EXPERIMENT NO. 5 : Flask

Name of Student	**Komal Milind Deolekar**
Class Roll No	**14**
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.

Github Link :

https://github.com/KomalDeolekar0607/Webx_Lab/tree/main/Webx_Lab_Exp_5

Theory :

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

The render_template() function in Flask is used to render HTML templates and return them as a response to the client. It allows developers to separate business logic (Python code) from the presentation layer (HTML).

Usage:

- It loads an HTML file from the templates folder.
- It can pass dynamic data (variables) from Flask to the HTML template.
- It integrates with Jinja2 for template rendering.

Example:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html', title="Home Page", message="Welcome to Flask!")

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

How It Works?

- render_template('index.html') loads the index.html file from the templates folder.
- title="Home Page" and message="Welcome to Flask!" are passed to the template.
- These variables can be used in index.html with Jinja2 syntax.

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

In Flask, the templates folder is the default location where HTML files are stored. Flask automatically looks for HTML templates inside this folder when using render_template().

Key Points:

- It helps **organize** the project structure.
- It supports **template inheritance** (reusing common HTML structures).
- It integrates **Jinja2 templating** for dynamic content.

Project Structure Example:

```
/FlaskApp
├── app.py              # Main Flask application
├── /templates
│   ├── index.html      # Homepage template
│   ├── about.html      # About page template
│   └── base.html       # Common layout for template inheritance
```

Example of Template Usage:

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

Flask will automatically look for about.html inside the templates folder.

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

Jinja2 is Flask's built-in templating engine that allows the use of dynamic content, loops, conditions, and template inheritance inside HTML files.

Features of Jinja2:

Supports **template variables** ({{ variable }})
Allows **control structures** ({% if %}, {% for %})
Enables **template inheritance** (using {% extends %} and {% block %})

Example 1: Using Variables in Jinja2

Python (Flask):

```
@app.route('/')
def home():
   return render_template('index.html', user="Komal")
```

index.html (Jinja2 Template):

```
<!DOCTYPE html>
<html>
<head><title>Flask App</title></head>
<body>
   <h1>Welcome, {{ user }}!</h1>  <!-- Outputs: Welcome, Komal! -->
</body>
</html>
```

Example 2: Using Loops in Jinja2

```
@app.route('/users')
def users():
   user_list = ["Alice", "Bob", "Charlie"]
   return render_template('users.html', users=user_list)
```

users.html:

```
<ul>
   {% for user in users %}
      <li>{{ user }}</li>
   {% endfor %}
</ul>
```

This dynamically generates a list of users.

How Jinja2 Integrates with Flask:

- Flask **automatically processes** Jinja2 syntax in render_template().
- It allows **passing Python variables** into HTML templates.
- It supports **complex logic** (loops, conditionals) inside HTML files.

Jinja2 makes Flask templates **dynamic and flexible**, allowing developers to create **reusable and structured** web applications.

Code:

app.py

```
from flask import Flask, render_template , url_for,redirect,request

app = Flask(__name__)

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

@app.route('/user/<username>')
def user(username):
   return render_template("user.html", username=username)

# @app.route('/user_redirect')
# def user_redirect():
#     username = request.args.get('name' , 'Guest')
#     return redirect(url_for('user', username = username))
```

```
@app.route('/user_redirect', methods=['GET', 'POST'])
def user_redirect():
    if request.method == 'POST':
        username = request.form.get('name', 'Guest')  # Use request.form for POST
    else:
        username = request.args.get('name', 'Guest')  # Use request.args for GET

    return redirect(url_for('user', username=("Guest" if username=="" else
username[0].upper()+username[1:])))


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

templates/home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Flask Home</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Welcome to the Flask Application!</h1>
        <p>This app demonstrates dynamic template rendering using Jinja2.</p>
        <a href="{{ url_for('user', username='Komal') }}" class="btn">Visit Komal's Page</a>
        <a href="{{ url_for('user', username='Guest') }}" class="btn">Visit Guest Page</a>
        <p>Or Else</p>
        <form action="{{url_for('user_redirect')}}" method="get">
            Enter Your Name :
            <input type="text" name="name">
            <button type="submit" class="btn">Visit Your Page</button>
        </form>


    </div>
</body>
</html>
```

templates/user.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>User Page</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
   <div class="container">
      <h1>Hello, {{ username }}! </h1>
      <p>Welcome to your personalized page.</p>

      {% if username.lower() == "komal" %}
         <p>Hey Komal! You are the owner of this application. </p>
      {% elif username|length <= 5 %}
         <p>Feel free to explore the site, {{ username }}!</p>
      {% else %}
         <p>Hope you enjoy our site, {{ username }}!</p>
      {% endif %}

      <a href="{{ url_for('home') }}" class="btn">Back to Home</a>
   </div>
</body>
</html>
```

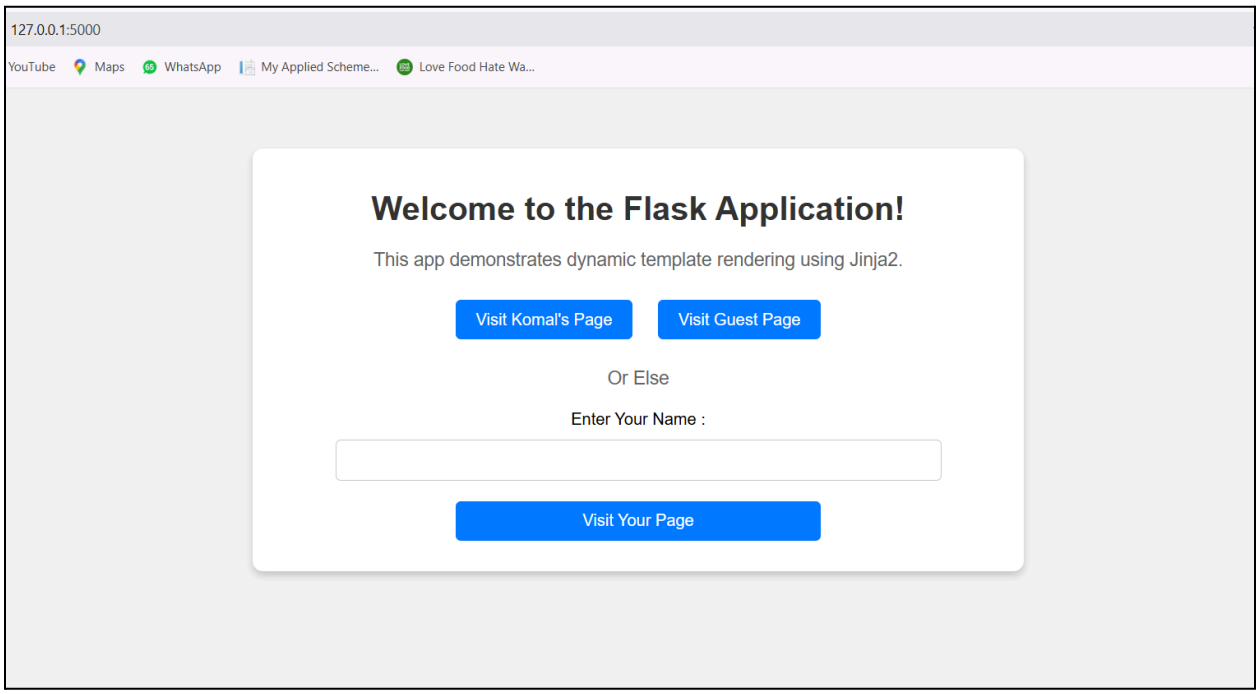
static/styles.css

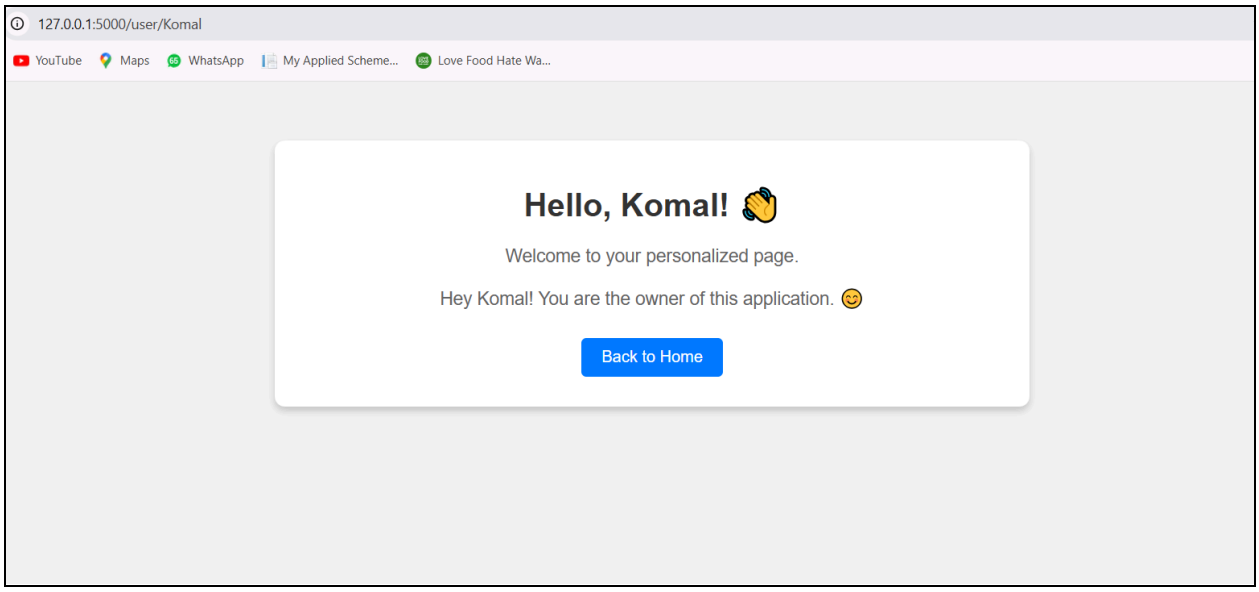
```
body {
   font-family: Arial, sans-serif;
   background: #f4f4f4;
   text-align: center;
   padding: 50px;
}

.container {
   background: white;
   padding: 20px;
   border-radius: 10px;
   box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);
   display: inline-block;
   width: 50%;
}
```

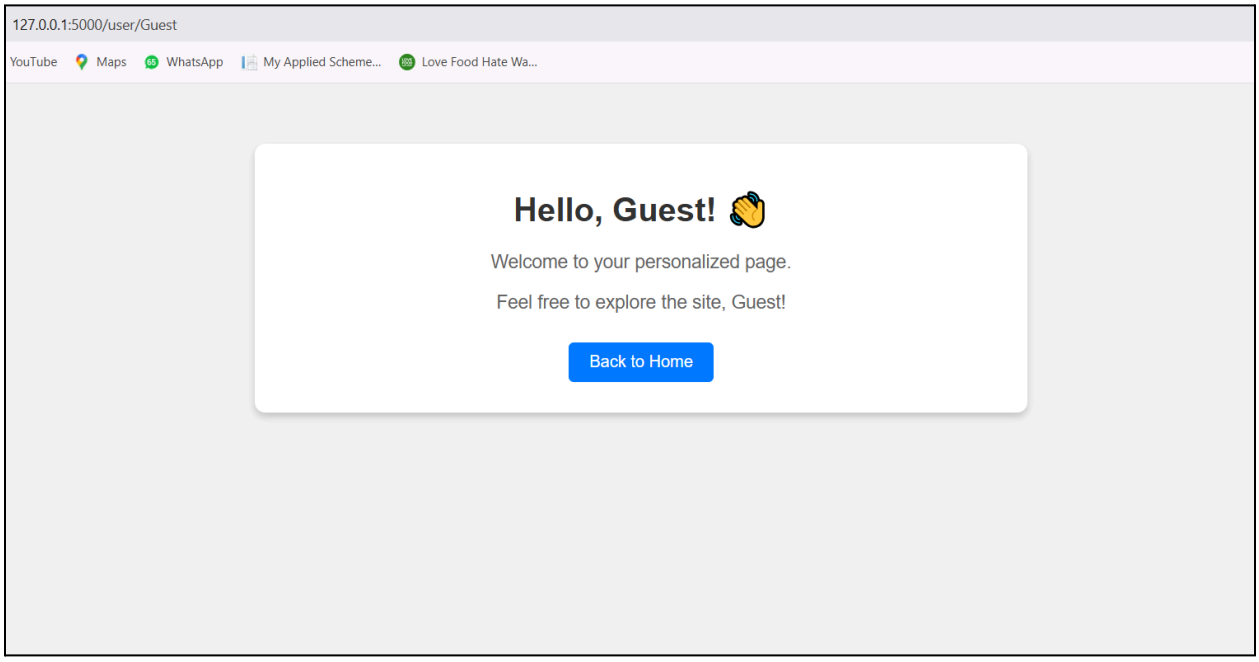
```
h1 {
    color: #333;
}

p {
    font-size: 18px;
    color: #666;
}

.btn {
    display: inline-block;
    padding: 10px 20px;
    margin: 10px;
    text-decoration: none;
    color: white;
    background: #007BFF;
    border-radius: 5px;
    border: none;
    cursor: pointer;
    font-size: 16px;
}

.btn:hover {
    background: #0056b3;
}

/* Styling for the Form */
form {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

input[type="text"] {
    width: 80%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}

button {
```

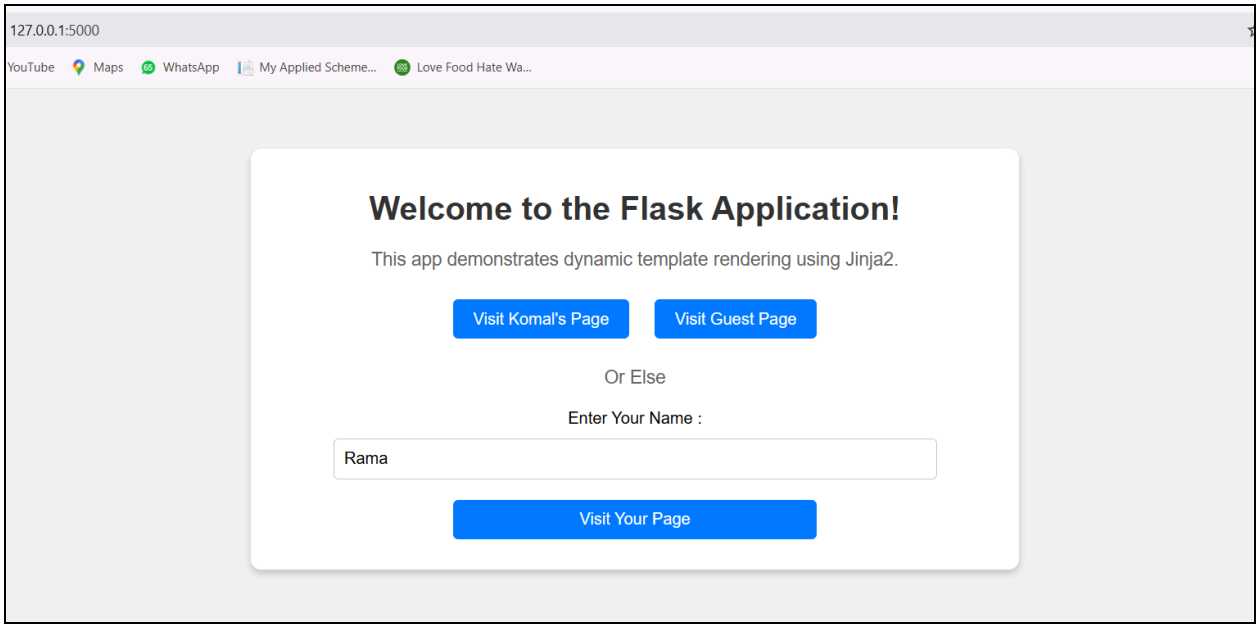
```
    width: 50%;
    padding: 10px;
    font-size: 16px;
    background: #28a745;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background: #218838;
}
```

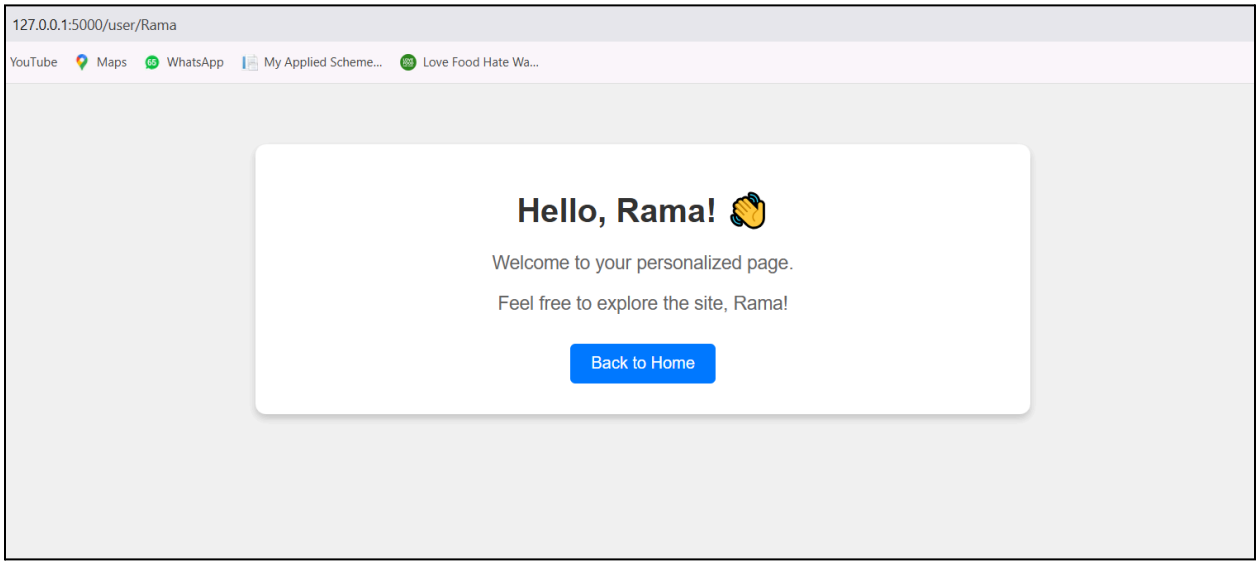
Output :

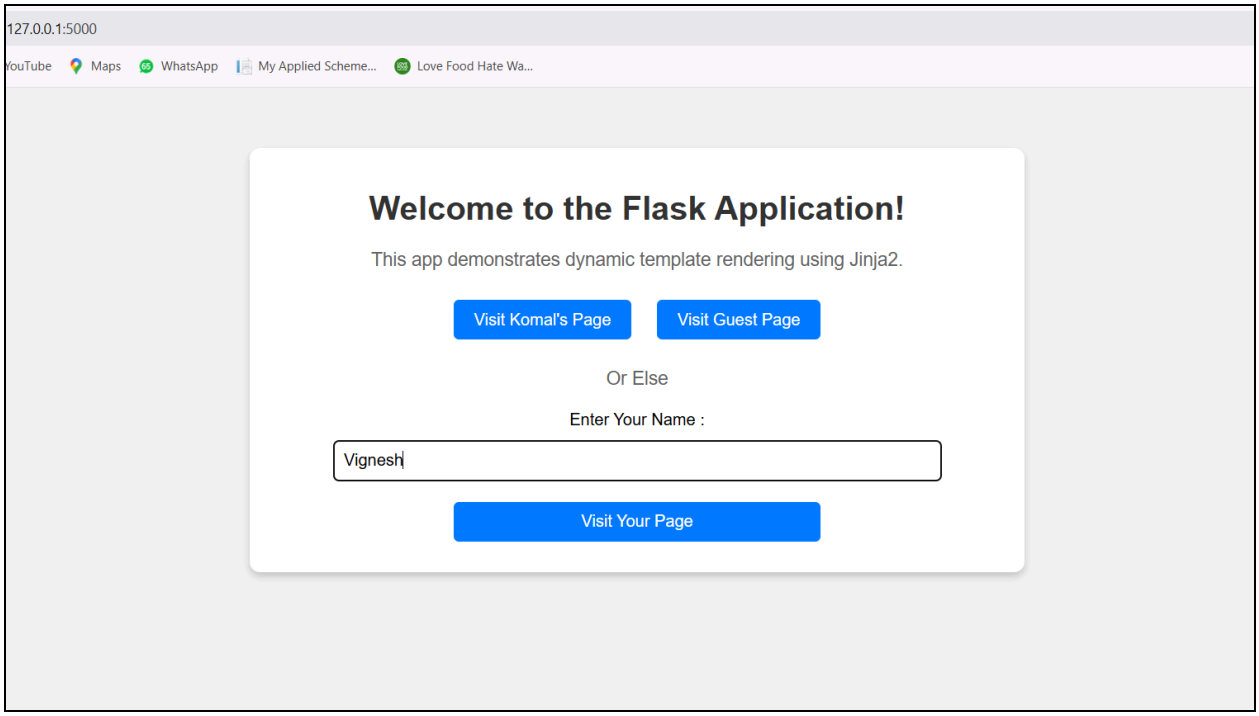


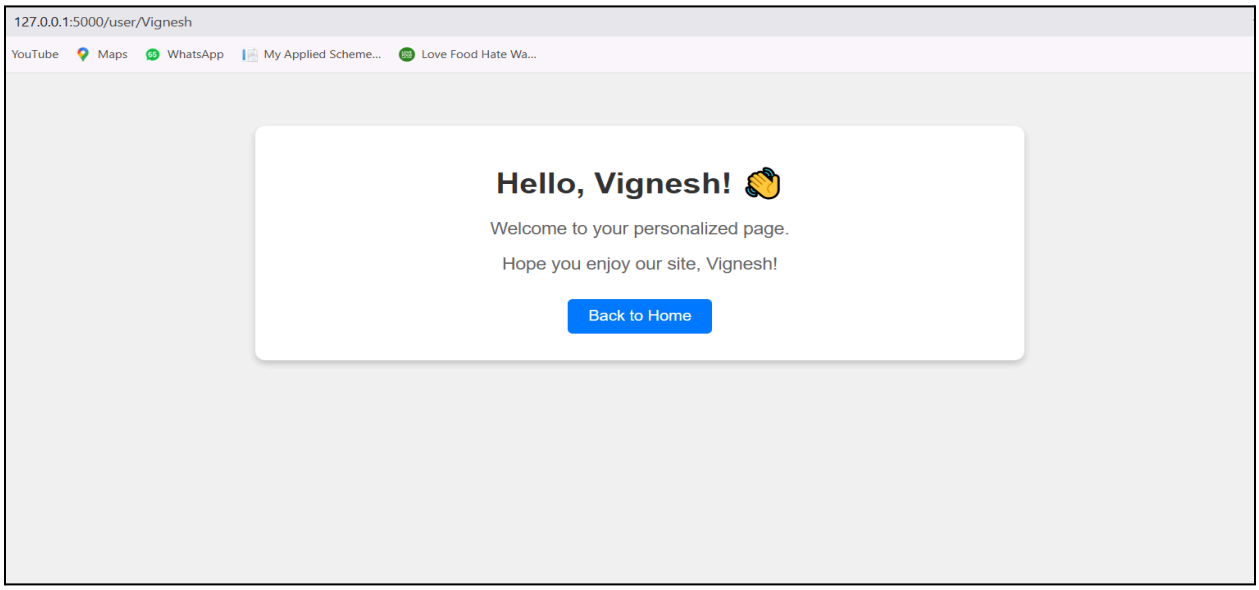

127.0.0.1:5000/user/Komal
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Hello, Komal! 👋
Welcome to your personalized page.
Hey Komal! You are the owner of this application. 😊
Back to Home


127.0.0.1:5000/user/Guest
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Hello, Guest! 👋
Welcome to your personalized page.
Feel free to explore the site, Guest!
Back to Home


127.0.0.1:5000
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Welcome to the Flask Application!
This app demonstrates dynamic template rendering using Jinja2.
Visit Komal's Page
Visit Guest Page
Or Else
Enter Your Name :
Rama
Visit Your Page


127.0.0.1:5000/user/Rama
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Hello, Rama!
Welcome to your personalized page.
Feel free to explore the site, Rama!
Back to Home





Conclusion :

In this practical, we developed a Flask web application that effectively demonstrates template rendering by dynamically generating HTML content using the render_template() function. The application included a homepage route that displayed a welcome message along with links to other pages. Additionally, a dynamic route /user/<username> was implemented, which rendered a personalized greeting using the username passed in the URL. We utilized Jinja2 templating features such as variables and control structures to enhance the HTML templates, making the content more interactive and user-specific. This practical helped in understanding how Flask separates the logic of application and presentation, making web development more organized and efficient.