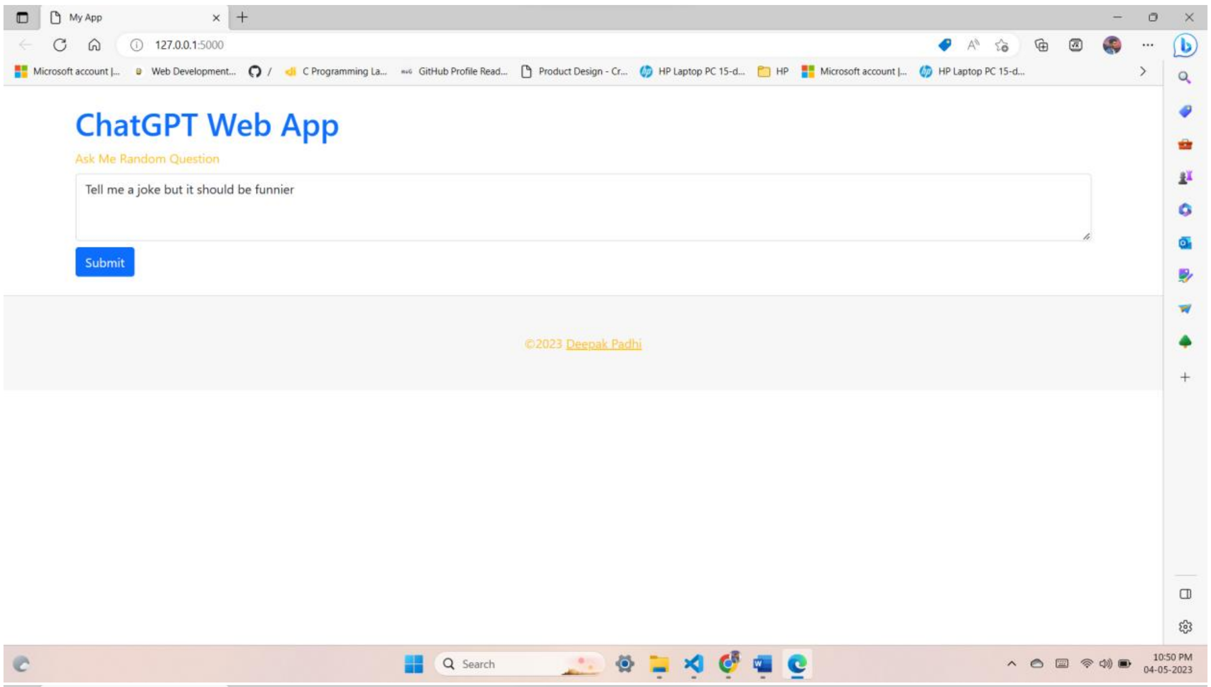
ChatGPT API Integration and Flask Web App Development

Objective: -

The purpose of this assignment is to evaluate your skills in Python, Flask, API integration, prompt engineering, and web development. You will develop a simple web application that integrates with the ChatGPT API and demonstrates your ability to work with the mentioned technologies.

Result: - Built Using (Python, Flask, HTML, CSS, Bootstrap, JavaScript)



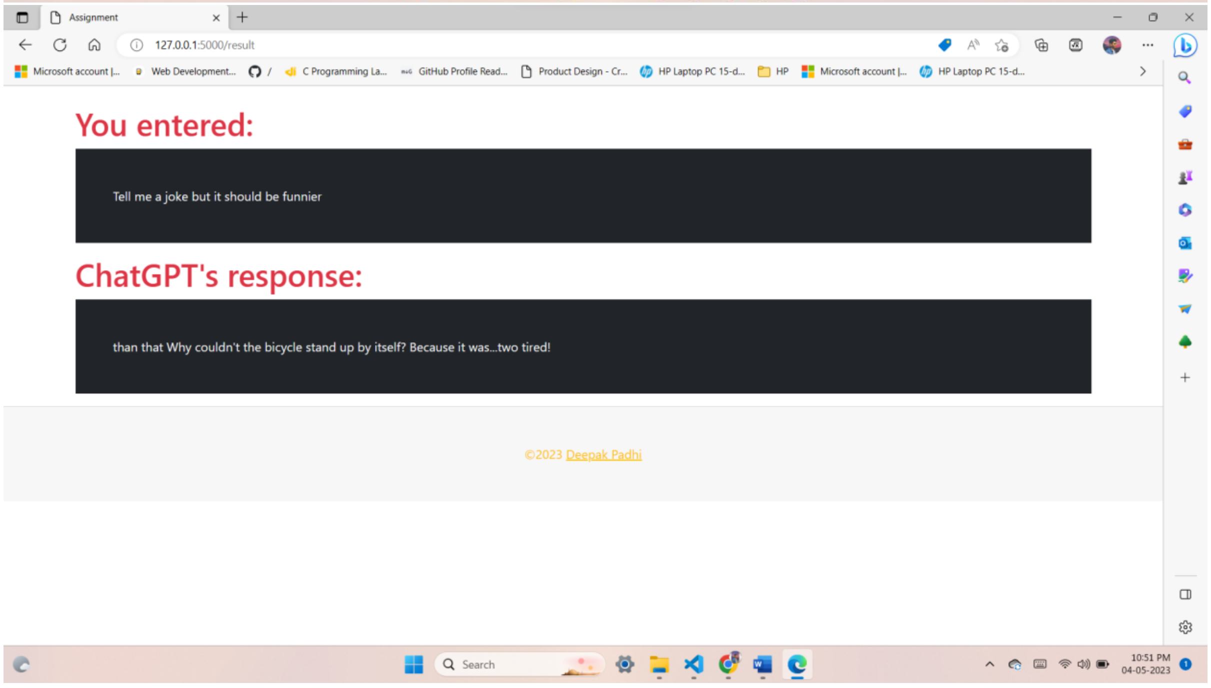
ChatGPT Web App

Ask Me Random Question

Tell me a joke but it should be funnier

Submit

©2023 Deepak Padhi



You entered:

Tell me a joke but it should be funnier

ChatGPT's response:

than that Why couldn't the bicycle stand up by itself? Because it was...two tired!

©2023 Deepak Padhi

Task 1: Create a Python Flask Web Application

1. Set up a Python Flask web application with the following features:
   - A homepage with a simple form that accepts user input for a question or prompt.
   - A results page that displays the message entered.

2. Integrate the ChatGPT API:
   - Use Python to make API calls to the ChatGPT API.
   - Design and implement ChatGPT prompts that provide context for user inputs.
   - Include error handling for API errors or failed requests.
   - Implement it in the flask app you created.

3. Implement basic web development principles:
   - Use HTML, CSS, and JavaScript for frontend design.
   - Ensure the web application is responsive and works on different screen sizes.
   - Ensure compatibility with popular web browsers.
   - Implement it in the flask app you created.

Task 2: Write Test Cases and Connect Flask Endpoints

1. Write test cases for the Python Flask app:
   - Test the ChatGPT API integration.
   - Test the proper rendering of the homepage and results page.

2. Connect Flask endpoints and make API calls to them:
   - Implement the necessary routes and controllers.
   - Make sure the web application is functional and can make API calls as required.

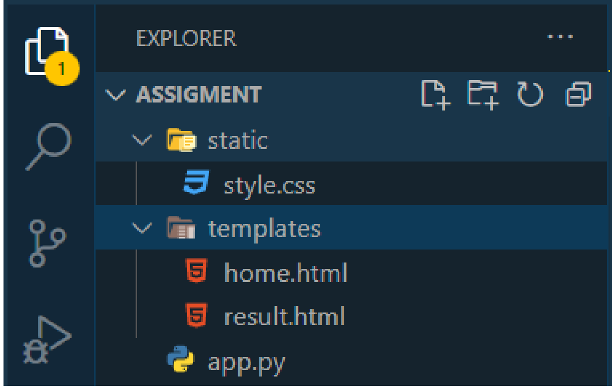
Solution: -

- To set up a Python Flask web application with a homepage that accepts user input and displays the results on a separate page, we first install the necessary package i.e. flask by typing

```
pip install flask
```

- Create app.py to write the code and create home.html & result.html into templates folder

```
Our file structure would look like this
```



- Install openai package via pip

```
pip install openai
```

- In app.py, we have created two routes:
  one for the homepage (/) and one for the results page (/result).
  - The home() function renders the home.html template, which contains a simple form for the user to enter a message.
  - The result() function retrieves the message from the form and renders the result.html template, passing the message as a variable.

```python
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/result', methods=['POST'])
def result():
    message = request.form['message']
    return render_template('result.html', message=message)
```

- We Import the openai package and set up authentication (API key) and create a function that takes user input and sends it as a prompt to the ChatGPT API, which then generates a response. this function also handles any errors that may occur:

```
import openai
openai.api_key = "YOUR_API_KEY"

def generate_response(prompt):
    try:
        response = openai.Completion.create(
            engine="text-davinci-002",
            prompt=prompt,
            max_tokens=100,
            n=1,
            stop=None,
            temperature=0.5,
        )
        return response.choices[0].text
    except Exception as e:
        return f"Error: {e}"
```

- Here, we are using the text-davinci-002 engine to generate a response. We are requesting a maximum of 100 tokens, a single response, and a temperature of 0.5 (which controls the creativity of the response). We are also handling any exceptions that may occur.

- Finally modify the result() function in the Flask app to include the prompt and response from ChatGPT:

```
@app.route('/result', methods=['POST'])
def result():
    prompt = request.form['message']
    response = generate_response(prompt)
    return render_template('result.html', prompt=prompt, response=response)
```

Here, we are retrieving the prompt entered by the user, passing it to the generate_response() function to generate a response, and then rendering the result.html template with both the prompt and the response.

- Finally, we add a guard to only execute the test suite if the __name__ variable is equal to '__main__', which means that the file is being run directly and not imported as a module.

- Final flask app.py would look like this

```python
#importing required packages
import openai
from flask import Flask, render_template, request


app = Flask(__name__)

#assigning api key
openai.api_key = "YOUR_API_KEY"

#function that takes user input and sends it as a prompt to the ChatGPT API
def generate_response(prompt):
    try:
        response = openai.Completion.create(
            engine="text-davinci-002",
            prompt=prompt,
            max_tokens=100,
            n=1,
            stop=None,
            temperature=0.5,
        )
        return response.choices[0].text
    except Exception as e:
        return f"Error: {e}"

#route for home and its function
@app.route('/')
def home():
    return render_template('home.html')

#route for result and its function to include the response from ChatGPT
@app.route('/result', methods=['POST'])
def result():
    prompt = request.form['message']
    response = generate_response(prompt)
    return render_template('result.html', prompt=prompt, response=response)

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

- home.html (with customization)

```html
<!DOCTYPE html>
<html>

<head>
  <title>My App</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/static/style.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
</head>

<body>
  <div class="container mt-4">
    <h1 class="text-primary h1">ChatGPT Web App</h1>
    <form action="/result" method="POST">
      <div class="form-group">
        <label for="message" class="text-warning">Ask Me Random Question</label>
        <textarea class="form-control mt-2 form-text" id="message" name="message" rows="3"></textarea>
      </div>
      <button type="submit" class="btn btn-primary mt-2">Submit</button>
    </form>
  </div>
  <footer class="card-footer text-warning p-5 mt-4 text-center">
    ©<span id="year"></span> <a href="https://dnoobnerd.netlify.app/"
      class="text-decoration-underline text-warning">Deepak Padhi</a>
  </footer>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
    crossorigin="anonymous">
  </script>
  <script>
    document.getElementById("year").innerHTML = new Date().getFullYear();
  </script>
</body>

</html>
```

- result.html (with customization)

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Assignment</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/static/style.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
  </head>
  <body>
    <div class="container mt-4">
      <h1 class="text-danger h1">You entered:</h1>
      <p class="text-wrap text-sm-start text-light bg-dark p-5">{{ prompt }}</p>
      <h1 class="text-danger h1">ChatGPT's response:</h1>
      <p class="text-wrap text-sm-start text-light bg-dark p-5">{{ response }}</p>
    </div>
    <footer class="card-footer text-warning p-5 mt-2 text-center">
      ©<span id="year"></span> <a href="https://dnoobnerd.netlify.app/"
        class="text-decoration-underline text-warning">Deepak Padhi</a>
    </footer>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
      crossorigin="anonymous">
    </script>
    <script>
      document.getElementById("year").innerHTML = new Date().getFullYear();
    </script>
  </body>
</html>
```

Testing the Proper Rendering of the Homepage and Results Page

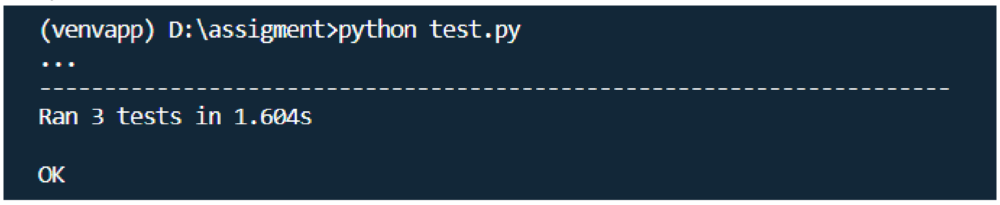
- Install the pytest for flask package

```
pip install pytest-flask
```

- Create a test.py file and add the final code

```
#importing required packages
import unittest
import app
#define a class that inherits the test case
class TestApp(unittest.TestCase):
    def setUp(self):
        self.app = app.app.test_client()

#generates a sample prompt and checks that the response returned is a string
    def test_generate_response(self):
        prompt = "Q: What is life?\nA:"
        response = app.generate_response(prompt)
        self.assertIsInstance(response, str)

#find the label with ask me a random question and assert the question in the textbox and sends a
GET request to the homepage
    def test_homepage(self):
        response = self.app.get('/')
        self.assertEqual(response.status_code, 200)
        self.assertIn(b'Ask Me Random Question', response.data)

#sends a POST request to the homepage with a status code 200
    def test_submit_message(self):
        response = self.app.post('/result', data=dict(message='What is life?'),
follow_redirects=True)
        self.assertEqual(response.status_code, 200)
        self.assertIn(b'You entered:', response.data)

if __name__ == '__main__':
    unittest.main()
```

- Output



```
(venvapp) D:\assigment>python test.py
...
----------------------------------------------------------------------
Ran 3 tests in 1.604s

OK
```

Thanks for reading the document, I don't have any professional knowledge about technical documentation but I know the basic ones to create, This document only contains the code and its simple explanation

GitHub: https://github.com/deepakpadhi986/py-assign