# EXPERIMENT NO. 5

| | |
|---|---|
| **Name of Student** | Spandan Deb |
| **Class Roll No** | 13 |
| **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 in Flask is used to render an HTML file as a response to a client's request. It allows dynamic content to be passed from the backend to the frontend by utilizing template files stored in the templates folder.

Example:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html', title='Home Page')

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

In this example, index.html will be served, and the title variable can be used inside the template.

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

The templates folder is a convention in Flask that stores all HTML templates. Flask automatically looks for templates in this directory when rendering an HTML page using render_template(). This helps separate the frontend (HTML files) from the backend (Python logic), promoting a clean project structure.

Folder structure example:

```
/my_flask_app
│-- app.py
│-- templates/
│   │-- index.html
│   │-- about.html
```

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

Jinja2 is a templating engine used in Flask to create dynamic HTML content. It allows embedding Python-like expressions inside HTML templates using special syntax.

Key Features of Jinja2:

- Template Variables: Pass dynamic data from Flask to HTML.
- Control Structures: Supports loops and conditionals.
- Template Inheritance: Allows reusing layouts using base templates.

Example Usage in an HTML Template (index.html):

```
<!DOCTYPE html>
<html>
<head>
   <title>{{ title }}</title>
</head>
<body>
   <h1>Welcome, {{ user }}</h1>
   {% if user %}
      <p>Hello, {{ user }}! You are logged in.</p>
   {% else %}
      <p>Please log in.</p>
   {% endif %}
</body>
</html>
```

Here, {{ title }} and {{ user }} are template variables, and {% if user %} is a control structure.

Jinja2 makes Flask applications more dynamic by allowing backend data to be seamlessly integrated into frontend templates.

**CODE:**

App.py

```
from flask import Flask, render_template

app = Flask(__name__)

# Simple user data
users = {
   "Spandan": {"name": "Spandan Deb", "role": "Admin"},
   "guest": {"name": "Guest User", "role": "Guest"}
}

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

@app.route('/user/<username>')
def user_profile(username):
   # Get user info or use default if username not found
   user_info = users.get(username, {"name": username, "role": "Visitor"})
```

```
    return render_template('users.html', username=username, user_info=user_info)

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

Templates

Home.html

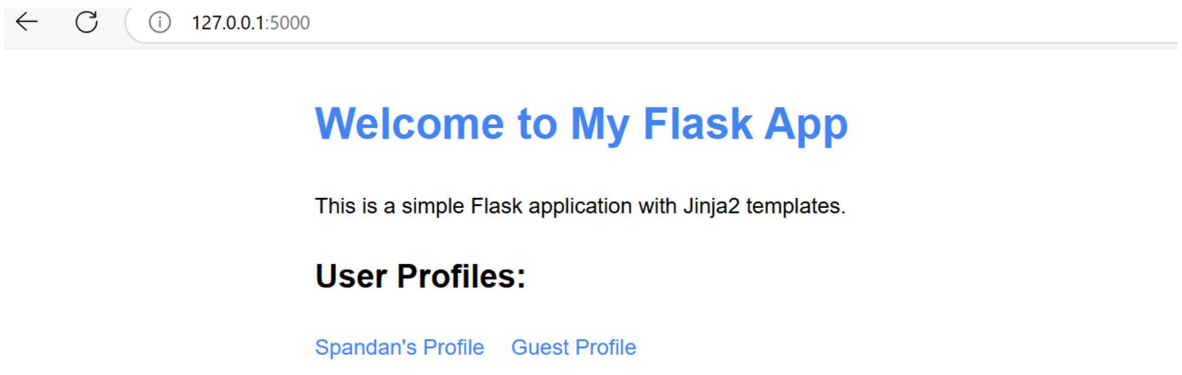
```
<!DOCTYPE html>
<html>
<head>
    <title>Flask App - Home</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 30px;
            line-height: 1.6;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
        h1 {
            color: #4285f4;
        }
        a {
            color: #4285f4;
            text-decoration: none;
            margin-right: 15px;
        }
        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to My Flask App</h1>
        <p>This is a simple Flask application with Jinja2 templates.</p>

        <h2>User Profiles:</h2>
        <p>
```

```
            <a href="{{ url_for('user_profile', username='spandan') }}">Spandan's
Profile</a>
            <a href="{{ url_for('user_profile', username='guest') }}">Guest Profile</a>
        </p>
    </div>
</body>
</html>
```

Users.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask App - Home</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 30px;
            line-height: 1.6;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
        h1 {
            color: #4285f4;
        }
        a {
            color: #4285f4;
            text-decoration: none;
            margin-right: 15px;
        }
        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to My Flask App</h1>
        <p>This is a simple Flask application with Jinja2 templates.</p>
        <h2>User Profiles:</h2>
        <p>
            <a href="{{ url_for('user_profile', username='spandan') }}">Spandan's
Profile</a>
            <a href="{{ url_for('user_profile', username='guest') }}">Guest Profile</a>
        </p>
    </div>
</body>
</html>
```

**OUTPUT:**



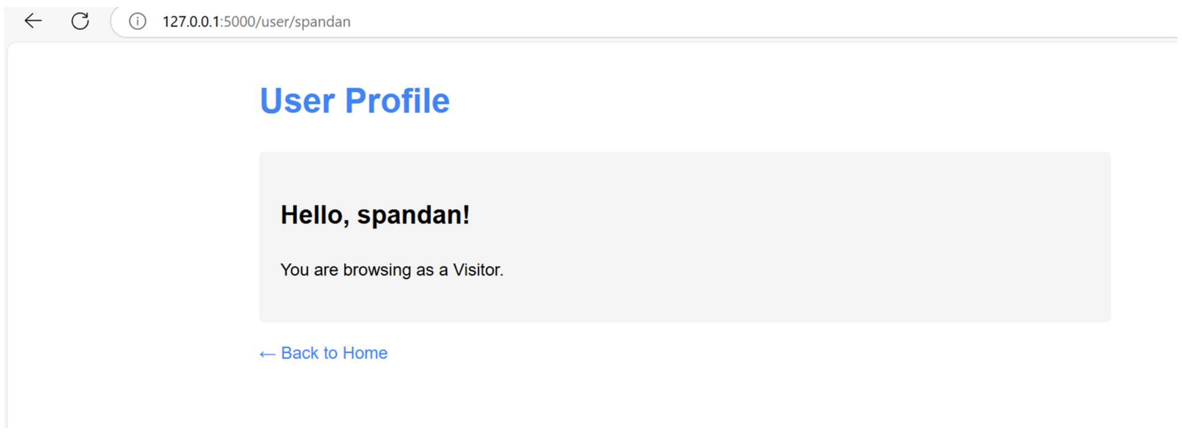
127.0.0.1:5000

Welcome to My Flask App

This is a simple Flask application with Jinja2 templates.

User Profiles:

Spandan's Profile    Guest Profile



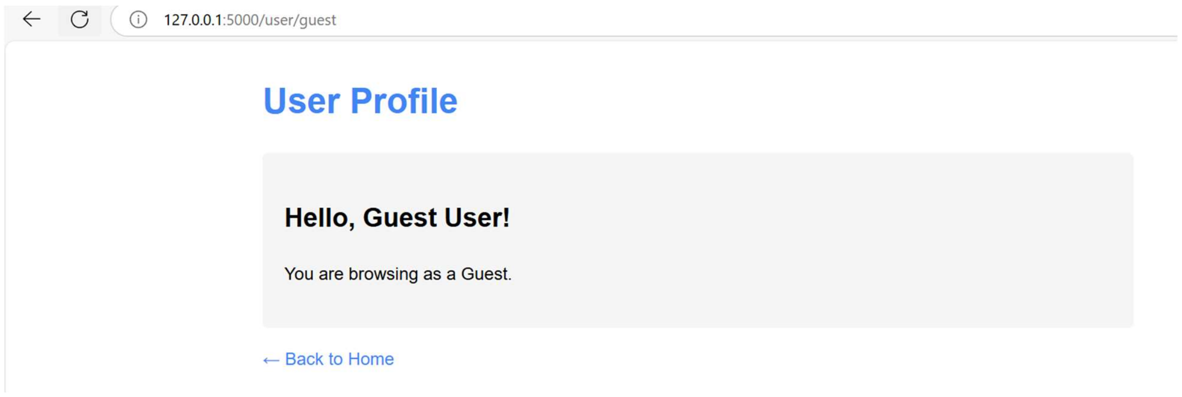
127.0.0.1:5000/user/spandan

User Profile

Hello, spandan!

You are browsing as a Visitor.

← Back to Home



User Profile

Hello, Guest User!

You are browsing as a Guest.

← Back to Home

## CONCLUSION:

This experiment successfully demonstrated the use of template rendering in Flask. By utilizing the render_template () function, dynamically generated HTML content and integrated backend data with frontend templates using Jinja2. This approach improves code organization, enhances reusability, and enables the development of interactive web applications