

create a chat gpt like ai with options of create image, help me write, get advice,summarize text, code, brainstorm,surprise me, make a plan, and all

ChatGPT said:

ChatGPT

Creating a ChatGPT-like AI application that includes the options to generate images, help write content, provide advice, summarize text, generate code, brainstorm ideas, surprise users, make a plan, and more involves several steps. This can be accomplished using technologies like React for the frontend and a Python-based backend (using Flask or Django), along with OpenAI's API to power the AI and additional APIs for image generation.

1. Technology Stack:

Frontend: React (for the user interface)

Backend: Python (Flask/Django)

AI Integration: OpenAI API (for GPT-3/4)

Image Generation: DALL-E API (for image generation)

Text Summarization: OpenAI API (for summarizing text)

Deployment: Heroku or any other cloud service (for hosting both frontend and backend)

2. Features:

Your application will have the following features:

Create Image: Generate images based on user prompts using DALL-E or similar models.

Help Me Write: Assist in writing content (articles, blogs, etc.) using GPT-3/4.

Get Advice: Provide general advice based on user queries.

Summarize Text: Summarize long pieces of text into concise summaries.

Code: Help generate code for various programming tasks.

Brainstorm: Generate creative ideas or solutions for given problems.

Surprise Me: Generate a random fun fact, quote, joke, etc.

Make a Plan: Create a step-by-step plan based on user needs (e.g., meal plans, study plans).

All: Aggregate all available features into one accessible interface.

3. Setup & Code Implementation:

Step 1: Setup Frontend with React

Let's create the frontend to give users the option to choose various features like "Create Image", "Help Me Write", etc.

```
npx create-react-app ai-chat-app
```

```
cd ai-chat-app
```

```
npm install axios
```

In src/App.js, let's set up the basic structure and frontend components.

```
import React, { useState } from 'react';
```

```
import axios from 'axios';
```

```
const App = () => {
```

```
  const [input, setInput] = useState("");
```

```
  const [output, setOutput] = useState("");
```

```
  const [category, setCategory] = useState('helpWrite');
```

```
  const handleInputChange = (event) => setInput(event.target.value);
```

```
  const handleSubmit = async () => {
```

```
    if (input.trim()) {
```

```
      try {
```

```
        const response = await axios.post('http://localhost:5000/api/ai', {
```

```
category,  
input,  
});  
setOutput(response.data.output);  
} catch (error) {  
setOutput('Error: Unable to get a response');  
}  
}  
};
```

```
return (  
<div>  
<h1>AI Chatbot</h1>  
<select onChange={(e) => setCategory(e.target.value)} value={category}>  
<option value="helpWrite">Help Me Write</option>  
<option value="createImage">Create Image</option>  
<option value="getAdvice">Get Advice</option>  
<option value="summarizeText">Summarize Text</option>  
<option value="code">Code</option>  
<option value="brainstorm">Brainstorm</option>  
<option value="surpriseMe">Surprise Me</option>  
<option value="makePlan">Make a Plan</option>  
</select>  
<textarea  
placeholder="Type something..."  
value={input}  
onChange={handleInputChange}  
>  
<button onClick={handleSubmit}>Submit</button>
```

```
<div>
<h3>Output:</h3>
<p>{output}</p>
</div>
</div>
);
};
```

```
export default App;
```

Step 2: Set Up the Backend in Python

Now, we need to set up the backend using Flask (or Django). Flask is simpler for a small application like this.

Install the necessary dependencies:

```
pip install Flask openai requests
```

Create app.py:

```
from flask import Flask, request, jsonify
import openai
import requests
```

```
app = Flask(__name__)
```

```
# Initialize OpenAI API key (set your API key here)
openai.api_key = "your-openai-api-key"
```

```
@app.route('/api/ai', methods=['POST'])
def ai_response():
    data = request.get_json()
    category = data.get('category')
    user_input = data.get('input')

    if category == 'helpWrite':
        response = openai.Completion.create(
            engine="text-davinci-003",
            prompt=f"Write a detailed and engaging article on: {user_input}",
            max_tokens=300
        )
        output = response.choices[0].text.strip()

    elif category == 'createImage':
        # Use DALL-E to create an image (this is a placeholder)
        response = openai.Image.create(
            prompt=user_input,
            n=1,
            size="1024x1024"
        )
        output = response['data'][0]['url']

    elif category == 'getAdvice':
        response = openai.Completion.create(
            engine="text-davinci-003",
            prompt=f"Provide advice on: {user_input}",
            max_tokens=150
        )
```

```
output = response.choices[0].text.strip()
```

```
elif category == 'summarizeText':  
    response = openai.Completion.create(  
        engine="text-davinci-003",  
        prompt=f"Summarize the following text:\n{user_input}",  
        max_tokens=150  
    )  
    output = response.choices[0].text.strip()
```

```
elif category == 'code':  
    response = openai.Completion.create(  
        engine="code-davinci-002",  
        prompt=f"Write a code to solve the following problem:\n{user_input}",  
        max_tokens=150  
    )  
    output = response.choices[0].text.strip()
```

```
elif category == 'brainstorm':  
    response = openai.Completion.create(  
        engine="text-davinci-003",  
        prompt=f"Brainstorm ideas for: {user_input}",  
        max_tokens=150  
    )  
    output = response.choices[0].text.strip()
```

```
elif category == 'surpriseMe':  
    response = openai.Completion.create(  
        engine="text-davinci-003",
```

```
prompt="Give me a random fact, joke, or quote!",
max_tokens=100
)
output = response.choices[0].text.strip()

elif category == 'makePlan':
response = openai.Completion.create(
engine="text-davinci-003",
prompt=f"Create a plan for: {user_input}",
max_tokens=200
)
output = response.choices[0].text.strip()

return jsonify({'output': output})
```

```
if __name__ == '__main__':
app.run(debug=True)
```

Step 3: Running the App

Start the Flask server:

```
python app.py
```

Start the React app:

```
npm start
```

4. Deployment:

Once everything is ready, you can deploy the backend and frontend. You can deploy the backend using Heroku or AWS Lambda for easy deployment, and the frontend can be hosted on services like Netlify, Vercel, or Heroku.

5. Summary of Features:

Create Image: The AI generates images based on the prompt using the DALL-E API.

Help Me Write: Generates text (articles, essays, etc.) based on the user's input using OpenAI's GPT model.

Get Advice: Provides advice based on the user's request.

Summarize Text: Summarizes any long text to make it shorter and concise.

Code: Helps generate code for specific programming problems.

Brainstorm: Generates creative ideas and solutions for various topics.

Surprise Me: Random facts, jokes, or inspirational quotes.

Make a Plan: The AI creates a structured plan for a specific goal.

6. Final Thoughts:

This setup provides a foundation for building a powerful ChatGPT-like AI assistant with various functionalities. You can further enhance it by adding more advanced features like multi-turn conversations, personalization, and better UI/UX design.