Experiment 5 : Flask Application using render_template() function.

Name of Student	**Eesha Chavan**
Class Roll No	**D15A 08**
D.O.P.	
D.O.S.	
Sign and Grade	

AIM :

To create a Flask application that demonstrates template rendering by dynamically generating HTML content using the `render_template()` function.

PROBLEM STATEMENT :

Develop a Flask application that includes:

1. A homepage route (`/`) displaying a welcome message with links to additional pages.
2. A dynamic route (`/user/<username>`) that renders an HTML template with a personalized greeting.
3. Use Jinja2 templating features, such as variables and control structures, to enhance the templates.

Theory:

1.What does the `render_template()` function do in a Flask application?

The `render_template()` function is used to render HTML templates stored in the **templates** folder. It dynamically generates web pages by passing variables from the Flask app to the template using Jinja2.

2. What is the significance of the `templates` folder in a Flask project?

- The **templates** folder is the default location where Flask looks for HTML files.
- It maintains a clean separation between business logic (Python code) and presentation logic (HTML).
- Using the templates folder allows developers to use Jinja2 for rendering dynamic content.
- The folder can also store reusable components like base templates, headers, or footers using **template inheritance**.

3. What is Jinja2, and how does it integrate with Flask?

- Jinja2 is a templating engine used in Flask to render dynamic HTML content. It allows embedding Python expressions inside HTML files.
- Using Jinja2, you can display variables, apply logic (like loops and conditionals), and apply filters for formatting.
- Flask integrates Jinja2 by default using `render_template()`

Codes:

app.py

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    features = [
        "Dynamic User Pages",
        "Templating with Jinja2",
        "CSS Styling"
    ]
    return render_template('home.html', features=features)

@app.route('/user/<username>')
def user_page(username):
    # Sample user data - in a real app, this might come from a database
    user_data = {
        'username': username,
        'visits': 5,
        'is_admin': username.lower() == 'admin'
    }
    return render_template('user.html', user=user_data)

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

templates/base.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Flask Demo{% endblock %}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
```

```
</head>
<body>
    <header>
        <nav>
            <a href="/">Home</a>
            <a href="/user/guest">Guest Page</a>
            <a href="/user/admin">Admin Page</a>
        </nav>
    </header>

    <main>
        {% block content %}{% endblock %}
    </main>

    <footer>
        <p>&copy; 2025 Flask Demo App</p>
    </footer>
</body>
</html>
```

templates/user.html:

```
{% extends "base.html" %}

{% block title %}{{ user.username }}'s Page{% endblock %}

{% block content %}
<div class="container">
    <h1>Hello, {{ user.username }}!</h1>

    <p>Welcome to your personalized page.</p>

    <p>You have visited this page {{ user.visits }} times.</p>

    {% if user.is_admin %}
        <div class="admin-panel">
            <h2>Admin Panel</h2>
            <p>As an admin, you have access to special features.</p>
            <ul>
                <li>Manage Users</li>
                <li>View Statistics</li>
```

```
                <li>System Settings</li>
            </ul>
        </div>
    {% else %}
        <p>This is your personal user area. Explore the site to discover
more!</p>
    {% endif %}
</div>
{% endblock %}
```

templates/home.html:

```
{% extends "base.html" %}

{% block title %}Welcome to Flask Demo{% endblock %}

{% block content %}
<div class="container">
    <h1>Welcome to Our Flask Demo</h1>

    <p>This is a simple Flask application demonstrating various
features:</p>

    <ul>
        {% for feature in features %}
            <li>{{ feature }}</li>
        {% endfor %}
    </ul>

    <div class="cta">
        <p>Check out the user pages:</p>
        <a href="/user/guest" class="button">Visit as Guest</a>
        <a href="/user/admin" class="button">Visit as Admin</a>
    </div>
</div>
{% endblock %}
```

static/style.css:

```
body {
    font-family: 'Arial', sans-serif;
    line-height: 1.6;
    margin: 0;
    padding: 0;
```

```
    background-color: #f4f4f4;
    color: #333;
}

header {
    background-color: #3498db;
    color: #fff;
    padding: 1rem;
}

nav {
    display: flex;
    gap: 1rem;
}

nav a {
    color: #fff;
    text-decoration: none;
}

nav a:hover {
    text-decoration: underline;
}

main {
    padding: 2rem;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background-color: #fff;
    padding: 2rem;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

footer {
    text-align: center;
    padding: 1rem;
    background-color: #333;
    color: #fff;
}

.button {
    display: inline-block;
    background-color: #3498db;
```

```
    color: #fff;
    padding: 0.5rem 1rem;
    text-decoration: none;
    border-radius: 3px;
    margin-right: 0.5rem;
}

.button:hover {
    background-color: #2980b9;
}

.cta {
    margin-top: 2rem;
    text-align: center;
}

.admin-panel {
    background-color: #f9f9f9;
    border-left: 4px solid #3498db;
    padding: 1rem;
    margin-top: 1rem;
}

ul {
    margin-bottom: 1.5rem;
}
```

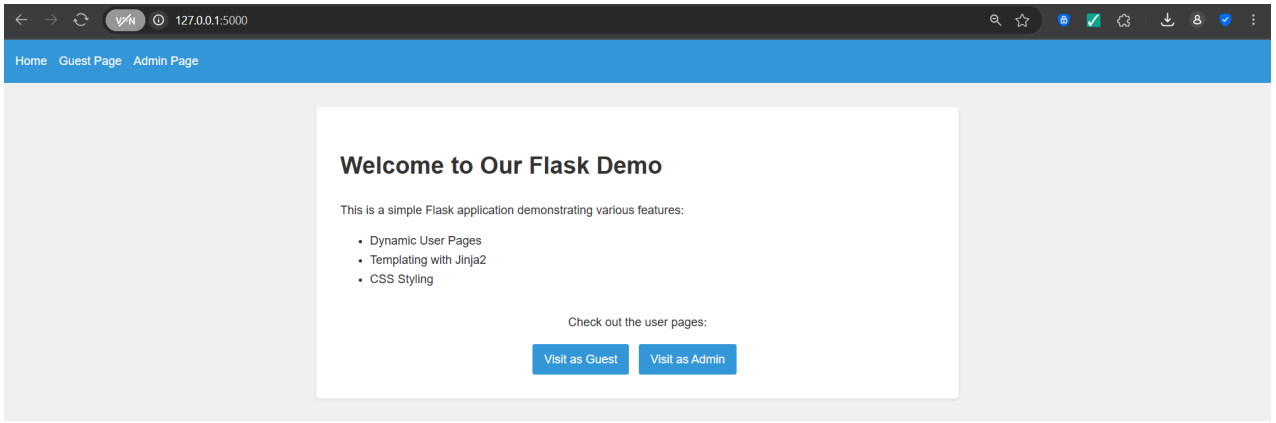
GitHub

Link:https://github.com/eeshachavan/WebX_Exp_5_eesha

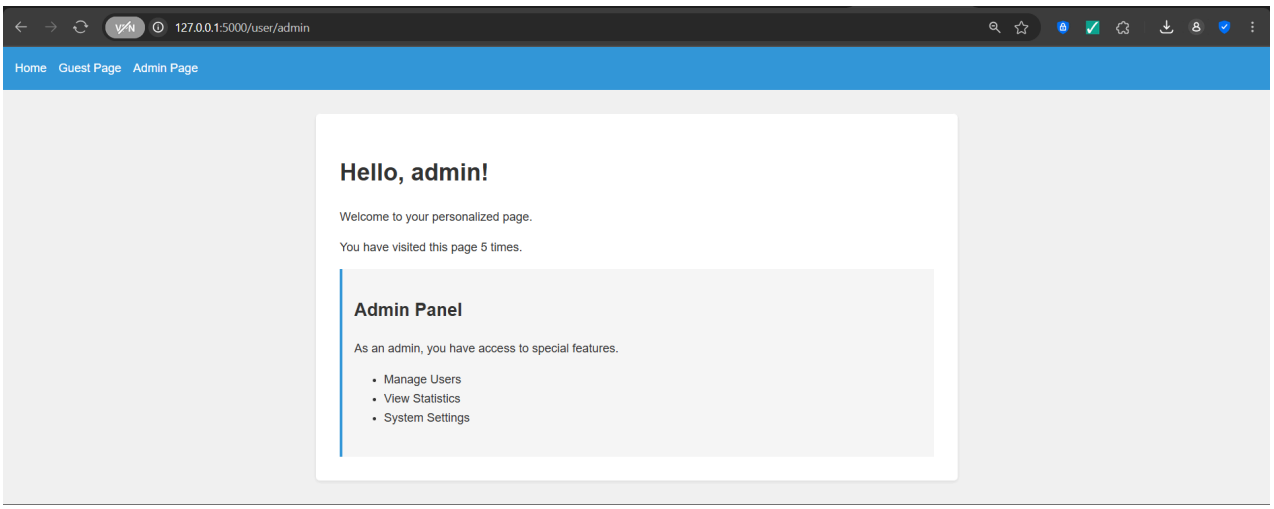
Output:

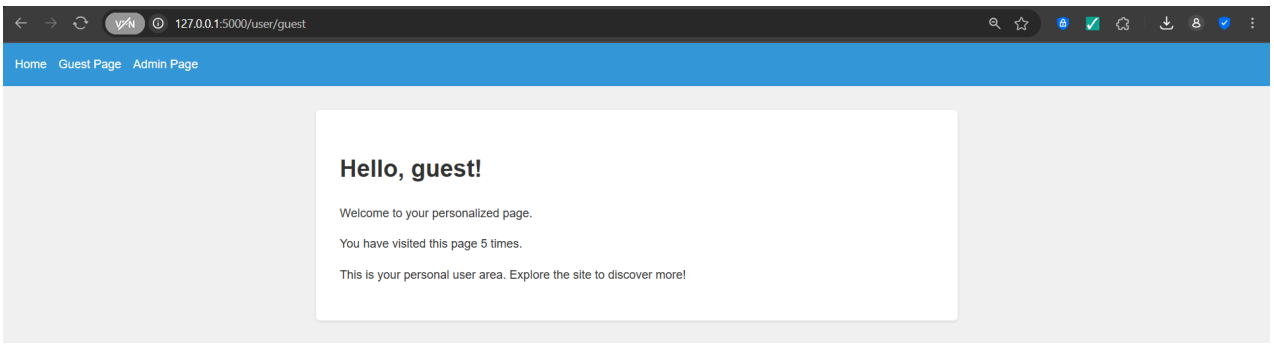
When you run the Flask application, the homepage (`/`) will display a welcoming message with links to dynamically generate user pages. Clicking on a user's link will take you to a personalized greeting page, as shown below:

- **Home Page:**
    - "Welcome to My Flask App" message.
    - Links for "Guest Page" and "Admin Page."



- **User Page:**
  Displays a personalized greeting such as "Hello, admin! 👋" based on the username passed in the URL.





Conclusion:

In this experiment, I successfully developed a Flask application demonstrating template rendering using the `render_template()` function. By creating dynamic routes and using Jinja2 templates, I displayed personalized user greetings based on URL parameters. The separation of business logic and presentation using HTML templates ensured clean and maintainable code. Additionally, I applied CSS for styling, enhancing the visual appeal of the application. This experiment helped me understand how to build interactive and user-friendly web applications using Flask.