EXPERIMENT NO. 3

Name of Student	**RONAK KATARIYA**
Class Roll No	**D15A 23**
D.O.P.	
D.O.S.	
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:

1. Core Features of Flask

- Lightweight and minimal framework
- Built-in development server and debugger
- RESTful request handling

- URL routing
- Jinja2 templating engine
- Support for secure cookies (session management)
- Extensible with Flask extensions (e.g., Flask-SQLAlchemy, Flask-WTF)
- WSGI compliance

2. Why do we use `Flask(__name__)` in Flask?

- `Flask(__name__)` initializes a Flask application.
- `__name__` helps Flask determine the root path of the application for locating resources like templates and static files.
- It enables Flask to define routes relative to the application's directory.

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

- Flask uses Jinja2 as its templating engine.
- Template inheritance allows reusing base templates by extending them.

Example:

```
 <!-- base.html -->
<html>
<body>
   <h1>Flask App</h1>
   {% block content %}{% endblock %}
```

```
</body>

</html>

 <!-- child.html -->

{% extends "base.html" %}

{% block content %}

   <p>Welcome to the Contact Page</p>

{% endblock %}
```

4. HTTP Methods Implemented in Flask

- `GET`: Retrieves data (e.g., fetching a webpage)
- `POST`: Sends data (e.g., submitting a form)
- `PUT`: Updates existing resources
- `DELETE`: Removes resources

5. Difference between Flask and Django

Feature	Flask	Django
Type	Micro-framework	Full-stack framework
Flexibility	Highly flexible	More structured
Database Support	No built-in ORM (uses extensions)	Built-in ORM

Use Case	Small apps, APIs, microservices	Large web applications
Learning Curve	Easier	Steeper

6. Routing

- Routing maps URLs to specific view functions.

Example:

```
@app.route('/')
def home():
    return "Welcome to Flask!"
```

7. URL Building

- `url_for()` helps generate dynamic URLs.

Example:

```
url_for('profile', username='John')
```

8. GET Request

- Used to retrieve data.

Example:

```
@app.route('/user')
def get_user():
```

```
    name = request.args.get('name')
    return f"Hello, {name}"
```

-

9. POST Request

- Used to send data.

Example:

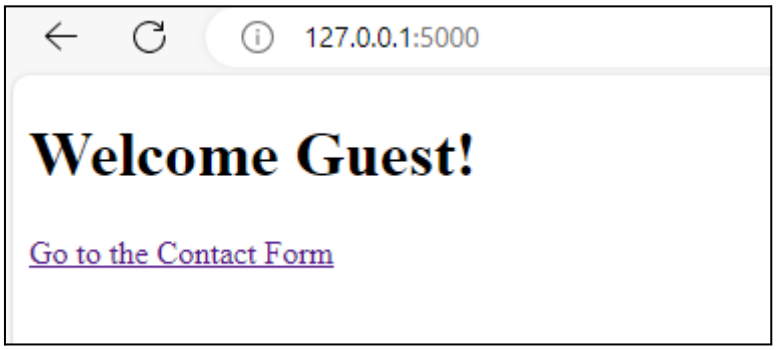
```
 @app.route('/submit', methods=['POST'])
def submit():
    name = request.form['name']
    return f"Submitted: {name}"
```

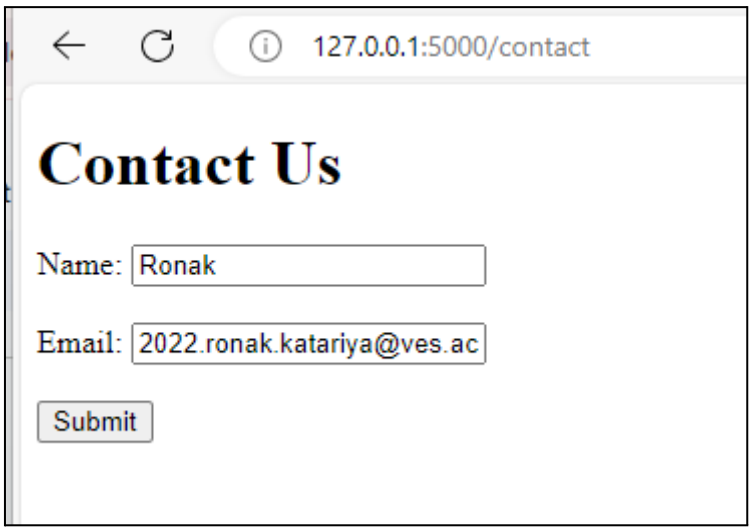
Code:

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

app = Flask(__name__)

@app.route('/')
def home():
    name = request.args.get('name', 'Guest')
    return f'''<h1>Welcome, {name}!</h1>
            <p><a href="/contact">Go to Contact Page</a></p>'''

@app.route('/contact', methods=['GET', 'POST'])
def contact():
    if request.method == 'POST':
```

```
        name = request.form['name']
        email = request.form['email']
        return redirect(url_for('thank_you', name=name, email=email))
    return '''<form method="post">
                Name: <input type="text" name="name" required><br>
                Email: <input type="email" name="email" required><br>
                <input type="submit" value="Submit">
            </form>'''

@app.route('/thank_you')
def thank_you():
    name = request.args.get('name')
    email = request.args.get('email')
    return f'<h1>Thank You!</h1><p>Name: {name}</p><p>Email: {email}</p>'

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

OUTPUT




127.0.0.1:5000/contact
Contact Us
Name: Ronak
Email: 2022.ronak.katariya@ves.ac
Submit


127.0.0.1:5000/thank_you?name=Ronak&email=2022.ronak.katariya@ves.ac.in
Thank You for Contacting Us!
We have received the following information:
Name: Ronak
Email: 2022.ronak.katariya@ves.ac.in
Go back to the homepage