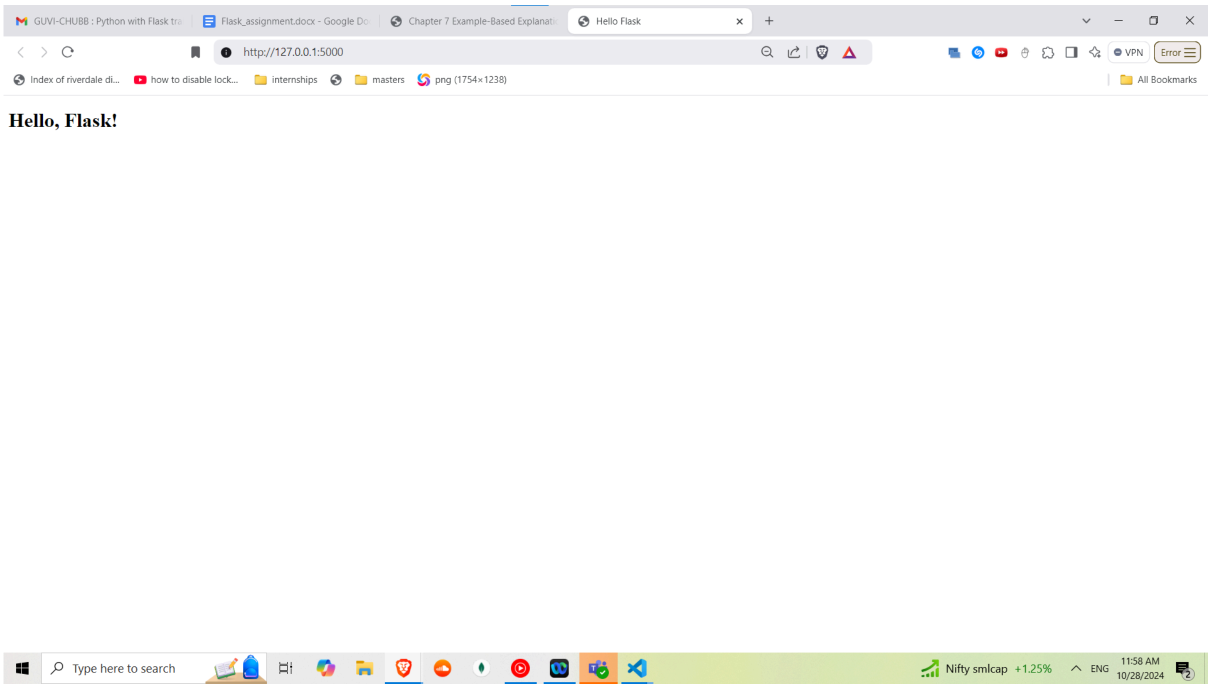
Weekly Assignment 27-Oct-24 : Python with Flask

Submission Date : 31-Oct-24 (Before 12:00 PM)

1. Hello Flask Website

- **Task**: Create a simple "Hello, World!" Flask application.
- **Requirements**: Make a single route (`/`) that displays “Hello, Flask!” on a web page.
- **Hint**: Start by setting up a basic Flask app with a single route. Use `app.route()` to set the URL path.



Hello, Flask!

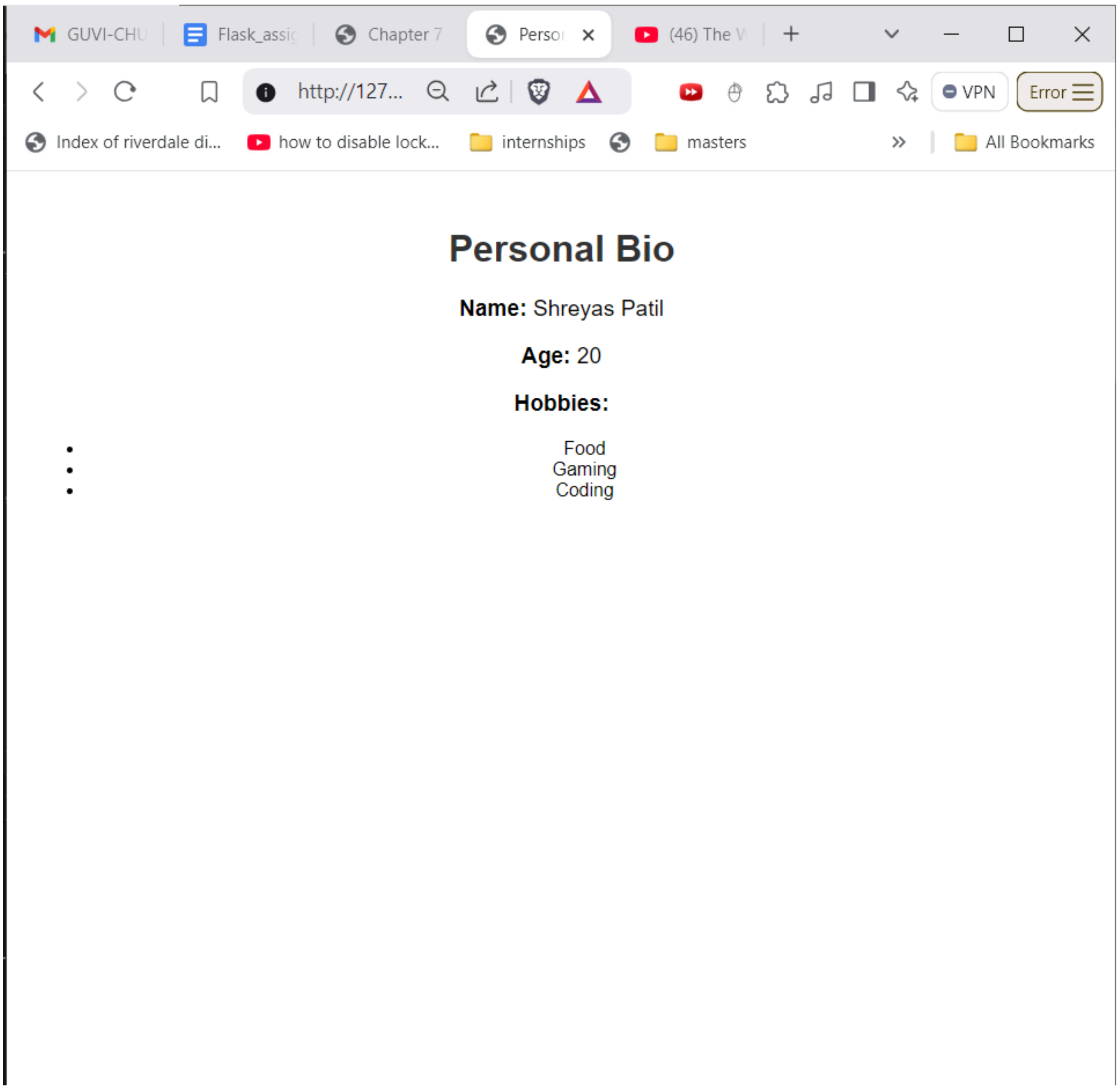
-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Hello Flask</title>
</head>
<body>
    <h1>Hello, Flask!</h1>
</body>
</html>
```

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def hello_flask():
    return render_template("index.html")

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

2. Personal Bio Page

- **Task**: Design a simple personal bio page.
- **Requirements**: Add a route (`/bio`) with basic information like name, age, and hobbies displayed in HTML.
- **Hint**: Use Flask's render_template function and create a basic HTML file to display personal details.



Personal Bio

Name: Shreyas Patil

Age: 20

Hobbies:

- Food
- Gaming
- Coding

•

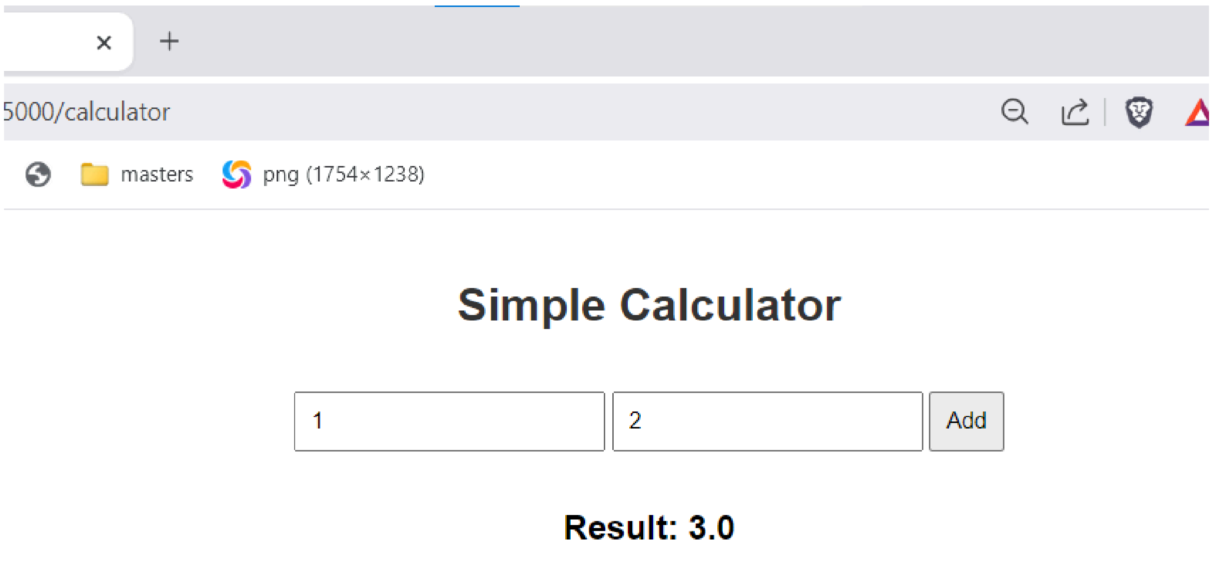
```
@app.route('/bio')
def bio():
    details = {
        "name": "Shreyas Patil",
        "age": 20,
        "hobbies": ["Food", "Gaming", "Coding"]
    }
    return render_template("bio.html", details=details)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Personal Bio</title>
    <style>
        body { font-family: Arial, sans-serif; text-align:
center; padding: 20px; }
        h1 { color: #333; }
        p { font-size: 1.2em; }
    </style>
</head>
<body>
    <h1>Personal Bio</h1>
    <p><strong>Name:</strong> {{ details.name }}</p>
    <p><strong>Age:</strong> {{ details.age }}</p>
    <p><strong>Hobbies:</strong></p>
    <ul>
        {% for hobby in details.hobbies %}
            <li>{{ hobby }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

-

3. Calculator App

- **Task**: Build a simple calculator that can add two numbers.
- **Requirements**: Create a form where users enter two numbers. Display the result on submission.
- **Hint**: Use HTML forms and handle data in Flask using the `request.form` method to get the inputs.



Simple Calculator

1 2 Add

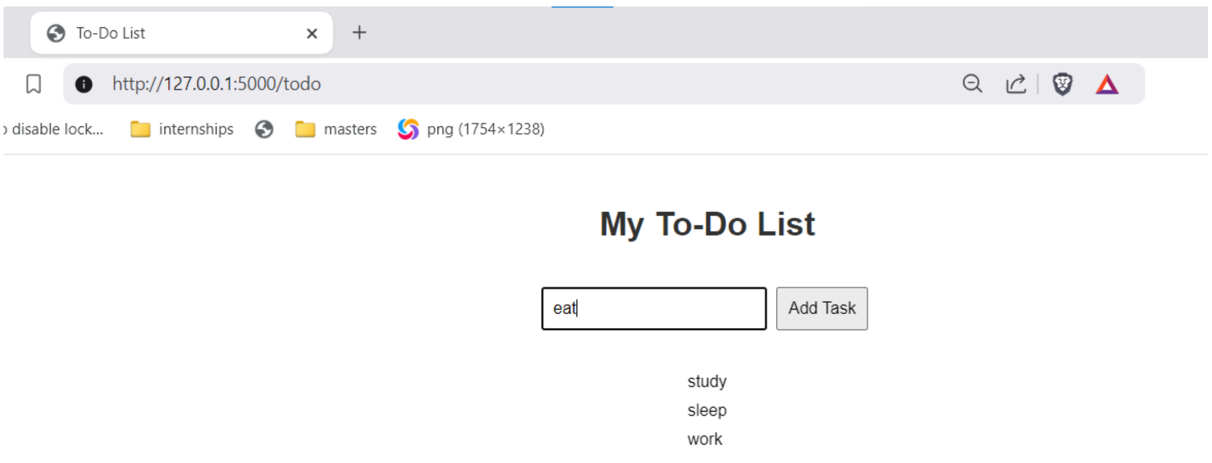
Result: 3.0

```
@app.route('/calculator', methods=['GET', 'POST'])
def calculator():
    result = None
    if request.method == 'POST':
        try:
            num1 = float(request.form['num1'])
            num2 = float(request.form['num2'])
            result = num1 + num2
        except ValueError:
            result = "Invalid input. Please enter numeric
values."
    return render_template("calculator.html", result=result)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Calculator</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex;
flex-direction: column; align-items: center; padding: 20px; }
        h1 { color: #333; }
        form { margin: 20px 0; }
        input[type="text"], button { padding: 10px; font-size:
1em; }
    </style>
</head>
<body>
    <h1>Simple Calculator</h1>
    <form method="POST" action="/calculator">
        <input type="text" name="num1" placeholder="Enter first
number" required>
        <input type="text" name="num2" placeholder="Enter second
number" required>
        <button type="submit">Add</button>
    </form>
    {% if result is not none %}
        <h2>Result: {{ result }}</h2>
    {% endif %}
</body>
</html>
```

4. Mini To-Do List

- **Task**: Create a basic to-do list web app where users can add tasks.
- **Requirements**: Implement a form to add tasks and display the tasks on the same page.
- **Hint**: Use a list to store tasks temporarily and a `POST` method to add new tasks to the list.



To-Do List

http://127.0.0.1:5000/todo

My To-Do List

eat

Add Task

study
sleep
work

-

```
@app.route('/todo', methods=['GET', 'POST'])
def todo():
    if request.method == 'POST':
        task = request.form.get('task')
        if task:
            tasks.append(task)
        return redirect(url_for('todo'))  # avoid  resubmission
    return render_template("todo.html", tasks=tasks)
```

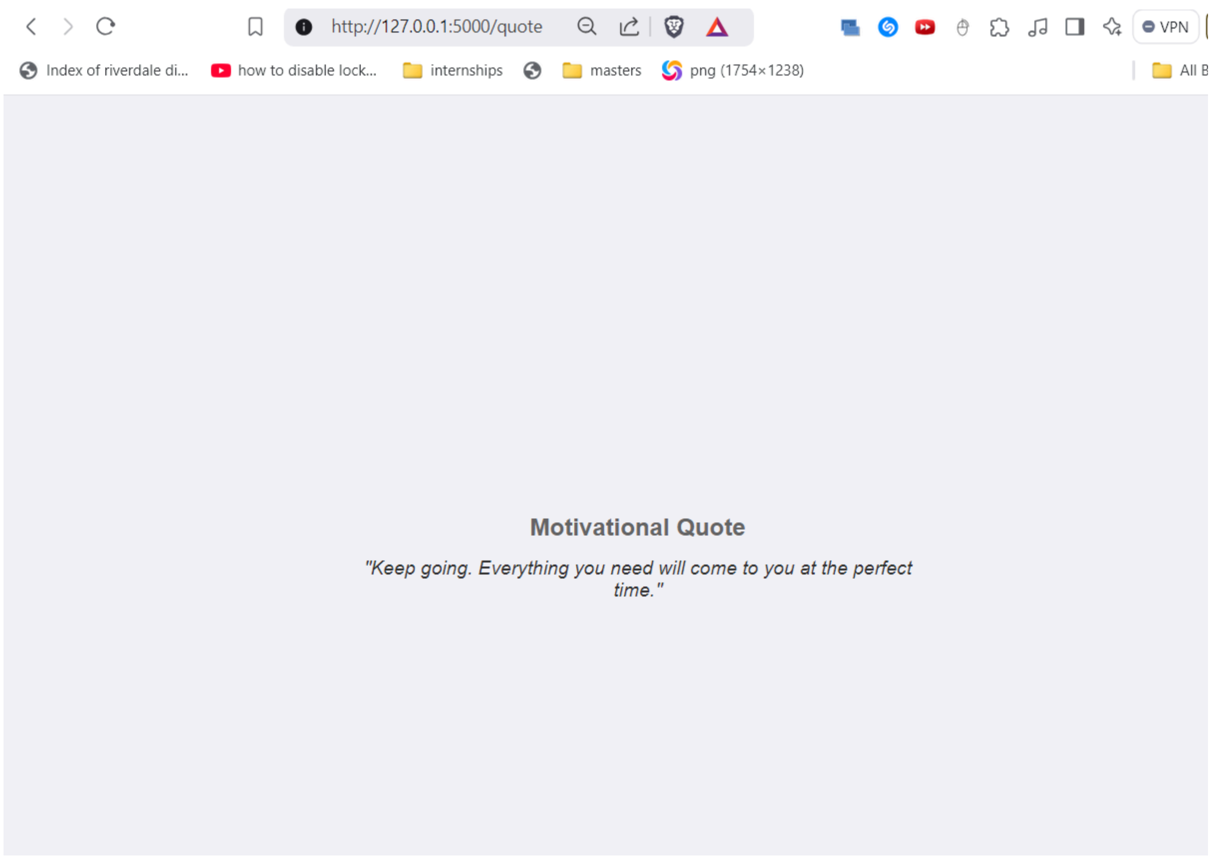
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>To-Do List</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex;
flex-direction: column; align-items: center; padding: 20px; }
        h1 { color: #333; }
        form { margin: 20px 0; }
        input[type="text"], button { padding: 10px; font-size:
```

```
1em; margin-right: 5px; }
        ul { list-style-type: none; padding: 0; }
        li { padding: 5px 0; }
    </style>
</head>
<body>
    <h1>My To-Do List</h1>
    <form method="POST" action="/todo">
        <input type="text" name="task" placeholder="Enter a new
task" required>
        <button type="submit">Add Task</button>
    </form>
    <ul>
        {% for task in tasks %}
            <li>{{ task }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

-

5. Random Quote Generator

- **Task**: Create an app that displays a random motivational quote from a predefined list each time the page is refreshed.
- **Requirements**: Display one random quote from a list of quotes every time the user visits the `/quote` page.
- **Hint**: Use Python's `random.choice()` to select a quote from a list and display it using HTML.



Motivational Quote

"Keep going. Everything you need will come to you at the perfect time."

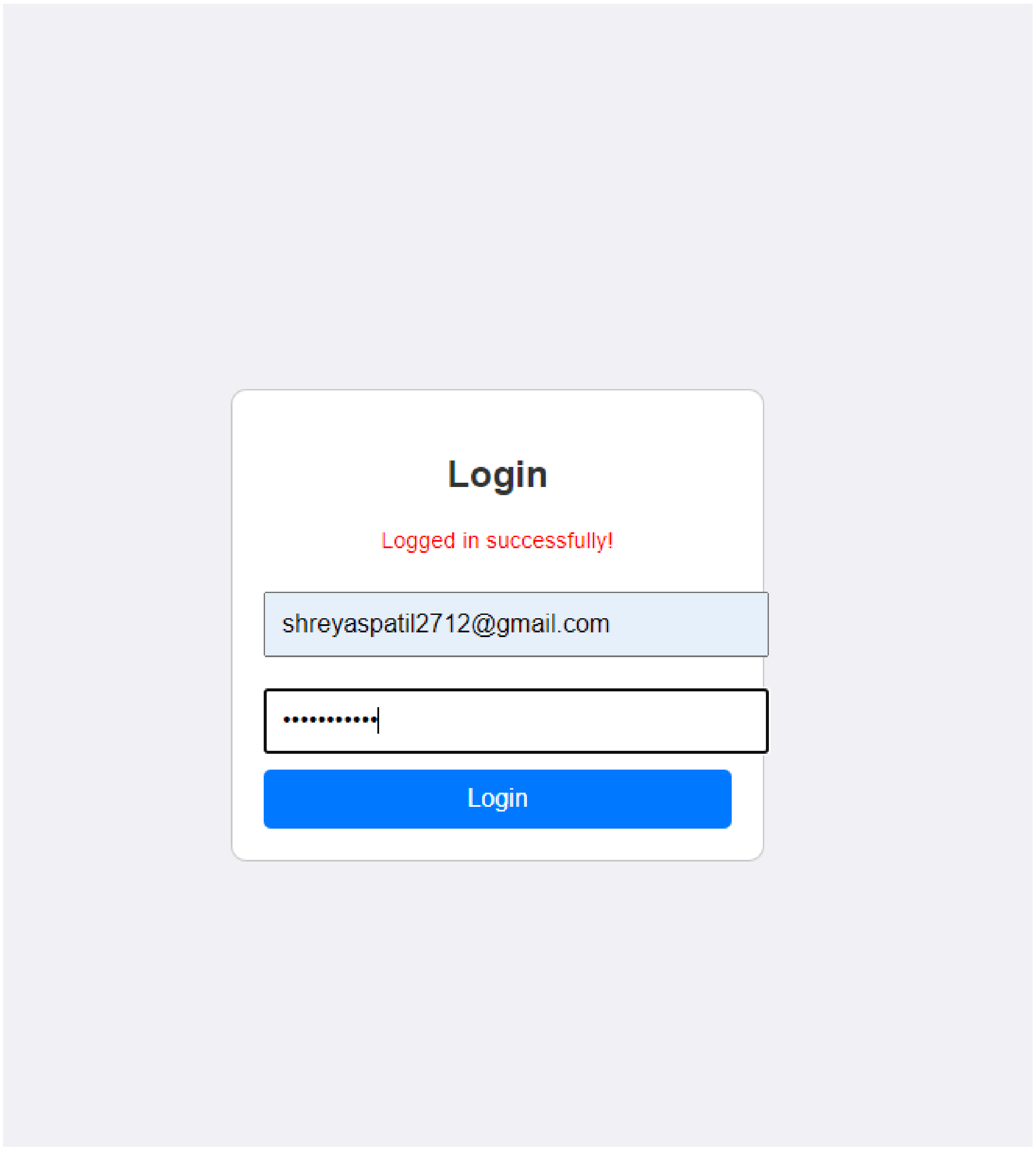
```
import random

quotes = [
    "Believe you can and you're halfway there.",
    "Act as if what you do makes a difference. It does.",
    "Success is not final, failure is not fatal: It is the
courage to continue that counts.",
    "Keep going. Everything you need will come to you at the
perfect time.",
    "You don’t have to be perfect to be amazing.",
    "The only limit to our realization of tomorrow is our doubts
of today."
]

@app.route('/quote')
def quote():
    random_quote = random.choice(quotes)
    return render_template("quote.html", quote=random_quote)
```

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Random Quote</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex;
justify-content: center; align-items: center; height: 100vh;
margin: 0; background-color: #f4f4f9; color: #333; text-align:
center; padding: 20px; }
        .quote-container { max-width: 600px; }
        h1 { font-size: 1.5em; color: #666; }
        p { font-size: 1.2em; font-style: italic; margin-top:
10px; }
    </style>
</head>
<body>
    <div class="quote-container">
        <h1>Motivational Quote</h1>
        <p>"{{ quote }}"</p>
    </div>
</body>
</html>
```

6. Simple Login Page

- **Task**: Build a basic login form with username and password fields.
- **Requirements**: Display a welcome message if the username is “user” and the password is “password.”
- **Hint**: Use `POST` requests and `if` statements to check login credentials.

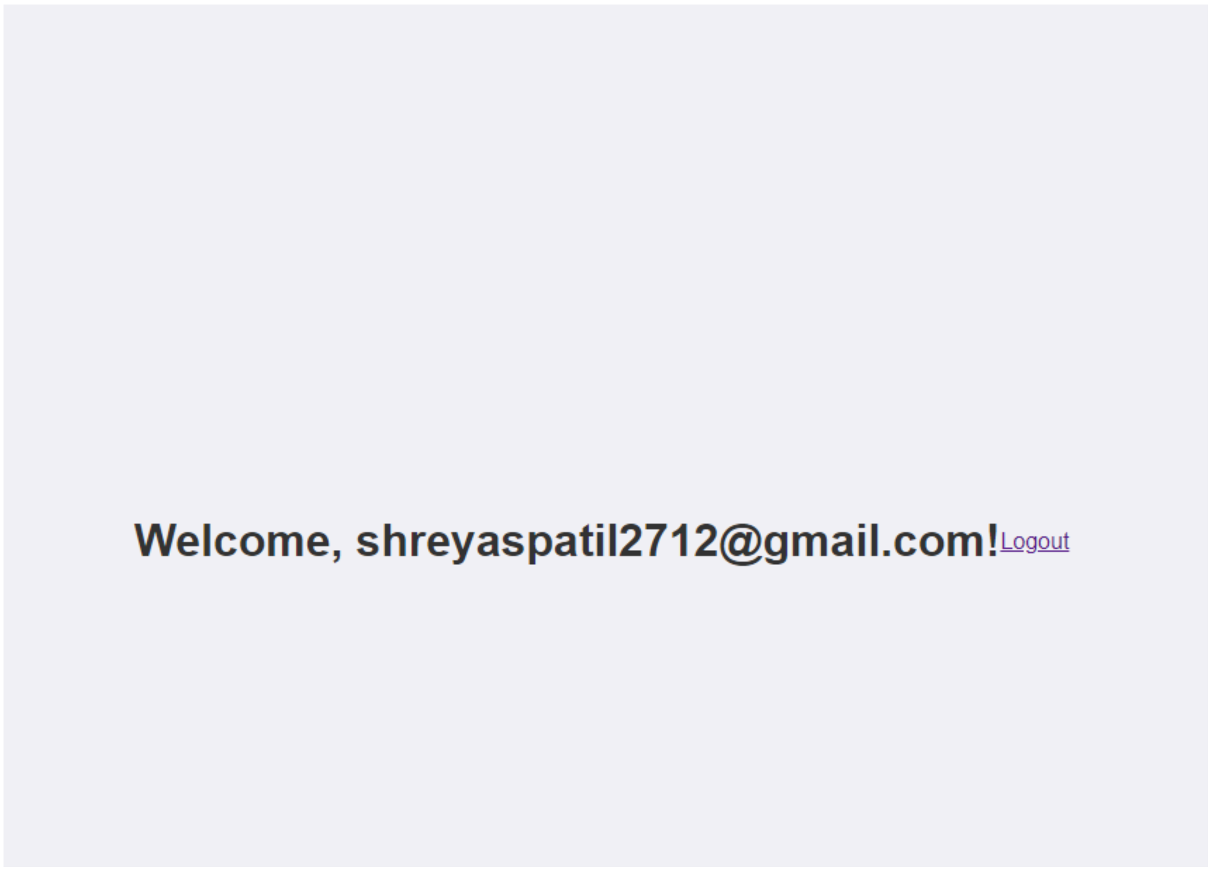


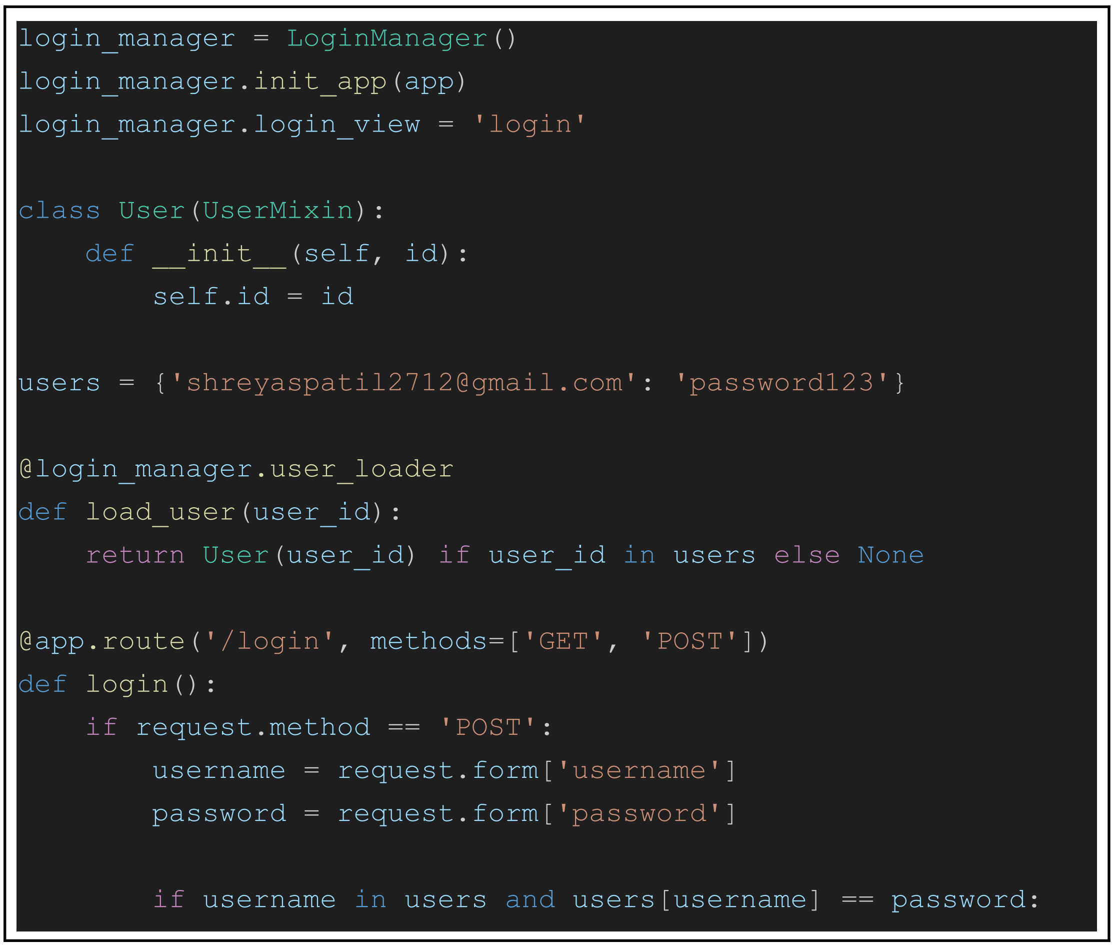
Login

Logged in successfully!

shreyaspatil2712@gmail.com

•••••••••••


Welcome, shreyaspatil2712@gmail.com!Logout


```python
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

class User(UserMixin):
    def __init__(self, id):
        self.id = id

users = {'shreyaspatil2712@gmail.com': 'password123'}

@login_manager.user_loader
def load_user(user_id):
    return User(user_id) if user_id in users else None

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if username in users and users[username] == password:
            user = User(username)
            login_user(user)
            flash('Logged in successfully!')
            return redirect(url_for('welcome'))
        else:
            flash('Invalid username or password. Please try 
again.')

    return render_template('login.html')


@app.route('/welcome')
@login_required
def welcome():
    return render_template('welcome.html', name=current_user.id)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out.')
    return redirect(url_for('login'))
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, 
initial-scale=1.0">
    <title>Login</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex; 
justify-content: center; align-items: center; height: 100vh; 
background-color: #f4f4f9; margin: 0; }
        .login-container { max-width: 300px; width: 100%; 
text-align: center; padding: 20px; border: 1px solid #ccc; 
border-radius: 10px; background-color: #fff; }
        h2 { color: #333; }
        input { width: 100%; padding: 10px; margin: 10px 0; 
font-size: 1em; }
```

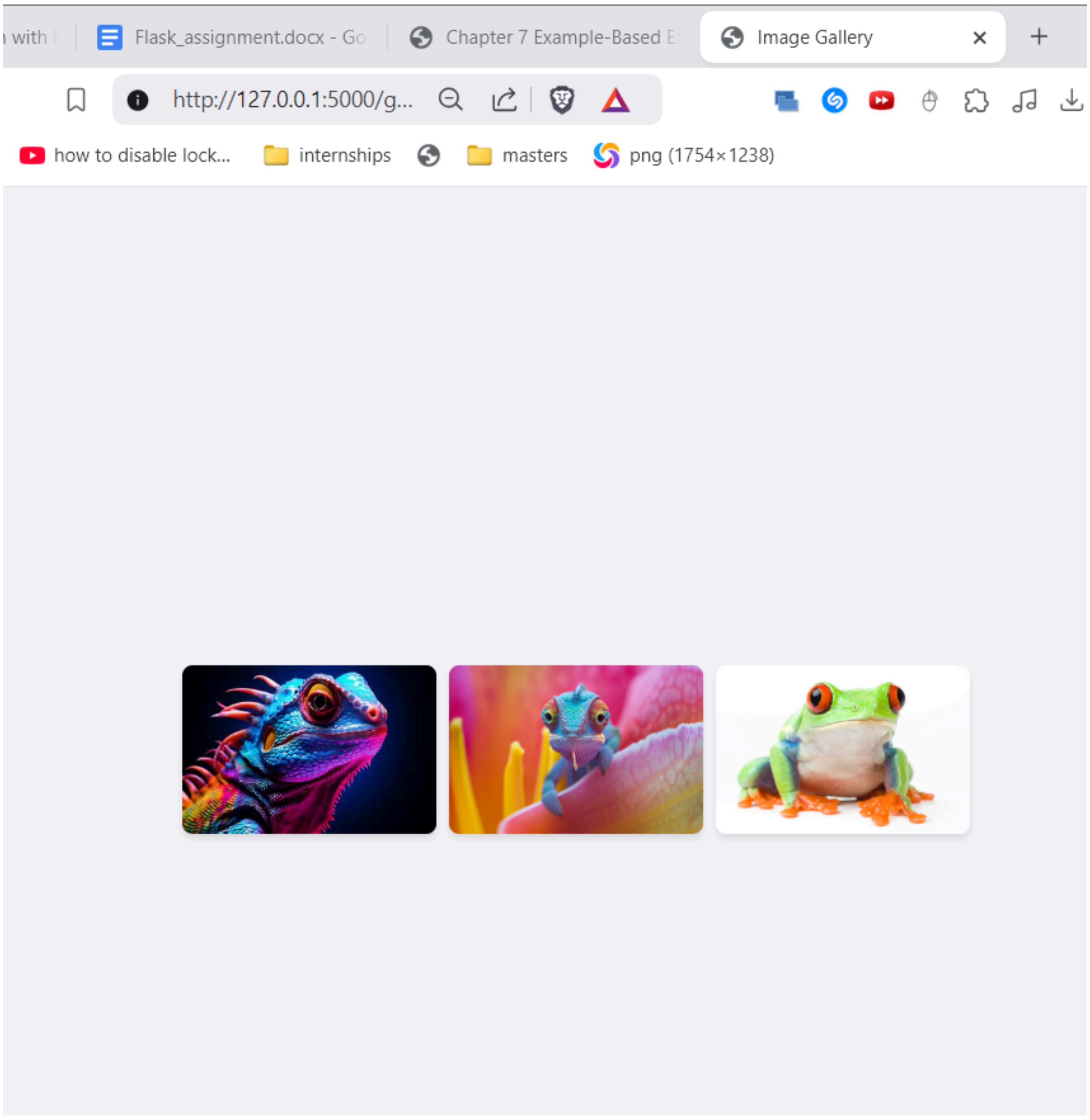
```
        button { width: 100%; padding: 10px; font-size: 1em;
background-color: #007bff; color: #fff; border: none;
border-radius: 5px; cursor: pointer; }
        .error { color: red; font-size: 0.9em; }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login</h2>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <p class="error">{{ messages[0] }}</p>
            {% endif %}
        {% endwith %}
        <form method="POST" action="{{ url_for('login') }}">
            <input type="text" name="username"
placeholder="Username" required>
            <input type="password" name="password"
placeholder="Password" required>
            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Welcome</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex;
justify-content: center; align-items: center; height: 100vh;
background-color: #f4f4f9; margin: 0; }
        h1 { color: #333; }
    </style>
</head>
<body>
```

```
    <h1>Welcome, {{ name }}!</h1>
    <a href="{{ url_for('logout') }}">Logout</a>
</body>
</html>
```

7. Image Gallery

- **Task**: Create a simple gallery page that displays three static images.
- **Requirements**: Display three images side by side on a page.
- **Hint**: Use HTML `<img>` tags in your HTML template and store the images in a `/static` folder in the project.


with
Flask_assignment.docx - Go
Chapter 7 Example-Based E
Image Gallery
http://127.0.0.1:5000/g...
how to disable lock...
internships
masters
png (1754×1238)

-

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

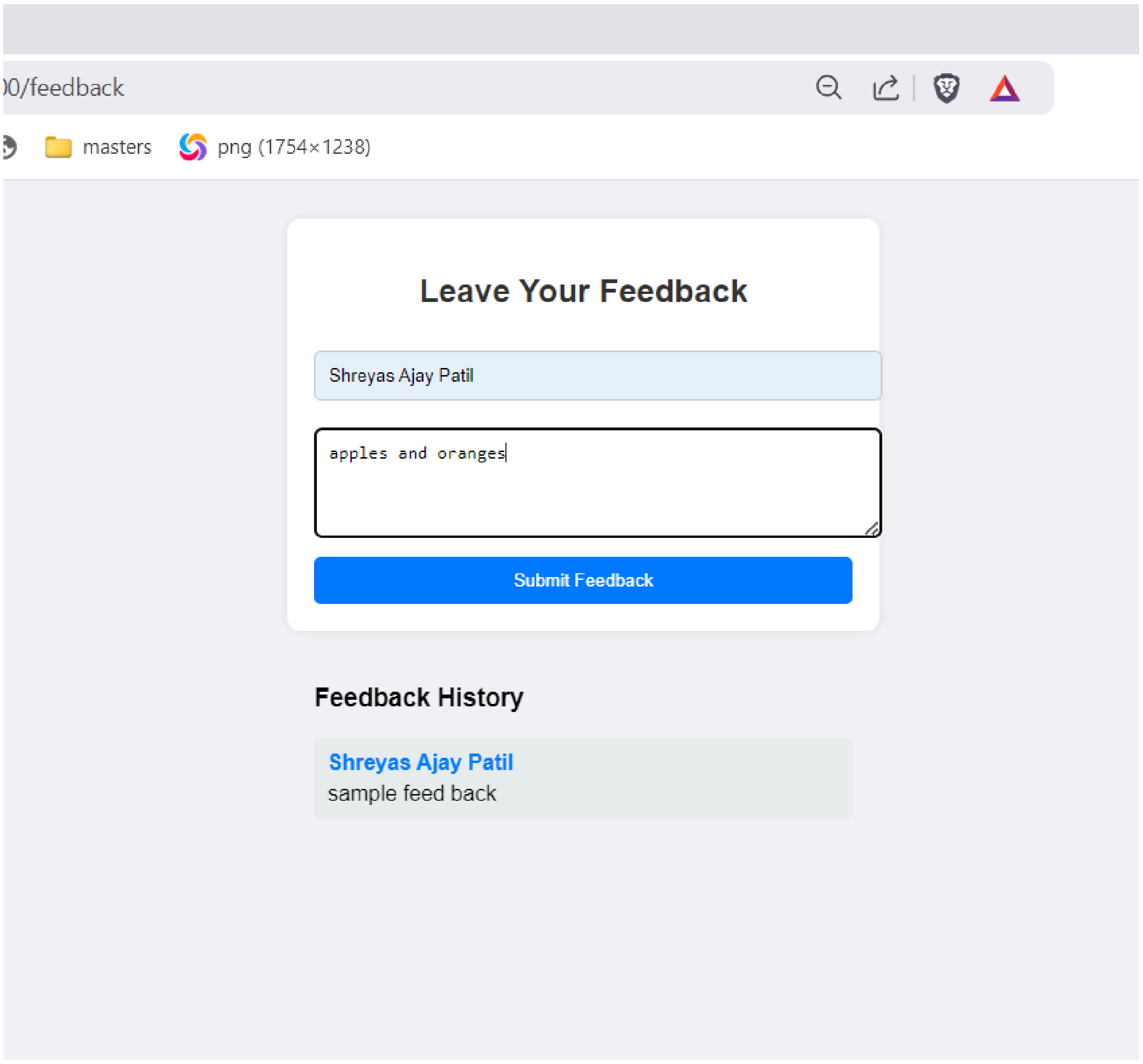
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Image Gallery</title>
    <style>
```

```
        body { font-family: Arial, sans-serif; display: flex;
justify-content: center; align-items: center; height: 100vh;
margin: 0; background-color: #f4f4f9; }
        .gallery-container { display: flex; gap: 10px; }
        .gallery-container img { width: 200px; height: auto;
border-radius: 8px; box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
    </style>
</head>
<body>
    <div class="gallery-container">
        <img src="{{ url_for('static', filename='image1.jpg')
}}" alt="Image 1">
        <img src="{{ url_for('static', filename='image2.jpg')
}}" alt="Image 2">
        <img src="{{ url_for('static', filename='image3.jpg')
}}" alt="Image 3">
    </div>
</body>
</html>
```

-

8. Feedback Form

- **Task**: Make a feedback form that saves users' names and feedback temporarily.
- **Requirements**: Save the submitted feedback to a list and display the list on the same page.
- **Hint**: Use Python lists or dictionaries to store each feedback entry, and display feedback history at the bottom of the page.

- 
)0/feedback

masters   png (1754×1238)

Leave Your Feedback

Shreyas Ajay Patil

apples and oranges

Submit Feedback

Feedback History

Shreyas Ajay Patil
sample feed back

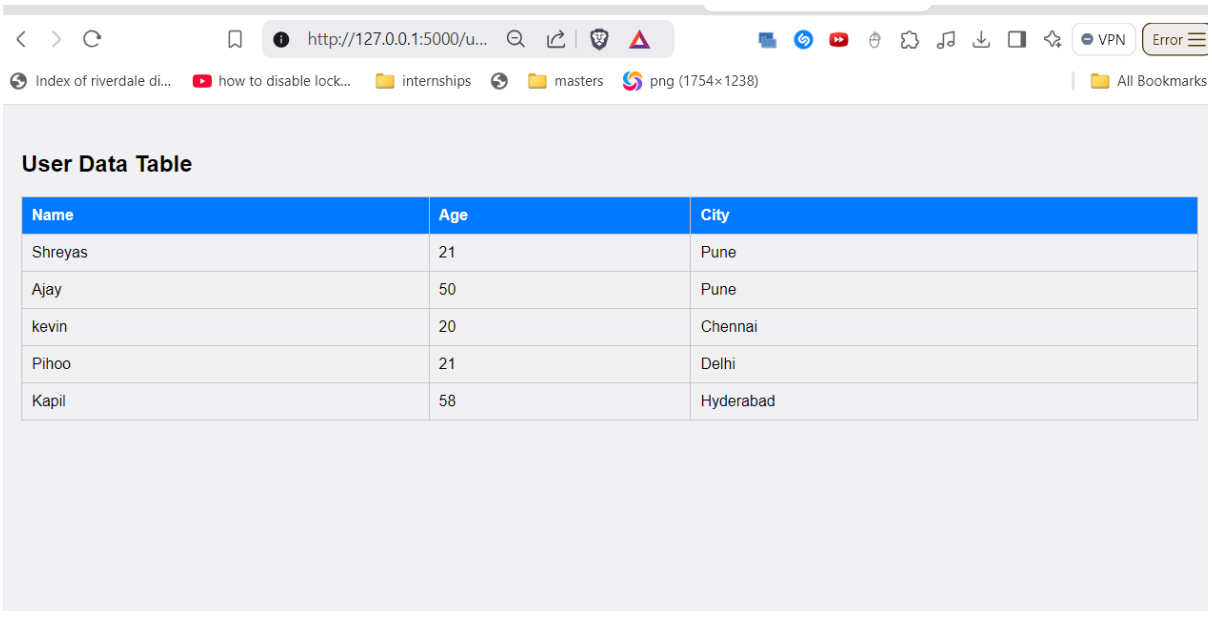
```
feedback_list = []

@app.route('/feedback', methods=['GET', 'POST'])
def feedback():
    if request.method == 'POST':
        name = request.form.get('name')
        feedback_text = request.form.get('feedback')

        if name and feedback_text:
            feedback_list.append({'name': name, 'feedback':
feedback_text})

        return redirect(url_for('feedback'))
    return render_template('feedback.html',
feedback_list=feedback_list)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Feedback Form</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex;
flex-direction: column; align-items: center; padding: 20px;
background-color: #f4f4f9; }
        .form-container { max-width: 400px; width: 100%;
padding: 20px; background-color: #fff; border-radius: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }
        h2 { text-align: center; color: #333; }
        input, textarea { width: 100%; padding: 10px;
margin-top: 10px; margin-bottom: 10px; border-radius: 5px;
border: 1px solid #ccc; }
        button { width: 100%; padding: 10px; background-color:
#007bff; color: #fff; border: none; border-radius: 5px; cursor:
pointer; }
        .feedback-list { margin-top: 20px; width: 100%;
max-width: 400px; }
        .feedback-item { padding: 10px; background-color:
#e9ecef; border-radius: 5px; margin-bottom: 10px; }
        .feedback-item h4 { margin: 0; color: #007bff; }
        .feedback-item p { margin: 5px 0 0; }
    </style>
</head>
<body>
    <div class="form-container">
        <h2>Leave Your Feedback</h2>
        <form method="POST" action="{{ url_for('feedback') }}">
            <input type="text" name="name" placeholder="Your
Name" required>
            <textarea name="feedback" rows="4" placeholder="Your
Feedback" required></textarea>
```

```
            <button type="submit">Submit Feedback</button>
        </form>
    </div>

    <div class="feedback-list">
        <h3>Feedback History</h3>
        {% for item in feedback_list %}
            <div class="feedback-item">
                <h4>{{ item.name }}</h4>
                <p>{{ item.feedback }}</p>
            </div>
        {% endfor %}
    </div>
</body>
</html>
```

9. Basic Data Table with Jinja

- **Task**: Display a table of users and their details (name, age, city).
- **Requirements**: Use a predefined list of dictionaries and render it in an HTML table.
- **Hint**: Use Jinja templating with `for` loops to iterate over the list and display the data in a table format.



User Data Table

Name	Age	City
Shreyas	21	Pune
Ajay	50	Pune
kevin	20	Chennai
Pihoo	21	Delhi
Kapil	58	Hyderabad

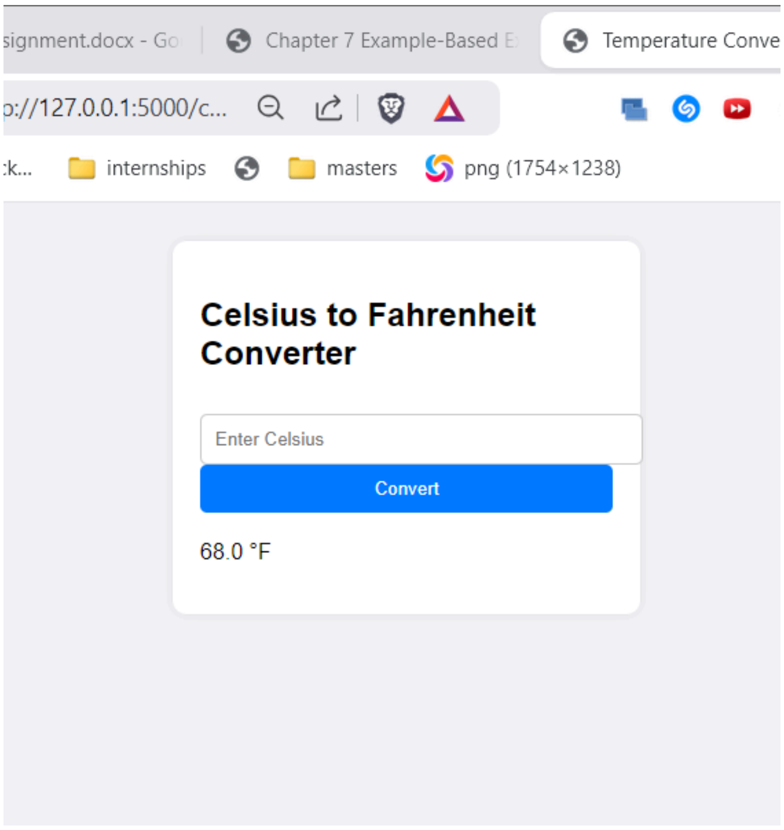
```python
users = [
    {'name': 'Shreyas', 'age': 21, 'city': 'Pune'},
    {'name': 'Ajay', 'age': 50, 'city': 'Pune'},
    {'name': 'kevin', 'age': 20, 'city': 'Chennai'},
    {'name': 'Pihoo', 'age': 21, 'city': 'Delhi'},
    {'name': 'Kapil', 'age': 58, 'city': 'Hyderabad'},
]

@app.route('/users')
def users_table():
    return render_template('users.html', users=users)
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>User Data Table</title>
    <style>
        body { font-family: Arial, sans-serif; padding: 20px;
background-color: #f4f4f9; }
        table { width: 100%; border-collapse: collapse; margin-top:
20px; }
        th, td { padding: 10px; border: 1px solid #ccc; text-align:
left; }
        th { background-color: #007bff; color: #fff; }
        tr:nth-child(even) { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h2>User Data Table</h2>
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>City</th>
            </tr>
        </thead>
        <tbody>
```

```
            {% for user in users %}
                <tr>
                    <td>{{ user.name }}</td>
                    <td>{{ user.age }}</td>
                    <td>{{ user.city }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

10. Temperature Converter

- **Task**: Create a simple temperature converter that converts Celsius to Fahrenheit.
- **Requirements**: Allow users to enter a Celsius value and display the converted Fahrenheit value.
- **Hint**: Use forms for input, `request.form` to get the Celsius value, and simple arithmetic in Python to calculate the Fahrenheit temperature.

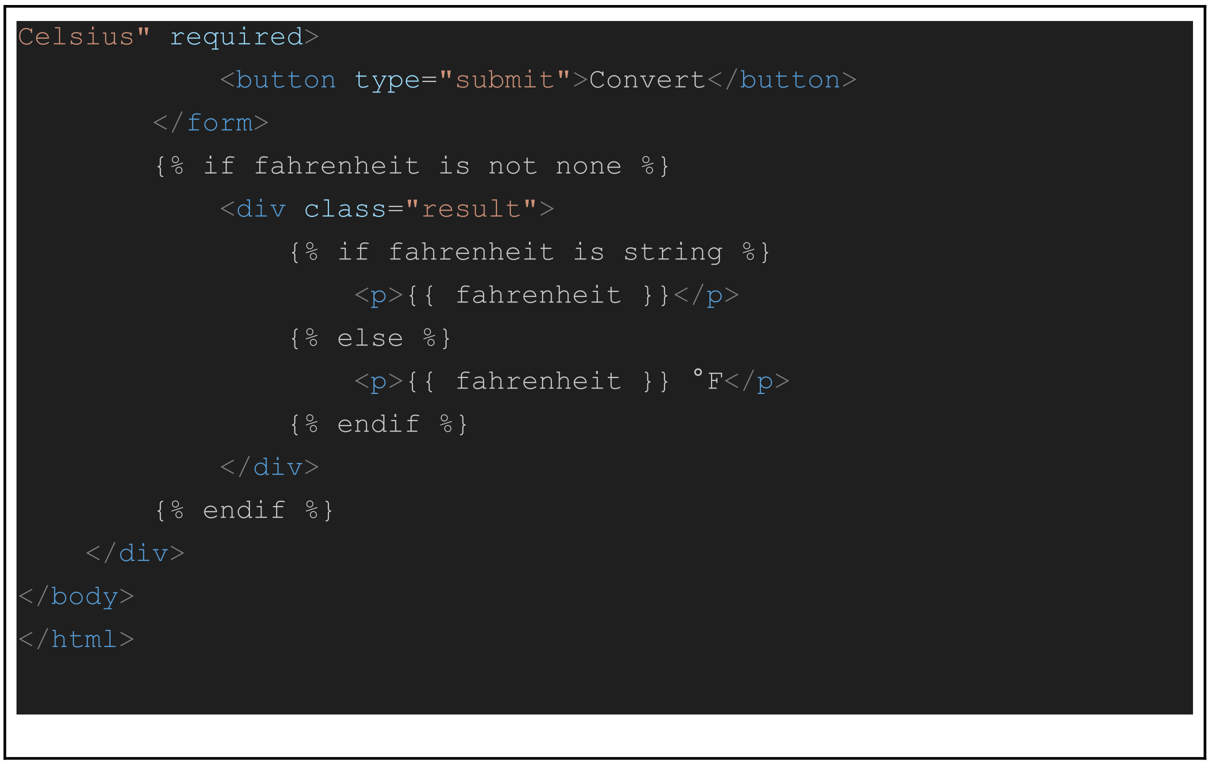


Celsius to Fahrenheit Converter

68.0 °F

```python
@app.route('/converter', methods=['GET', 'POST'])
def converter():
    fahrenheit = None
    if request.method == 'POST':
        celsius = request.form.get('celsius')
        if celsius:
            try:
                celsius = float(celsius)
                fahrenheit = (celsius * 9/5) + 32
            except ValueError:
                fahrenheit = "Invalid input. Please enter a number."

    return render_template('converter.html', fahrenheit=fahrenheit)
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Temperature Converter</title>
    <style>
        body { font-family: Arial, sans-serif; padding: 20px;
background-color: #f4f4f9; }
        .form-container { max-width: 300px; margin: auto; padding:
20px; background-color: #fff; border-radius: 10px; box-shadow: 0 0
10px rgba(0, 0, 0, 0.1); }
        input { width: 100%; padding: 10px; margin-top: 10px;
border-radius: 5px; border: 1px solid #ccc; }
        button { width: 100%; padding: 10px; background-color:
#007bff; color: #fff; border: none; border-radius: 5px; cursor:
pointer; }
        .result { margin-top: 20px; }
    </style>
</head>
<body>
    <div class="form-container">
        <h2>Celsius to Fahrenheit Converter</h2>
        <form method="POST" action="{{ url_for('converter') }}">
            <input type="text" name="celsius" placeholder="Enter
```



```
Celsius" required>
            <button type="submit">Convert</button>
        </form>
        {% if fahrenheit is not none %}
            <div class="result">
                {% if fahrenheit is string %}
                    <p>{{ fahrenheit }}</p>
                {% else %}
                    <p>{{ fahrenheit }} °F</p>
                {% endif %}
            </div>
        {% endif %}
    </div>
</body>
</html>
```