| --- |
| const config = new Configuration({  apiKey: "Your own API key", });  const openai = new OpenAIApi(config); |

Next, set up express, the body-parser for receiving input, and cors to allow free communications between the frontend and backend.

|  |
| --- |
| const app = express(); app.use(bodyParser.json()); app.use(cors()); |

*// endpoint for ChatGPT*

Now, I created an endpoint that will receive messages from the frontend using the request body and return a response to the caller.

|  |
| --- |
| app.post("/chat", async (req, res) => {  const {prompt} = req.body;   const completion = await openai.createCompletion({  model: "text-davinci-003",  max\_tokens: 512,  temperature: 0,  prompt: prompt,  });  res.send(completion.data.choices[0].text); }); |

This endpoint will be called using <http://localhost:8080/>

|  |
| --- |
| const PORT = 8080; app.listen(PORT, () => {  console.log(`Server is running on port: ${PORT}`); }); |

*//run node server.js*

Full Nodejs Code-

|  |
| --- |
| const express = require("express");  const cors = require("cors");  const bodyParser = require("body-parser");  const {Configuration, OpenAIApi } = require("openai");  const config = new Configuration({  apiKey: "YOUR\_API\_KEY",  });  const openai = new OpenAIApi(config);  // Setup server  const app = express();  app.use(bodyParser.json());  app.use(cors());    // endpoint for ChatGPT    app.post("/chat", async (req, res) => {  const { prompt } = req.body;  const completion = await openai.createCompletion({  model: "text-davinci-003",  max\_tokens: 512,  temperature: 0,  prompt: prompt,  });  res.send(completion.data.choices[0].text);  });  const PORT = 8080;  app.listen(PORT, () => {  console.log (`Server is running on port: ${PORT}`);  });    //run node server.js |

>Create the ChatGPT API App Frontend Part-

First, we will import the required statements like states, axios.

|  |
| --- |
| import React, { useState } from "react";  import axios from "axios"; |

Now create a default export function and setup all the states before the return statement:

|  |
| --- |
| export default function ChatGPT() {  const [prompt, setPrompt] = useState("");  const [response, setResponse] = useState("");  const HTTP = <http://localhost:8080/chat>;  return (  <div>    </div>  );  }  export default ChatGPT; |

* The prompt will hold the information sent from the app to the AI.
* The response will send the required output request by the user.
* Endpoint calls have been stored in HTTP variable.

Now add H1 tag in the return statement inside the div container. You can add any heading as per your wish -

|  |
| --- |
| <h1 className="title text-center text-darkGreen">ChatGPT API</h1>  or  <h1> Add Any Heading as you want to give </h1> |

Now I have created a form in which a user can type in a message

|  |
| --- |
| <form className="form" onSubmit={handleSubmit}>  <div className="form-group">  <label htmlFor="">Ask questions</label>  <input className="shadow-sm" type="text"  placeholder="Enter text"  value={prompt}  onChange={handlePrompt}  />  </div>  </form> |

Now when the form is submitted, we call the handleSubmit () function, which prevents the page from reloading and a post request will be sent using axios.post to the endpoint. This takes an object to send the data in and also adds the data to the state and it also handles errors using .then() and .catch() method.

|  |
| --- |
| const handleSubmit = (e) => {  e.preventDefault();  axios.post(`${HTTP}`, { prompt })  .then((res) => {  setResponse(res.data);  console.log(prompt);  })  .catch((error) => {  console.log(error);  });  setPrompt("");  }; |

Now I have set the value of input equal to prompt so that whatever user types in the input will be shown there.

|  |
| --- |
| const handlePrompt = (e) => { setPrompt(e.target.value);}; |

Now I have added the paragraph (p Tag) inside a div where user will receive the responses.

|  |
| --- |
| <div className="bg-darkGreen mt-2 p-1 border-5">  <p className="text-light"> {response ? response : "Ask me anything..."} </p>  </div> |

Here is some JavaScript logic I have written that is if user asks a prompt, they will get required response otherwise if input is empty, it will show “Ask me anything...”

i.e.

|  |
| --- |
| {response ? response : "Ask me anything..."} |

ChatGPT API App Frontend Full Code-

import React, { useState } from "react";

import axios from "axios";

export default function ChatGPT() {

const [prompt, setPrompt] = useState("");

const [response, setResponse] = useState("");

const HTTP = "http://localhost:8080/chat";

const handleSubmit = (e) => {

e.preventDefault();

axios.post(`${HTTP}`, { prompt })

.then((res) => {

setResponse(res.data);

console.log(prompt);

})

.catch((error) => {

console.log(error);

});

};

const handlePrompt = (e) => {

setPrompt(e.target.value);

};

return (

<div className="container container-sm p-1">

<h1 className="title text-center text-darkGreen">ChatGPT API</h1>

<form className="form" onSubmit={handleSubmit}>

<div className="form-group">

<label htmlFor="">Ask questions</label>

<input

className="shadow-sm"

type="text"

placeholder="Enter text"

value={prompt}

onChange={handlePrompt}

/>

</div>

</form>

<div className="bg-darkGreen mt-2 p-1 border-5">

<p className="text-light">{response ? response : "Ask me anything..."}</p>

</div>

</div>

);

}

>ChatGPT API App CSS Part-

.container {

max-width: 900px;

margin: 0 auto;

}

.container.container-sm {

max-width: 782px;

}

.p-1 {

padding: 1rem;

}

.mt-2 {

margin-top: 2rem;

}

.border-5 {

border-radius: 5px;

}

.title {

font-size: 3.75rem;

font-weight: 500;

margin: 3rem 0;

}

.form-group {

margin-bottom: 1rem;

}

.form-group label {

display: block;

font-weight: 600;

margin-bottom: 0.5rem;

}

.form-group input,

.form-group select {

border: 1px solid rgba(238, 238, 238, 0.9333333333);

border-radius: 5px;

padding: 0.5rem;

.text-center {

text-align: center;

}

.shadow-sm {

box-shadow: 0 2px 8px 3px rgba(0, 0, 0, 0.15);

}

.bg-darkGreen {

background-color: rgb(36, 59, 36);

}

.text-darkGreen {

color: rgb(36, 59, 36);

}

.text-light {

color: #f8f9fa;

}

Remove all the pre applied CSS from the src/App.css file and then apply your own CSS.

**Output 1:**

A screenshot of a chat

Description automatically generated

**Output 2:**

A screenshot of a computer program

Description automatically generated