Experiment 5 : Flask Application using render_template() function.

Name of Student	**Sneha Patra**
Class Roll No	**D15A_40**
D.O.P.	**06/03/2025**
D.O.S.	**13/03/2025**
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, request, redirect, url_for

app = Flask(__name__)

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

@app.route('/user', methods=['GET'])
def user():
    username = request.args.get('username', 'Guest')
    return render_template('user.html', username=username)

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

`templates/index.html:`

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Flask App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <div class="card">
            <h1>🌿 Welcome to My Flask App 🌿</h1>
            <p>Explore user pages dynamically with just one click!</p>
            <div class="links">
                <a href="{{ url_for('user', username='Sneha') }}" class="btn">Visit Sneha's Page</a>
                <a href="{{ url_for('user', username='User') }}" class="btn">Visit User's Page</a>
            </div>
        </div>
    </div>
</body>
```

```
</html>
```

templates/user.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Page</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <div class="card">
            <h1>👋 Hello, {{ username }}!</h1>
            <p>Welcome to your personalized space. Enjoy your time here!</p>
            <a href="{{ url_for('home') }}" class="btn">Back to Home</a>
        </div>
    </div>
</body>
</html>
```

static/style.css:

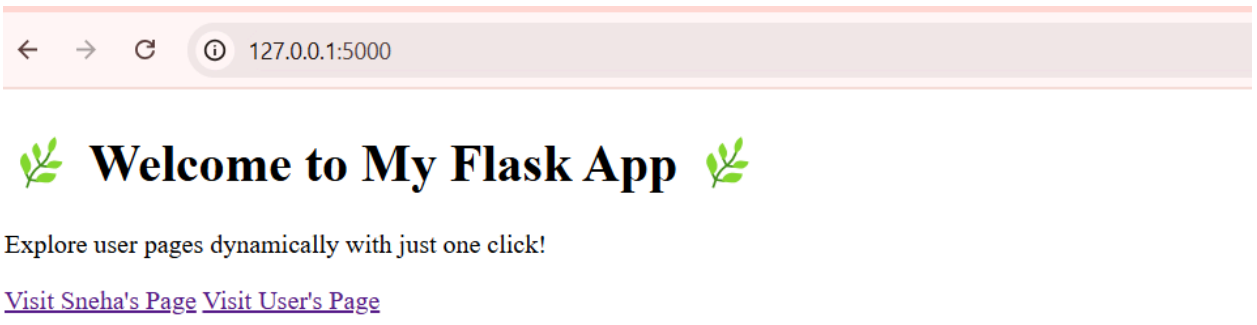
```
body {
    font-family: 'Segoe UI', sans-serif;
    background: linear-gradient(to right, #00c6ff, #0072ff);
    margin: 0;
    padding: 0;
    color: #fff;
}

.container {
    text-align: center;
    margin-top: 100px;
}

h1 {
    font-size: 2.5rem;
    color: #fff;
    margin-bottom: 10px;
}

p {
    font-size: 1.2rem;
    color: #e0e0e0;
}

.btn {
    background: #0072ff;
    color: white;
    padding: 12px 24px;
    text-decoration: none;
    border-radius: 25px;
    font-size: 1rem;
    transition: background 0.3s ease;
}

.btn:hover {
    background: #005bb5;
}

.card {
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    border-radius: 20px;
    padding: 40px;
    width: 90%;
    max-width: 500px;
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.2);
}

.links {
    display: flex;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}

/* Responsive Design */
@media (max-width: 600px) {
    .btn {
        padding: 10px 20px;
    }
}
```

GitHub Link: https://github.com/Sneha0321/WebX_Exp_5

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 "Sneha's Page" and "User's Page."



127.0.0.1:5000

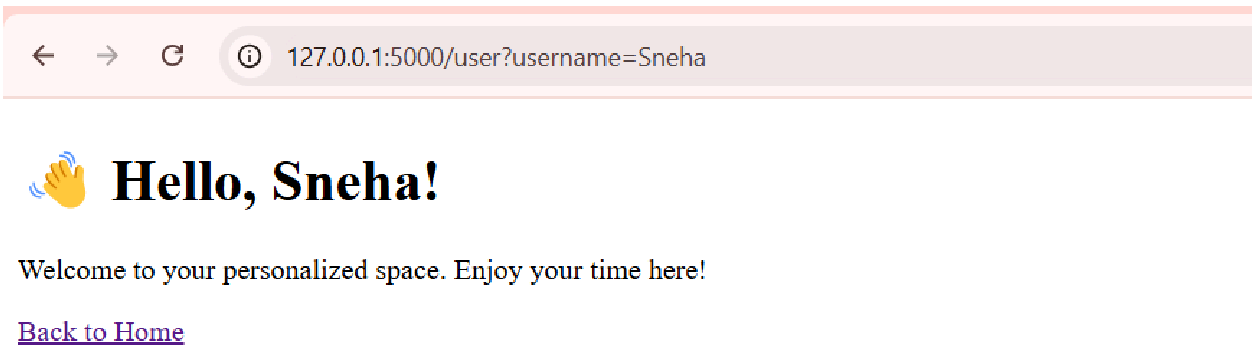
🌿 Welcome to My Flask App 🌿

Explore user pages dynamically with just one click!

Visit Sneha's Page Visit User's Page

- **User Page:**

  Displays a personalized greeting such as "Hello, Sneha! 👋" based on the username passed in the URL.

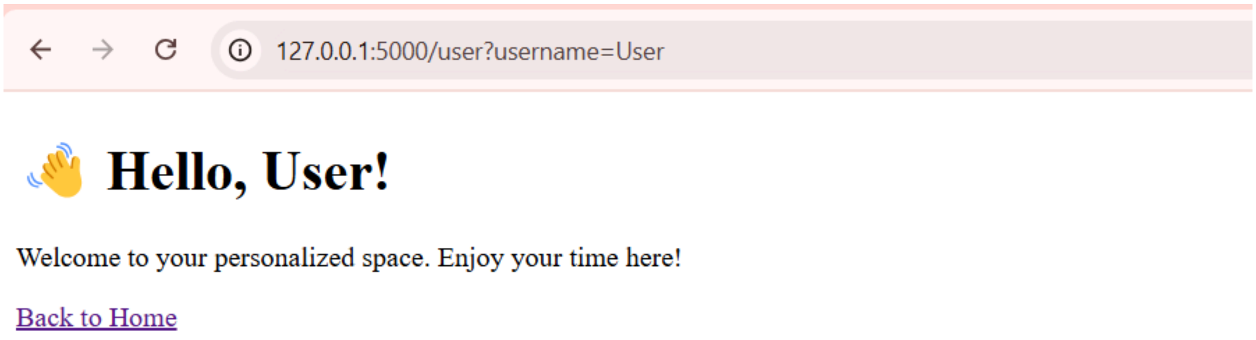


127.0.0.1:5000/user?username=Sneha

👋 Hello, Sneha!

Welcome to your personalized space. Enjoy your time here!

Back to Home


127.0.0.1:5000/user?username=User

👋 Hello, User!

Welcome to your personalized space. Enjoy your time here!

Back to Home

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.