EXPERIMENT NO. 4

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:

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

A route in Flask is a URL pattern that maps to a specific function, allowing users to navigate web pages or interact with an application. Routes define how the app responds to specific URL requests.

Definition: In Flask, routes are defined using the `@app.route()`

decorator: python

```
from flask import
Flask app = Flask(__
name__________)
@app.route('/')
def home():
    return "Welcome to the homepage!"
```

How can you pass parameters in a URL route?

You can pass parameters in a Flask URL by defining dynamic routes using angle brackets (<>). Parameters are captured from the URL and passed to the function.

Example:

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

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

If two routes have the same URL pattern, Flask will raise an error when the application starts. Flask does not allow duplicate routes because it wouldn't know which function to execute for a given URL.

Example (incorrect usage):

```
@app.route('/home')
def home1():
     return "Home Page"

@app.route('/home')    # Duplicate route will cause error

def home2():
    return "Another Home Page"
```

What are the commonly used HTTP methods in web applications?

The most commonly used HTTP methods in Flask and web applications are:

- GET → Retrieves data from the server (default method).
- POST → Sends data to the server (used in forms, API requests).
- PUT → Updates an existing resource.
- DELETE → Removes a resource from the server.

Example with multiple methods:

```
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        return "Form submitted!"
    return "Fill out the form."
```

What is a dynamic route in Flask?

A dynamic route in Flask allows changing parts of the URL to pass data as parameters. This makes the URL flexible, so it can handle different user inputs dynamically.

Example:

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

Here, `username` is a dynamic part of the URL.

Example of a dynamic route that accepts a username as a parameter

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

If a user visits `/user/John`, the output will be:
"Hello, John!"

What is the purpose of enabling debug mode in Flask?

Debug mode in Flask helps developers by:

- Automatically restarting the server when code changes.
- Displaying detailed error messages in the browser.
- Allowing interactive debugging in case of errors.
- It makes development easier by catching errors early.

How do you enable debug mode in a Flask application?

You can enable debug mode in two ways:

- Using code in

  `app.py`: python

  CopyEdit

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

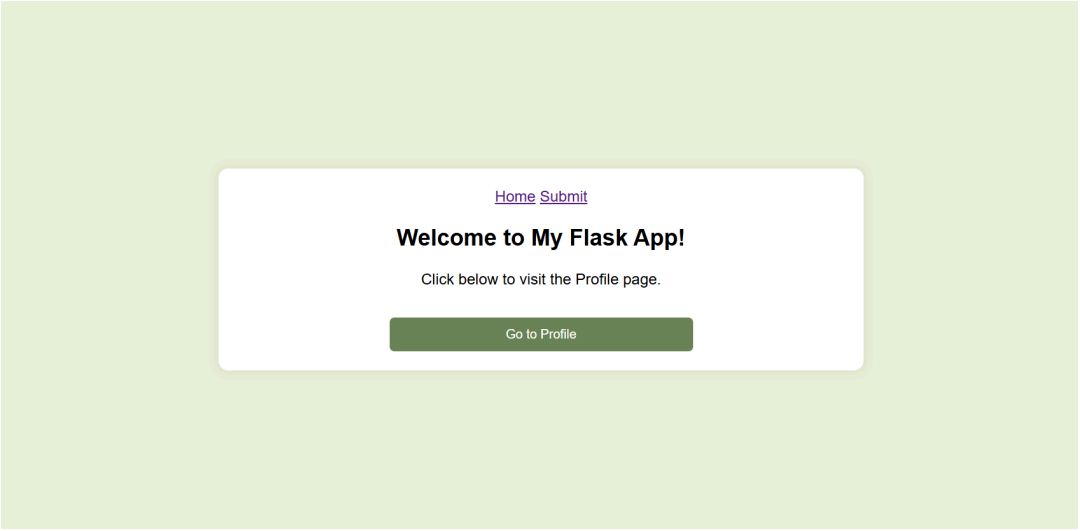
- Using environment variables (recommended for

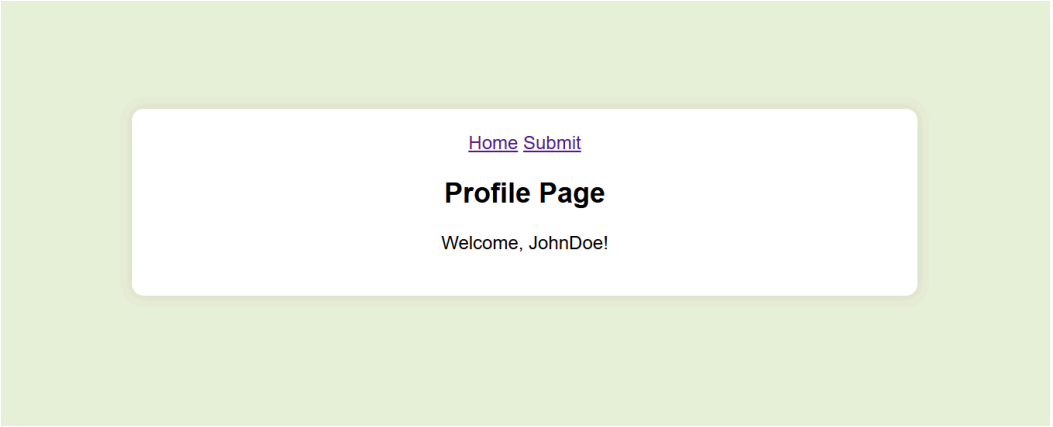
  security): bash

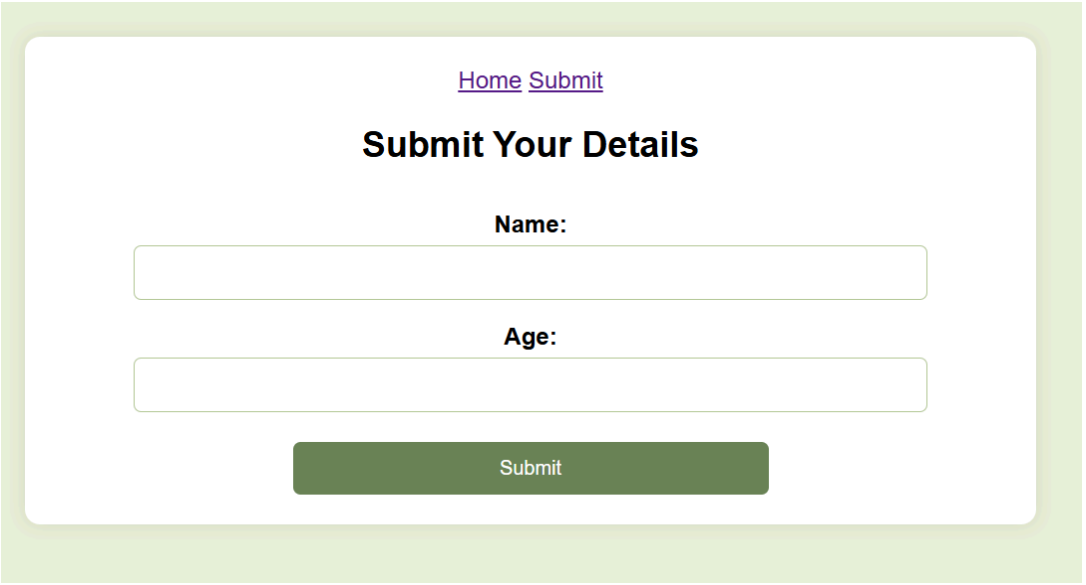
  CopyEdit

  ```
  export FLASK_ENV=development
  export FLASK_DEBUG=1
  flask run
  ```

OUTPUT


Home Submit
Welcome to My Flask App!
Click below to visit the Profile page.
Go to Profile


Home Submit
Profile Page
Welcome, JohnDoe!


Home Submit
Submit Your Details
Name:
Age:
Submit