

EXPERIMENT NO. 3

Name of Student	Ria Chaudhari
Class Roll No	D15A07
DOP	20/02/2025
DOS	27/02/2025
Sign and Grade	

AIM : To develop a basic Flask application with multiple routes and demonstrate the handling of GET and POST requests.

PROBLEM STATEMENT :

Design a Flask web application with the following features:

1. A homepage (/) that provides a welcome message and a link to a contact form.
 - a. Create routes for the homepage (/), contact form (/contact), and thank-you page (/thank_you).
2. A contact page (/contact) where users can fill out a form with their name and email.
3. Handle the form submission using the POST method and display the submitted data on a thank-you page (/thank_you).
 - a. On the contact page, create a form to accept user details (name and email).
 - b. Use the POST method to handle form submission and pass data to the thank-you page
4. Demonstrate the use of GET requests by showing a dynamic welcome message on the homepage when the user accesses it with a query parameter, e.g., /welcome?name=<user_name>.
 - a. On the homepage (/), use a query parameter (name) to display a personalized welcome message.

Theory:

A. List some of the core features of Flask

Flask is a lightweight and flexible web framework for Python. Some of its core features include:

- **Minimal and Lightweight** – Flask is a microframework with no built-in ORM or authentication system.
- **Built-in Development Server & Debugger** – Helps in testing applications easily.
- **RESTful Request Handling** – Supports multiple HTTP methods (GET, POST, etc.).

- **Jinja2 Templating Engine** – Supports dynamic HTML generation using templates.
- **URL Routing** – Provides a simple way to define application URLs.
- **Session and Cookie Management** – Supports user sessions and cookies.
- **Extensibility** – Can be extended with Flask extensions for database handling, authentication, etc.
- **WSGI Support** – Built on the WSGI standard for web applications.

B. Why do we use Flask(__name__) in Flask?

When initializing a Flask application, we write:

```
app = Flask(__name__)
```

- `__name__` represents the name of the current module.
- Flask uses this to determine the root path of the application.
- It helps locate resources like templates and static files.
- It also helps in debugging and logging.

C. What is Template (Template Inheritance) in Flask?

Flask uses **Jinja2**, a templating engine, to separate logic from presentation.

- A **template** is an HTML file with placeholders for dynamic content.
- **Template Inheritance** allows reusing code by defining a base template and extending it in child templates.

Example:

Base Template (base.html)

```
<!DOCTYPE html>

<html>

<head>

    <title>{% block title %}Default Title{% endblock %}</title>

</head>

<body>

    <header>Flask Application</header>

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

</body>
```

</html>

Child Template (home.html)

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

```
{% block title %}Home Page{% endblock %}
```

```
{% block content %}
```

```
    <h1>Welcome to Flask</h1>
```

```
{% endblock %}
```

D. What methods of HTTP are implemented in Flask.

Flask supports various HTTP methods:

- A. **GET** – Requests data from a server (default method).
- B. **POST** – Sends data to the server (used in forms).
- C. **PUT** – Updates an existing resource.
- D. **DELETE** – Deletes a resource.
- E. **PATCH** – Partially updates a resource.
- F. **HEAD** – Similar to GET but retrieves only headers, not content.
- G. **OPTIONS** – Returns HTTP methods supported by the server.

Example:

```
@app.route('/submit', methods=['GET', 'POST'])
```

```
def submit():
```

```
    if request.method == 'POST':
```

```
        return "Data Submitted!"
```

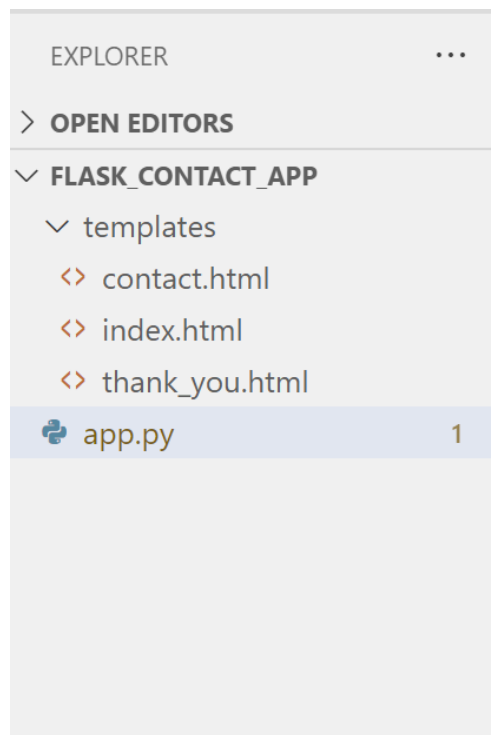
```
    return "Submit Form"
```

E.What is difference between Flask and Django framework

Feature	Flask	Django
Type	Lightweight, micro-framework	Full-stack, "batteries-included" framework
Flexibility	Highly flexible	Less flexible
Best For	Small to medium-sized projects	Larger, more complex projects
Database Integration	No built-in ORM	Built-in ORM
Community	Growing community	Larger community

Code:

Folder structure:



app.py

```
app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def homepage():
7      name = request.args.get('name', 'Guest')
8      return render_template('index.html', name=name)
9
10 @app.route('/contact', methods=['GET', 'POST'])
11 def contact():
12     if request.method == 'POST':
13         name = request.form.get('name')
14         email = request.form.get('email')
15         return redirect(url_for('thank_you', name=name, email=email))
16     return render_template('contact.html')
17
18 @app.route('/thank_you')
19 def thank_you():
20     name = request.args.get('name', 'Unknown')
21     email = request.args.get('email', 'Unknown')
22     return render_template('thank_you.html', name=name, email=email)
23
24 if __name__ == '__main__':
25     app.run(debug=True)
```

index.html



```
templates > <> index.html > html > body > h1
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Home</title>
7  </head>
8  <body>
9      <h1>Welcome to the homepage!</h1>
10     <p><a href="/contact">Go to Contact Form</a></p>
11 </body>
12 </html>
13
```

contact.html

templates > <> contact.html > ...

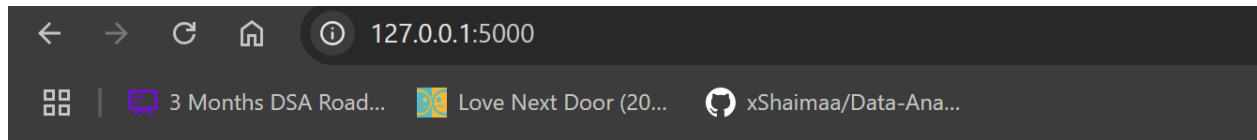
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Contact</title>
7  </head>
8  <body>
9      <h1>Contact Form</h1>
10     <form method="post">
11         <label for="name">Name:</label>
12         <input type="text" name="name" required><br>
13         <label for="email">Email:</label>
14         <input type="email" name="email" required><br>
15         <input type="submit" value="Submit">
16     </form>
17 </body>
18 </html>
19
```

thank_you.html

templates > <> thank_you.html >  html >  body

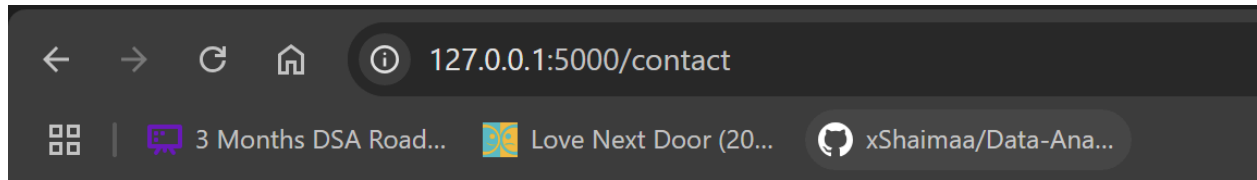
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Thank You</title>
7  </head>
8  <body>
9      <h1>Thank You!</h1>
10     <p>Name: {{ name }}</p>
11     <p>Email: {{ email }}</p>
12     <p><a href="/">Back to Home</a></p>
13 </body>
14 </html>
15
```

OUTPUT



Welcome to the homepage!

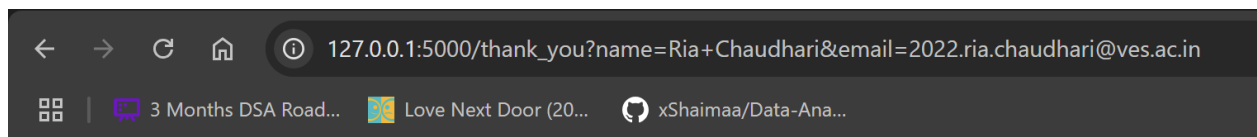
[Go to Contact Form](#)



Contact Form

Name:

Email:



Thank You!

Name: Ria Chaudhari

Email: 2022.ria.chaudhari@ves.ac.in

[Back to Home](#)