Experiment – 3

|  |  |
| --- | --- |
| Name of Student | Madhura Jangale |
| Class Roll No | D15A\_20 |
| 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.
   1. 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).
   1. On the contact page, create a form to accept user details (name and email).
   2. 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>.

* 1. On the homepage (/), use a query parameter (name) to display a personalized welcome message.

# Theory:

1. List some of the core features of Flask

Core Features of Flask

* + Flask is a lightweight and ﬂexible web framework for Python. Some of its core features include:
  + Micro-framework – Flask is minimalistic and does not include built-in ORM, authentication, or admin panels.
  + Lightweight and Modular – Developers can add only the necessary components, keeping applications eﬃcient.
  + Built-in Development Server and Debugger – Provides an interactive debugger for error tracking.
  + Jinja2 Templating Engine – Supports dynamic HTML rendering with template inheritance.

Routing System – Allows handling multiple URLs using route decorators.

* + WSGI Compliance – Uses Werkzeug as its WSGI toolkit for handling requests.
  + Support for RESTful APIs – Simpliﬁes API development with built-in support for request handling.
  + Extensible with Extensions – Many third-party extensions are available for ORM, authentication, and other features.

1. Why do we use Flask( name ) in Flask?

The Flask( name ) function initializes a Flask application. The parameter

name helps:

* + Identify the App’s Module – Flask uses it to locate resources, templates, and static ﬁles.
  + Enable Debugging and Error Handling – Helps in logging and debugging by determining the root path of the application.
  + Allow Different Import Conﬁgurations – Ensures Flask works correctly whether run as a script or imported as a module.

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

Flask uses Jinja2 as its templating engine, allowing developers to create dynamic HTML pages.

* + Templates: HTML ﬁles that contain dynamic placeholders ({{ }} for variables and

{% %} for control structures like loops and conditions).

* + Template Inheritance: A feature where a base template is created with common elements (like headers and footers), and child templates extend it.
  + Beneﬁt: Avoids code duplication by keeping the layout consistent across multiple pages.

1. What methods of HTTP are implemented in Flask.

Flask supports multiple HTTP methods, including:

* + GET – Retrieves data from the server.
  + POST – Submits data to the server (e.g., form submission).
  + PUT – Updates an existing resource.
  + DELETE – Deletes a resource.
  + PATCH – Partially updates an existing resource.
  + HEAD – Similar to GET but retrieves only headers.
  + OPTIONS – Returns the allowed HTTP methods for a resource.

1. What is difference between Flask and Django framework



# Routing

@app.route('/')

def home():

name = request.args.get('name', 'Guest')  Default to 'Guest' if no name is provided return render\_template('index.html', name=name)

# URL building

@app.route('/user/<username>') def user\_proﬁle(username):

return f"Hello, {username}!"

Instead of hardcoding URLs, Flask provides url\_for(): url\_for('user\_proﬁle', username='Alice')

# GET REQUEST

@app.route('/') def home():

name = request.args.get('name', 'Guest')  Default to 'Guest' if no name is provided return f"Welcome, {name}!"

# POST REQUEST

@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', username=name, email=email)) return render\_template('contact.html')

**Github Link:** [**https://github.com/mahijodhani/Webx\_exp3**](https://github.com/mahijodhani/Webx_exp3)

## OUTPUT

**app.py**

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

    name = request.args.get('name', '')

    message = f"Welcome, {name}!" if name else "Welcome to the Flask Web App!"

    return render\_template('home.html', message=message)

@app.route('/contact')

def contact():

    return render\_template('contact.html')

@app.route('/thank\_you', methods=['POST'])

def thank\_you():

    name = request.form['name']

    email = request.form['email']

    return render\_template('thank\_you.html', name=name, email=email)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

## Home.html

<!DOCTYPE html>

<html>

<head>

    <title>Home - Flask App</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <div class="container">

        <h1>{{ message }}</h1>

        <a href="{{ url\_for('contact') }}">Go to Contact Form</a>

    </div>

</body>

</html>

## Contact.html

<!DOCTYPE html>

<html>

<head>

    <title>Contact Us</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <div class="container">

        <h1>Contact Form</h1>

        <form action="{{ url\_for('thank\_you') }}" method="post">

            <label for="name">Name:</label>

            <input type="text" name="name" required>

            <label for="email">Email:</label>

            <input type="email" name="email" required>

            <button type="submit">Submit</button>

        </form>

    </div>

</body>

</html>

**Thank\_you.html**

<!DOCTYPE html>

<html>

<head>

    <title>Thank You</title>

    <link rel=”stylesheet” href=”{{ url\_for(‘static’, filename=’style.css’) }}”>

</head>

<body>

    <div class=”container”>

        <h1>Thank You, {{ name }}!</h1>

        <p>We received your email: <strong>{{ email }}</strong></p>

        <a href=”{{ url\_for(‘home’) }}”>Back to Home</a>

    </div>

</body>

</html>

**Output:**





