Code with Documentation

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

class ChatGPTBotAPI:
    def __init__(self, api_key, engine='text-davinci-002', temperature=0.7, max_tokens=150):
        """
        Initialize the ChatGPT Bot API.

        Parameters:
            api_key (str): Your OpenAI API key.
            engine (str): The GPT-3.5 engine to use for language generation.
                          Default: 'text-davinci-002'.
            temperature (float): Controls the randomness of the generated output.
                                 Values closer to 0 make the output more deterministic,
                                 while values closer to 1 make it more random.
                                 Default: 0.7.
            max_tokens (int): The maximum number of tokens to generate for a response.
                              Default: 150.
        """
        self.api_key = api_key
        self.engine = engine
        self.temperature = temperature
        self.max_tokens = max_tokens
        self.prompts = []

    def initialize_gpt3(self):
        """
        Initialize the OpenAI API with the provided API key.
        """
        openai.api_key = self.api_key

    def create_prompt(self, prompt):
        """
        Create a new prompt and store it for later interactions with the ChatGPT bot.

        Parameters:
            prompt (str): The text of the prompt to be created.
```

```
    Returns:
        int: The index of the newly created prompt in the prompts list.
    """
    self.prompts.append(prompt)
    return len(self.prompts) - 1

def get_response(self, prompt_index):
    """
    Get the ChatGPT bot's response to a previously stored prompt.

    Parameters:
        prompt_index (int): The index of the prompt to use for generating the response.

    Returns:
        str: The generated response from the GPT-3.5 model.
    """
    if prompt_index >= len(self.prompts) or prompt_index < 0:
        return "Invalid prompt index."

    prompt = self.prompts[prompt_index]
    response = openai.Completion.create(
        engine=self.engine,
        prompt=prompt,
        temperature=self.temperature,
        max_tokens=self.max_tokens
    )
    return response['choices'][0]['text']

def update_prompt(self, prompt_index, new_prompt):
    """
    Update an existing prompt at the given index with a new prompt provided by the user.

    Parameters:
        prompt_index (int): The index of the prompt to update.
        new_prompt (str): The text of the new prompt to replace the existing one.

    Returns:
        str: A message indicating whether the prompt was updated successfully.
    """
    if prompt_index >= len(self.prompts) or prompt_index < 0:
        return "Invalid prompt index."

    self.prompts[prompt_index] = new_prompt
```

```
        return "Prompt updated successfully."

    def delete_prompt(self, prompt_index):
        """
        Delete a prompt from the list of stored prompts at the given index.

        Parameters:
            prompt_index (int): The index of the prompt to delete.

        Returns:
            str: A message indicating whether the prompt was deleted successfully.
        """
        if prompt_index >= len(self.prompts) or prompt_index < 0:
            return "Invalid prompt index."

        del self.prompts[prompt_index]
        return "Prompt deleted successfully."


# Create Flask app and ChatGPT Bot instance
app = Flask(__name__)
chatbot =
ChatGPTBotAPI(api_key='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')

# Initialize the ChatGPT Bot with OpenAI API
chatbot.initialize_gpt3()

@app.route('/create_prompt', methods=['POST'])
def create_prompt():
    """
    API endpoint to create a new prompt.

    Request Body:
        {
          "prompt": "This is a test prompt."
        }

    Response:
        {
          "message": "Prompt created successfully.",
          "prompt_index": 0
        }
    """
    data = request.get_json()
```

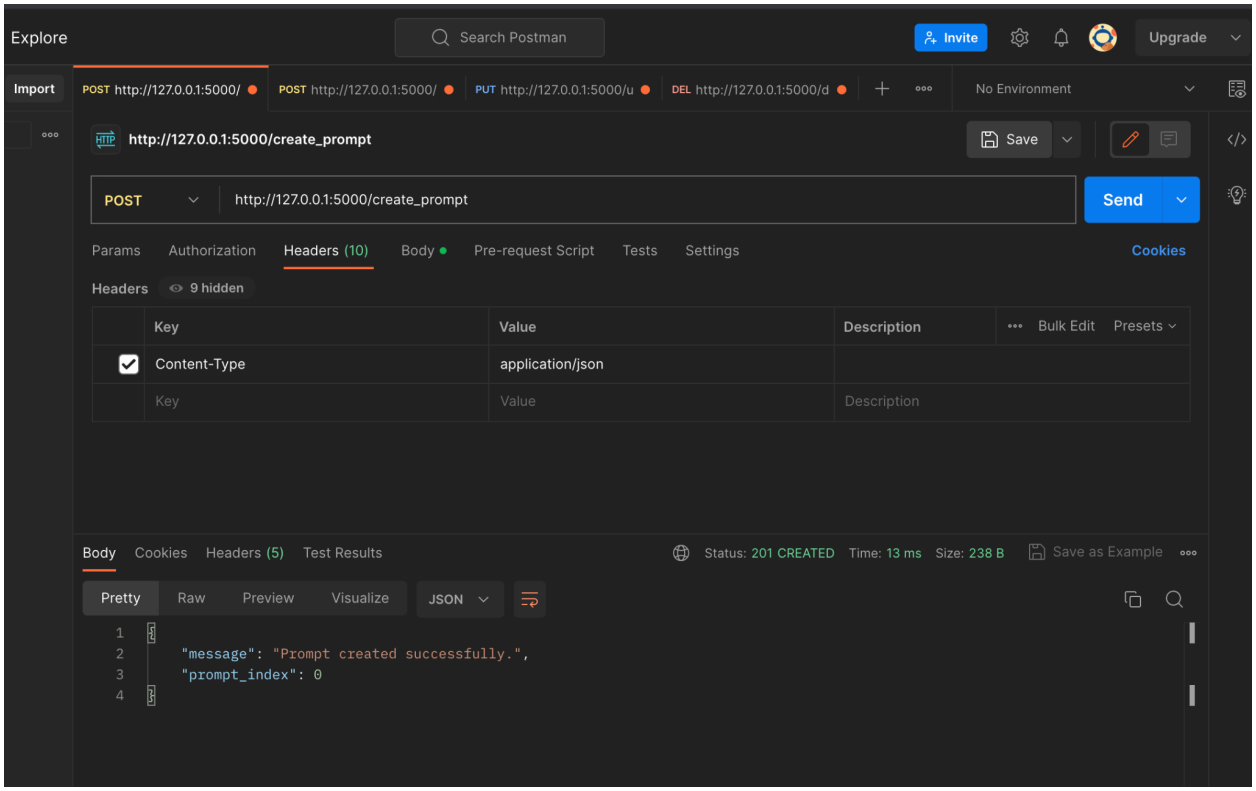
```
    prompt = data.get('prompt')
    if not prompt:
        return jsonify({"message": "Prompt is missing."}), 400

    prompt_index = chatbot.create_prompt(prompt)
    return jsonify({"message": "Prompt created successfully.", "prompt_index": prompt_index}),
201

@app.route('/get_response', methods=['POST'])
def get_response():
    """
    API endpoint to get the ChatGPT bot's response to a previously stored prompt.

    Request Body:
        {
          "prompt_index": 0
        }

    Response:
        {
          "response": "Generated response from the GPT-3.5 model."
        }
    """
    data = request.get_json()
    prompt_index = data.get('prompt_index')
    if prompt_index is None:
        return jsonify({"message": "Prompt index is missing."}), 400

    response = chatbot.get_response(prompt_index)
    return jsonify({"response": response}), 200

@app.route('/update_prompt', methods=['PUT'])
def update_prompt():
    """
    API endpoint to update an existing prompt.

    Request Body:
        {
          "prompt_index": 0,
          "new_prompt": "This is an updated prompt."
        }

    Response:
        {
```

```
        "message": "Prompt updated successfully."
      }
    """
    data = request.get_json()
    prompt_index = data.get('prompt_index')
    new_prompt = data.get('new_prompt')

    if prompt_index is None or new_prompt is None:
        return jsonify({"message": "Prompt index or new prompt is missing."}), 400

    result = chatbot.update_prompt(prompt_index, new_prompt)
    return jsonify({"message": result}), 200

@app.route('/delete_prompt', methods=['DELETE'])
def delete_prompt():
    """
    API endpoint to delete a prompt.

    Request Body:
        {
          "prompt_index": 0
        }

    Response:
        {
          "message": "Prompt deleted successfully."
        }
    """
    data = request.get_json()
    prompt_index = data.get('prompt_index')

    if prompt_index is None:
        return jsonify({"message": "Prompt index is missing."}), 400

    result = chatbot.delete_prompt(prompt_index)
    return jsonify({"message": result}), 200

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

CURL Commands to test CRUD Operations

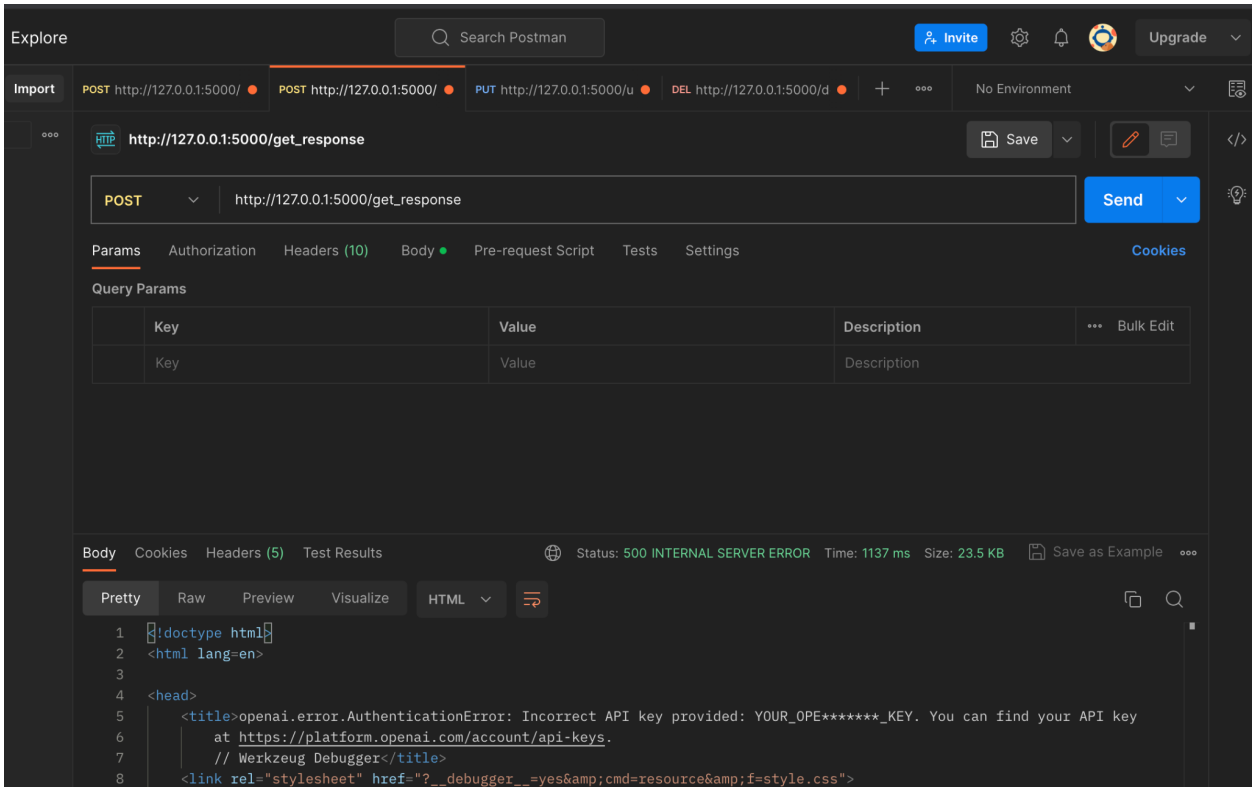
Create:

curl -X POST -H "Content-Type: application/json" -d '{"prompt": "This is a test prompt."}' http://127.0.0.1:5000/create_prompt



Retrieve:

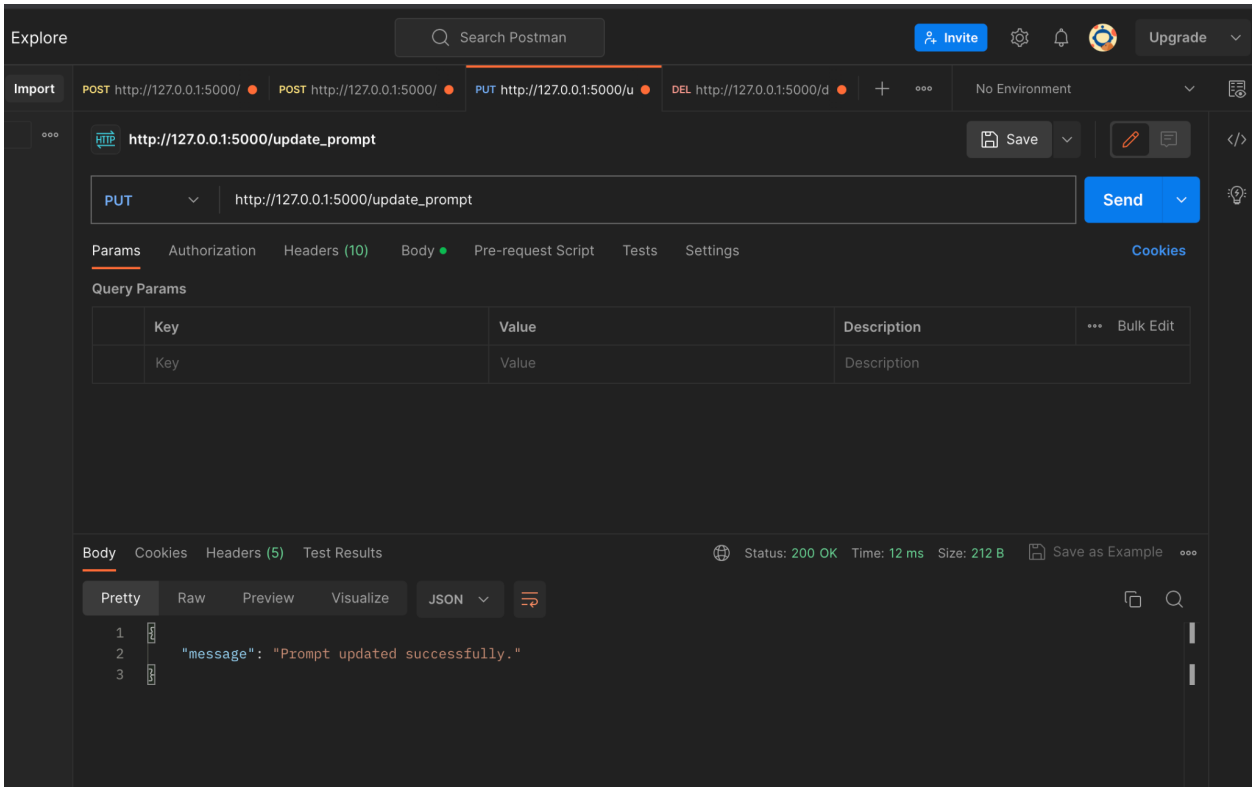
curl -X POST -H "Content-Type: application/json" -d '{"prompt_index": 0}' http://127.0.0.1:5000/get_response



Note: Error due to billing information not given in OpenAI

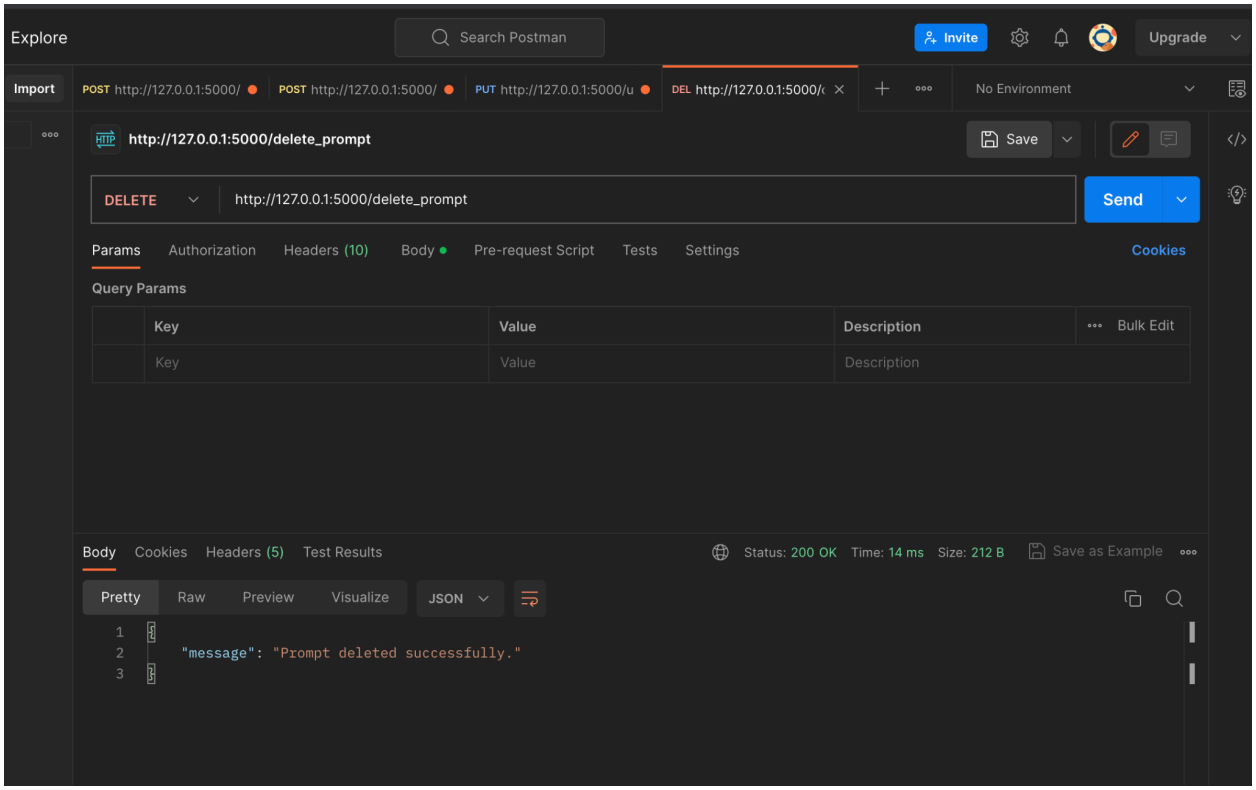
Update:

curl -X PUT -H "Content-Type: application/json" -d '{"prompt_index": 0, "new_prompt": "This is an updated prompt."}' http://127.0.0.1:5000/update_prompt



Delete:

curl -X DELETE -H "Content-Type: application/json" -d '{"prompt_index": 0}' http://127.0.0.1:5000/delete_prompt


Explore
Search Postman
Invite
Upgrade
Import
POST http://127.0.0.1:5000/
POST http://127.0.0.1:5000/
PUT http://127.0.0.1:5000/u
DEL http://127.0.0.1:5000/
No Environment
http://127.0.0.1:5000/delete_prompt
Save
DELETE
http://127.0.0.1:5000/delete_prompt
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 14 ms
Size: 212 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
1
2
3
"message": "Prompt deleted successfully."