Name: Gunjan Chandnani **Roll No: 06** **Div: D15A**

EXPERIMENT NO. 3 b

AIM: To design a Flask application that showcases URL building and demonstrates the use of HTTP methods (GET and POST) for handling user input and processing data.

PROBLEM STATEMENT: Create a Flask application with the following requirements:

1. A homepage (/) with links to a "Profile" page and a "Submit" page using the `url_for()` function.
2. The "Profile" page (`/profile/<username>`) dynamically displays a user's name passed in the URL.
3. A "Submit" page (`/submit`) displays a form to collect the user's name and age. The form uses the POST method to send the data, and the server displays a confirmation message with the input.

Theory:

1. What is a route in Flask, and how is it defined?

In Flask, a route is a URL pattern that is associated with a specific function in a web application. When a user visits a particular URL, Flask executes the corresponding function and returns the response. Routes define how the application should respond to different URLs.

A route is defined using the `@app.route()` decorator. For example:

```
from flask import Flask
app = Flask(__name__)

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

In this example, the root URL (/) is mapped to the `home` function, which returns a simple text response.

2. How can you pass parameters in a URL route?

In Flask, parameters can be passed in a URL route using angle brackets (`< >`). These parameters can be dynamic and help in passing values from the URL to the function.

Example:

```
@app.route('/user/<name>')
def greet_user(name):
    return f"Hello, {name}!"
```

If a user visits `/user/Sanket`, the function receives "Sanket" as the `name` parameter and returns "Hello, Sanket!".

Flask also allows specifying the data type of parameters:

```
@app.route('/square/<int:num>')
def square(num):
    return f"The square of {num} is {num**2}"
```

This ensures that `num` is treated as an integer.

3. What happens if two routes in a Flask application have the same URL pattern?

If two routes have the same URL pattern in a Flask application, only the last defined route will take effect, and the previous one will be overridden. Flask does not allow duplicate routes with the same URL pattern, as it would cause ambiguity.

Example:

```
@app.route('/hello')
def hello1():
    return "Hello from function 1"

@app.route('/hello')
def hello2():
    return "Hello from function 2"
```

In this case, when a user visits `/hello`, Flask will only execute `hello2()`, and `hello1()` will be ignored.

4. What are the commonly used HTTP methods in web applications?

The most commonly used HTTP methods in web applications are:

1. **GET** – Requests data from the server (e.g., retrieving a webpage).
2. **POST** – Sends data to the server (e.g., submitting a form).
3. **PUT** – Updates existing data on the server.

4. **DELETE** – Removes a resource from the server.
5. **PATCH** – Partially updates a resource.

In Flask, these methods can be specified in the `methods` parameter of the `@app.route()` decorator:

```
@app.route('/submit', methods=['POST'])
def submit():
    return "Form submitted!"
```

5. What is a dynamic route in Flask?

A dynamic route in Flask is a route that contains variables, allowing it to handle multiple different URLs with a single function. Dynamic routes make the web application more flexible by enabling the use of parameters within the URL.

Example:

```
@app.route('/user/<username>')
def profile(username):
    return f"User Profile: {username}"
```

If a user visits `/user/Sanket`, the function receives "Sanket" as a parameter and responds accordingly.

6. Write an example of a dynamic route that accepts a username as a parameter.

```
from flask import Flask
app = Flask(__name__)

@app.route('/user/<username>')
def show_user(username):
    return f"Welcome, {username}!"

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

In this example, the route `/user/<username>` accepts a username parameter from the URL and returns a personalized welcome message.

7. What is the purpose of enabling debug mode in Flask?

Enabling debug mode in Flask is useful for development because it provides:

1. **Automatic Code Reloading** – The server automatically restarts when changes are made to the code.
2. **Detailed Error Messages** – Flask displays an interactive debugger when an error occurs, making it easier to identify and fix issues.

However, debug mode should not be enabled in a production environment due to security risks.

8. How do you enable debug mode in a Flask application?

Debug mode can be enabled in two ways:

1. **Setting `debug=True` in `app.run()`**

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

2. **Using Environment Variables** In the terminal, set the environment variable before running the Flask app:

```
export FLASK_ENV=development
flask run
```

On Windows (Command Prompt):

```
set FLASK_ENV=development
flask run
```

This enables debug mode and allows for easier debugging during development.

Code:

```python
from flask import Flask, request, url_for, redirect, session



app = Flask(__name__)

app.secret_key = "supersecretkey"  # Required for session handling




@app.route('/')
```

```
def welcome():
    return '''
    <html>
    <head>
        <style>
            body { background: linear-gradient(to right, #ff512f,
#dd2476); text-align: center; color: white; }
            .container { margin-top: 100px; }
            a { display: inline-block; margin: 20px; padding: 15px 30px;
                background: rgba(0, 0, 0, 0.3); text-decoration: none;
                color: white; font-size: 18px; border-radius: 5px; }
            a:hover { background: rgba(0, 0, 0, 0.6); }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>Welcome to My Flask App</h1>
            <p>Your journey starts here!</p>
            <a href="{url_for('home')}">Get Started</a>
        </div>
    </body>
    </html>
    '''
```

```
@app.route('/home')
def home():
    return f'''
    <html>
    <head>
        <style>
            body {{ background: linear-gradient(to right, #ff7e5f, #feb47b); text-align: center; color: white; }}
            nav {{ padding: 10px; background: rgba(0, 0, 0, 0.4); }}
            nav a {{ margin: 10px; padding: 10px; text-decoration: none; color: white; border-radius: 5px; }}
            nav a:hover {{ background: rgba(0, 0, 0, 0.6); }}
        </style>
    </head>
    <body>
        <nav>
            <a href="{url_for('profile')}">Profile</a>
            <a href="{url_for('submit')}">Submit</a>
            <a href="{url_for('logout')}">Logout</a>
        </nav>
        <h1>Welcome to the Homepage</h1>
        <p>Explore our website using the links above.</p>
    </body>
```

```
    </html>

    '''




@app.route('/profile')

def profile():

    name = session.get('name', 'Guest')  # Get name from session or use
'Guest'

    age = session.get('age', 'Unknown')  # Get age from session or use
'Unknown'



    return f'''

    <html>

    <head>

        <style>

            body {{ background: linear-gradient(to right, #36d1dc,
#5b86e5); text-align: center; color: white; }}

            nav {{ padding: 10px; background: rgba(0, 0, 0, 0.4); }}

            nav a {{ margin: 10px; padding: 10px; text-decoration: none;
color: white; border-radius: 5px; }}

            nav a:hover {{ background: rgba(0, 0, 0, 0.6); }}

        </style>

    </head>

    <body>

        <nav>
```

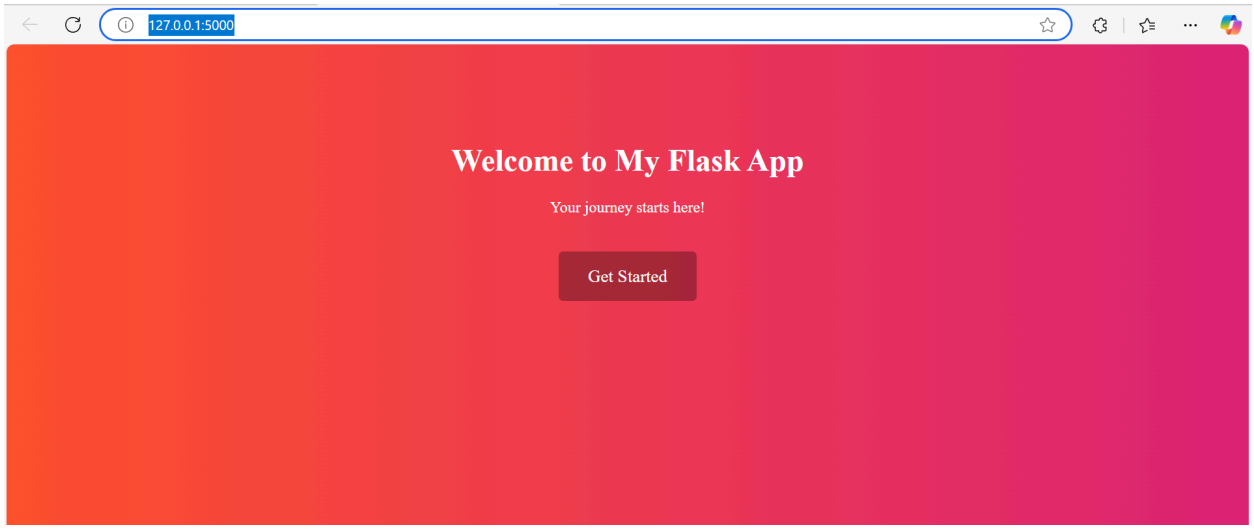
```
            <a href="{url_for('home')}">Home</a>
            <a href="{url_for('submit')}">Submit</a>
            <a href="{url_for('logout')}">Logout</a>
        </nav>
        <h1>Profile Page</h1>
        <p><strong>Name:</strong> {name}</p>
        <p><strong>Age:</strong> {age}</p>
    </body>
    </html>
    '''



@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        session['name'] = request.form.get('name', 'Unknown')
        session['age'] = request.form.get('age', 'Unknown')
        return redirect(url_for('profile'))

    return '''
    <html>
    <head>
        <style>
```

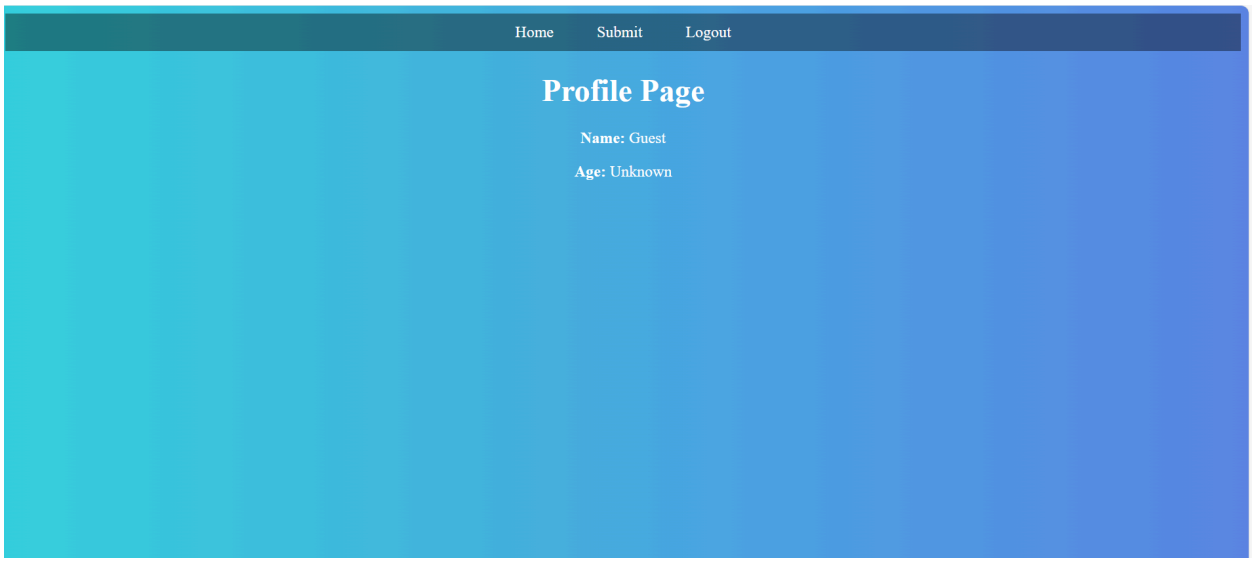
```
            body { background: linear-gradient(to right, #fc4a1a,
#f7b733); text-align: center; color: white; }

            nav { padding: 10px; background: rgba(0, 0, 0, 0.4); }

            nav a { margin: 10px; padding: 10px; text-decoration: none;
color: white; border-radius: 5px; }

            nav a:hover { background: rgba(0, 0, 0, 0.6); }

            form { display: inline-block; background: rgba(0, 0, 0, 0.2);
padding: 20px; border-radius: 10px; }

            input, select { margin: 5px; padding: 10px; border-radius:
5px; border: none; }

            input[type="submit"], input[type="reset"] { background:
rgba(0, 0, 0, 0.5); color: white; cursor: pointer; }

        </style>

    </head>

    <body>

        <nav>

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

            <a href="{url_for('profile')}">Profile</a>

            <a href="{url_for('logout')}">Logout</a>

        </nav>

        <h1>Submit Page</h1>

        <form method="post">

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

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

            <label for="age">Age:</label>
```

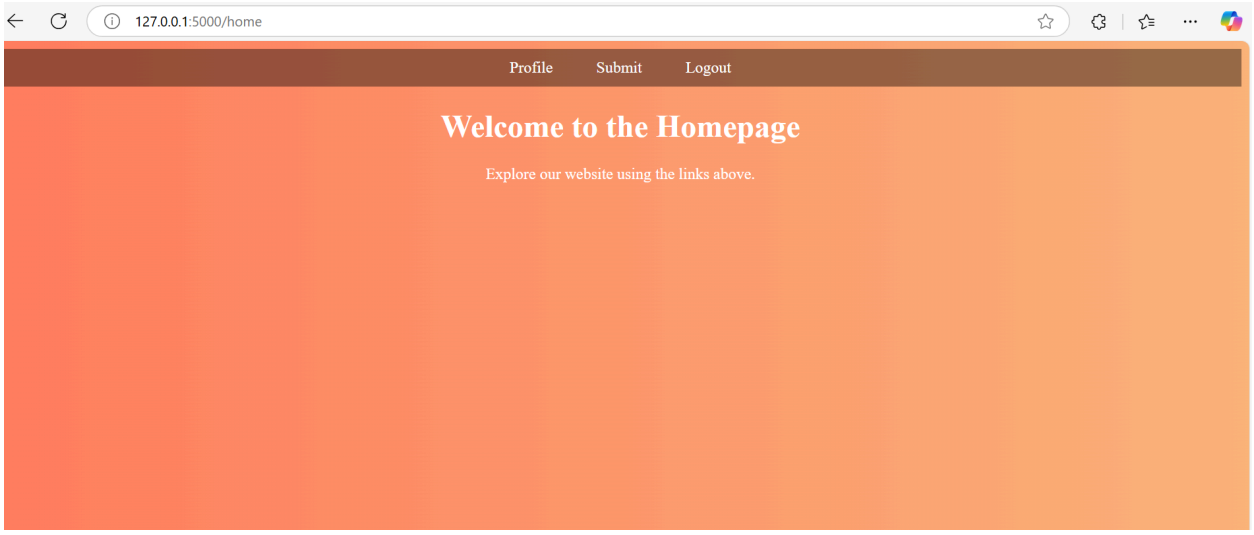
```
            <select id="age" name="age">

                ''' + ''.join([f'<option value="{i}">{i}</option>' for i
in range(18, 100)]) + '''

            </select><br>

            <input type="submit" value="Submit">

            <input type="reset" value="Reset">

        </form>

    </body>

    </html>

    '''




@app.route('/logout')

def logout():

    session.clear()  # Clear session data

    return redirect(url_for('home'))




if __name__ == '__main__':

    app.run(debug=True)
```

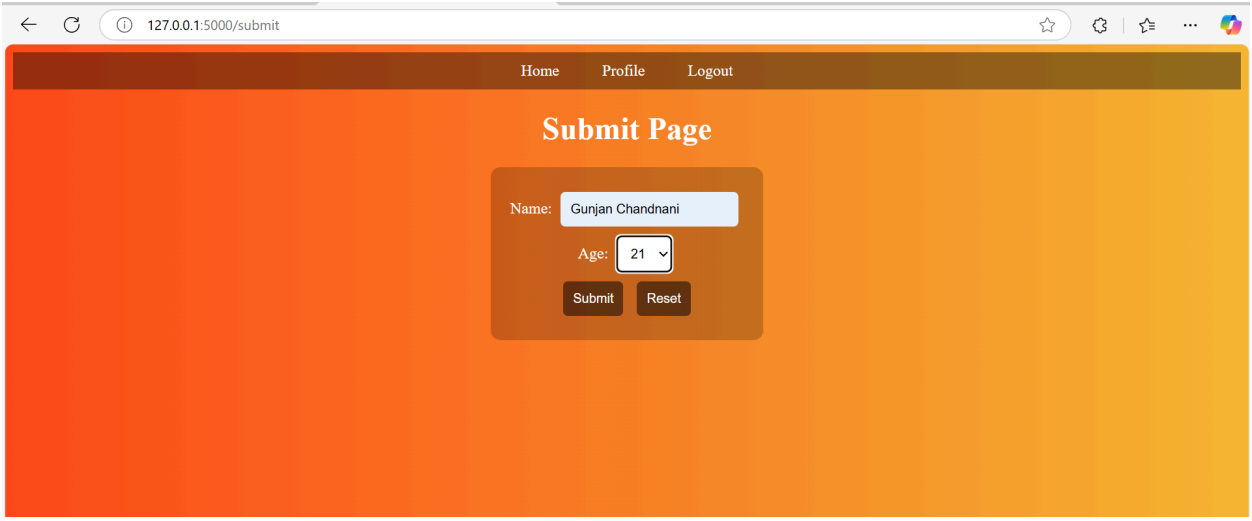
Result:

```
PS D:\programs\webx> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inst
ead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 184-134-416
```






127.0.0.1:5000/home
Profile
Submit
Logout
Welcome to the Homepage
Explore our website using the links above.


127.0.0.1:5000/submit
Home
Profile
Logout
Submit Page
Name:
Gunjan Chandnani
Age:
21
Submit
Reset