EXPERIMENT NO. 5

Name of Student	Arnav Santosh Sawant
Class Roll No	D15A - 52
D.O.P.	25/02/25
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?

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

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

Answers:-

1. In a Flask application, the `render_template()` function is used to render HTML templates and return them as a response to the client (usually a web browser). It allows you to separate your HTML from your Python code, making your application easier to maintain and scale. The `render_template()` function takes the name of an HTML template file (located in the `templates` folder) and renders it, potentially injecting dynamic content such as variables or data passed to it.

For example:
```
from flask import render_template
@app.route('/')
def home():
	return render_template('index.html', username="JohnDoe")
```

In the above code, the `render_template()` function is rendering the `index.html` template, and it's passing the variable `username` to the template, which can then be accessed within the HTML file using Jinja2 syntax (`{{ username }}`).

2. In a Flask project, the `templates` folder holds all of your HTML files (templates) that define the structure and presentation of your web pages. When you use `render_template()`, Flask looks for the specified HTML files inside the `templates` directory.
   This folder is crucial because Flask uses it to separate the business logic (Python code) from the presentation layer (HTML files). This separation adheres to the Model-View-Controller (MVC) design pattern and helps in organizing code in a more maintainable and scalable way.
   Flask automatically searches for HTML files in the `templates` folder when you call `render_template()`, so it is important to organize your HTML files there. If the folder is not named `templates`, you would need to specify the path manually.

3. Jinja2 is a modern and designer-friendly templating engine for Python. It allows you to write dynamic HTML templates by embedding Python-like expressions and control structures within HTML. Jinja2 provides powerful features like template inheritance, loops, conditionals, filters, and more.
   Flask integrates with Jinja2 to render templates. When you call `render_template()` in Flask, it automatically uses Jinja2 to process the template. This means you can pass variables, execute loops, use conditionals, and apply filters directly in your HTML files, which are then rendered to the client as regular HTML.

   Some key features of Jinja2 include:

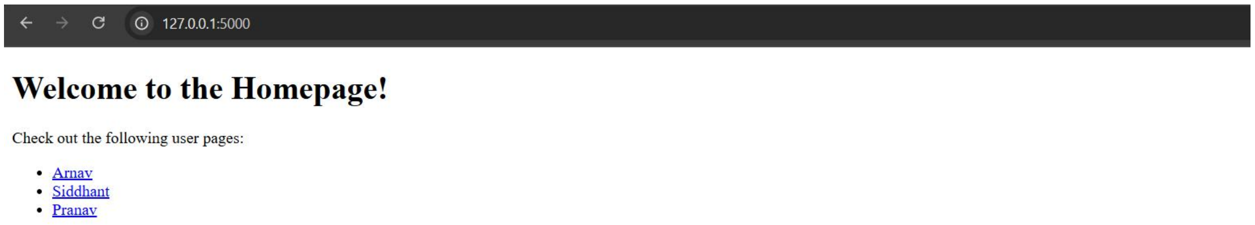
   - **Template Variables**: You can pass variables from Python to HTML (e.g., `{{ username }}`).
   - **Control Structures**: Jinja2 supports loops (`{% for %}`) and conditionals (`{% if %}`).

- **Template Inheritance**: You can create a base template and extend it in child templates using `{% extends %}` and `{% block %}` tags.
- **Filters**: Jinja2 allows you to use filters to modify output, such as `{{ name|capitalize }}`.

This is the code:-

```
<h1>Hello, {{ username }}!</h1>

{% if user_is_logged_in %}

    <p>Welcome back!</p>

{% else %}

    <p>Please log in to continue.</p>

{% endif %}
```

Screenshots:-

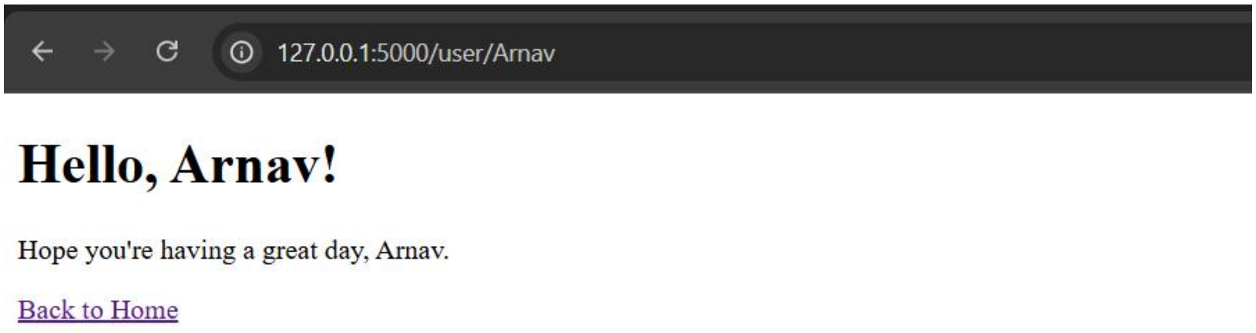


127.0.0.1:5000

Welcome to the Homepage!

Check out the following user pages:

- Arnav
- Siddhant
- Pranav



127.0.0.1:5000/user/Arnav

Hello, Arnav!

Hope you're having a great day, Arnav.

Back to Home

app.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")

def home():

    pages = ["Arnav", "Siddhant", "Pranav"]

    return render_template("home.html", pages=pages)


@app.route("/user/<username>")

def user(username):

    return render_template("user.html", username=username)


if __name__ == "__main__":

    app.run(debug=True)
```

templates/home.html

```
<!DOCTYPE html>

<html>

<head>

    <title>Home</title>

</head>

<body>

    <h1>Welcome to the Homepage!</h1>

    <p>Check out the following user pages:</p>
    <ul>
        {% for user in pages %}
            <li><a href="{{ url_for('user', username=user) }}">{{ user }}</a></li>
        {% endfor %}
    </ul>
</body>
</html>
```

templates/user.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello {{ username }}</title>
</head>
<body>
    <h1>Hello, {{ username }}!</h1>

    {% if username.lower() == 'admin' %}
        <p>Welcome back, mighty administrator!</p>
    {% else %}
        <p>Hope you're having a great day, {{ username }}.</p>
    {% endif %}

    <a href="{{ url_for('home') }}">Back to Home</a>
```

</body>

</html>