EXPERIMENT NO. 4 - Flask Application using GET and POST

Name of Student	**Rohan Lalchandani**
Class Roll No	**25**
D.O.P.	**27/02/2025**
D.O.S.	**06/03/2025**
Sign and Grade	

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?**

   A route in Flask is a URL pattern linked to a function using the `@app.route()` decorator. Example:

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

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

   Parameters can be passed in a route using angle brackets (`< >`) in the URL. Flask captures these values and passes them to the function as arguments. You can also specify data types like `<int:id>` or `<string:name>`. Example:

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

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

   If two routes share the same URL pattern, Flask will use the last-defined route, overriding the previous one. This causes unexpected behavior. Example:

```
@app.route('/hello') def
hello1():
     return "Hello from function 1"
@app.route('/hello') def
hello2():
     return "Hello from function 2"
# Only "Hello from function 2" will be shown.
```

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

   HTTP methods define the type of request a client sends to a server:

   - **GET:** Retrieve data (e.g., accessing a web page).
   - **POST:** Send data to the server (e.g., submitting a form).
   - **PUT:** Update existing data.
   - **DELETE:** Remove data.
   - **PATCH:** Partially update data.

5. **What is a dynamic route in Flask?**

   A dynamic route allows variables to be embedded within the URL, making it more flexible. The data in the URL is passed to the function for further processing. Example:

```
@app.route('/profile/<username>') def
show_profile(username):  return f"Welcome to
{username}'s Profile!"
```

6. **Write an example of a dynamic route that accepts a username as a parameter.**
   Example:
   ```
   @app.route('/user/<username>') def
   welcome_user(username):  return f"Hello, {username}!
   Glad to see you here."
   ```

7. **What is the purpose of enabling debug mode in Flask?**
   Debug Mode is used during development for easy troubleshooting. It enables:
   - **Automatic Code Reloading:** The app restarts when code changes.
   - **Detailed Error Messages:** Displays an interactive debugger in case of an error.

   It should be **disabled** in production for security reasons.

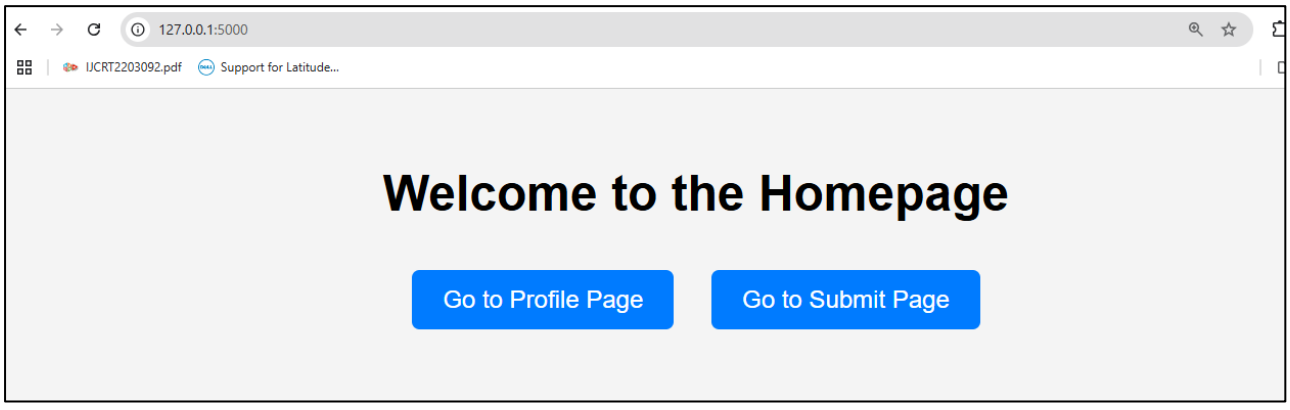
8. **How do you enable debug mode in a Flask application?** You can enable debug mode using one of these methods:
   - Using `app.run(debug=True)`
   - `export FLASK_ENV=development`
   - `flask run`

GITHUB LINK –

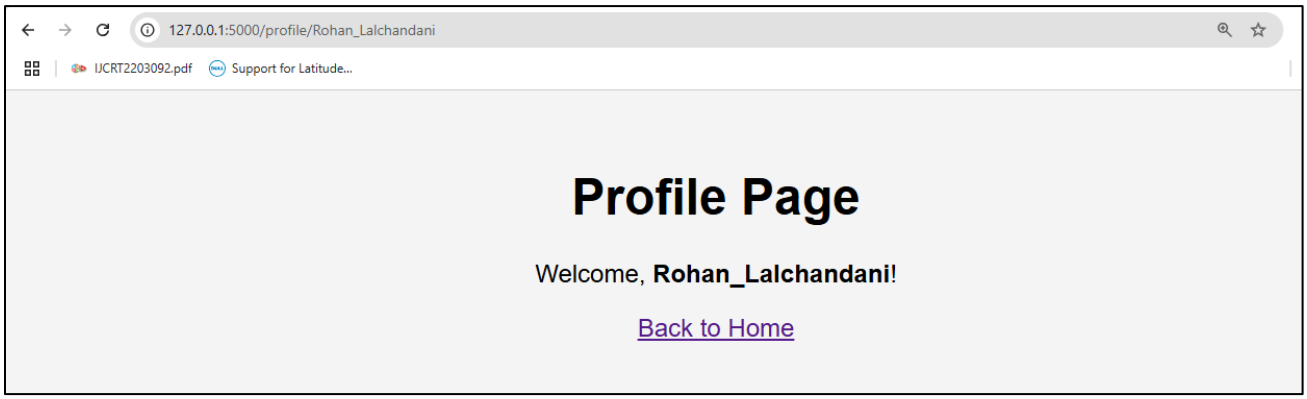
https://github.com/Rohan-Lalchandani08/WebX_Lab/tree/main/WebX_Exp4

OUTPUT

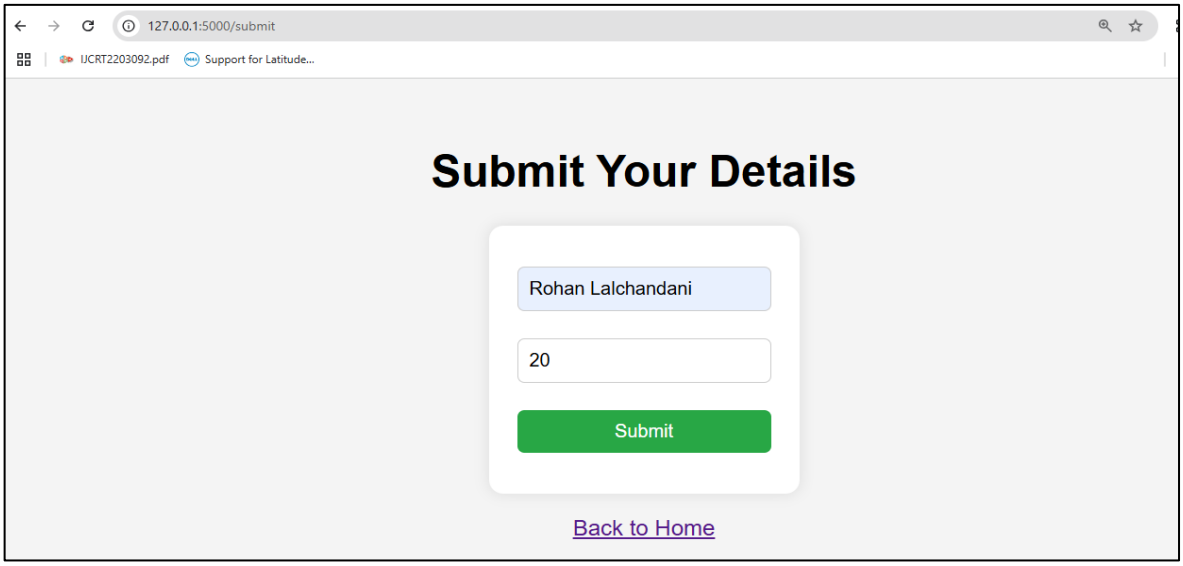
- **Homepage:** The homepage will display two links: one to the "Profile" page and another to the "Submit" page.

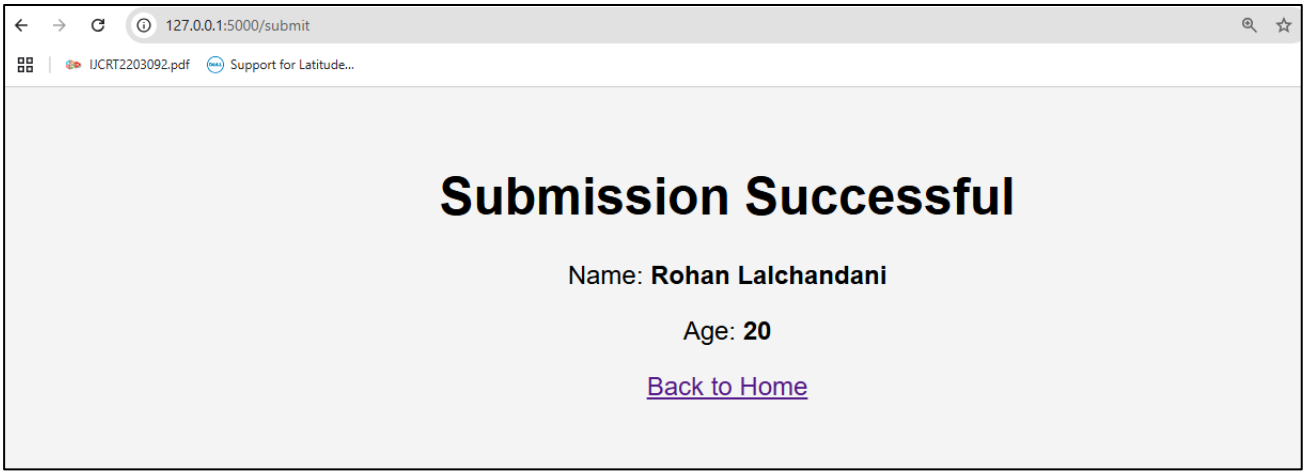


- **Profile Page:** After submitting the form with your name and age, the profile page will display your details dynamically.



- **Submit Page:** The form will allow users to enter their name and age, and upon submission, they will be redirected to the profile page with the entered details.





CONCLUSION

The experiment successfully demonstrated the implementation of **GET and POST** methods in a Flask application. It involved creating a **homepage (/)** with links to other pages, a **dynamic profile page (/profile/<username>)** that displayed user-specific data from the URL, and a **form-based submit page (/submit)** that handled user input via the **POST** method.

Through this experiment, key Flask concepts such as **URL building (url_for), dynamic routing, form handling, HTTP methods, and enabling debug mode** were explored. This practical implementation highlighted Flask's capability to create interactive web applications with minimal code.