Experiment NO. 5: Flask Application using render_template() function.

Name of Student	Rohan Lalchandani
Class Roll No	D15A_25
D.O.P.	06/03/2025
D.O.S.	13/03/2025
Sign and Grade	

AIM : To create a Flask application that demonstrates template rendering by dynamically generating HTML content using the `render_template()` function.

PROBLEM STATEMENT :

Develop a Flask application that includes:

1. A homepage route (`/`) displaying a welcome message with links to additional pages.
2. A dynamic route (`/user/<username>`) that renders an HTML template with a personalized greeting.
3. Use Jinja2 templating features, such as variables and control structures, to enhance the templates.

THEORY :

1. **What does the `render_template()` function do in a Flask application?**
   The `render_template()` function is used to render HTML templates stored in the **templates** folder. It dynamically generates web pages by passing variables from the Flask app to the template using **Jinja2**.

2. **What is the significance of the templates folder in a Flask project?**
   - The **templates** folder is the default location where Flask looks for HTML files.
   - It maintains a clean separation between business logic (Python code) and presentation logic (HTML).

- Using the **templates** folder allows developers to use **Jinja2** for rendering dynamic content.

- The folder can also store reusable components like base templates, headers, or footers using **template inheritance**.

3. **What is Jinja2, and how does it integrate with Flask?**
   Jinja2 is a templating engine used in Flask to render dynamic HTML content. It allows embedding Python expressions inside HTML files. Using **Jinja2**, you can:
   - Display variables
   - Apply logic (like loops and conditionals)
   - Apply filters for formatting

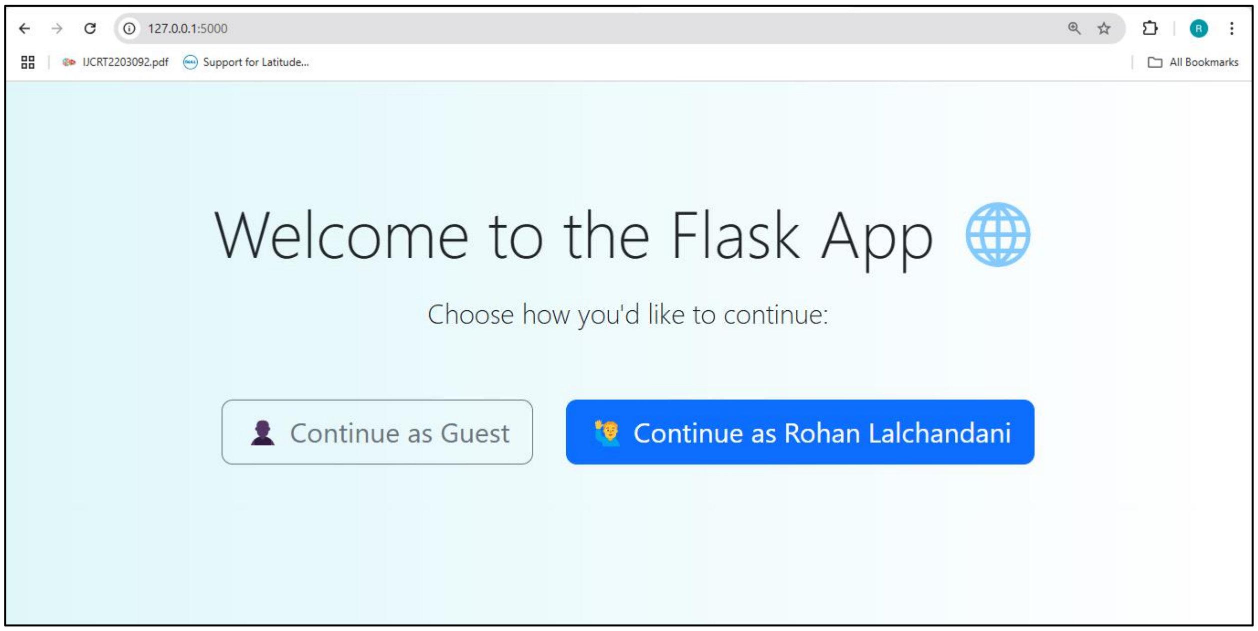
   Flask integrates **Jinja2** by default using the `render_template()` function.

GITHUB LINK –

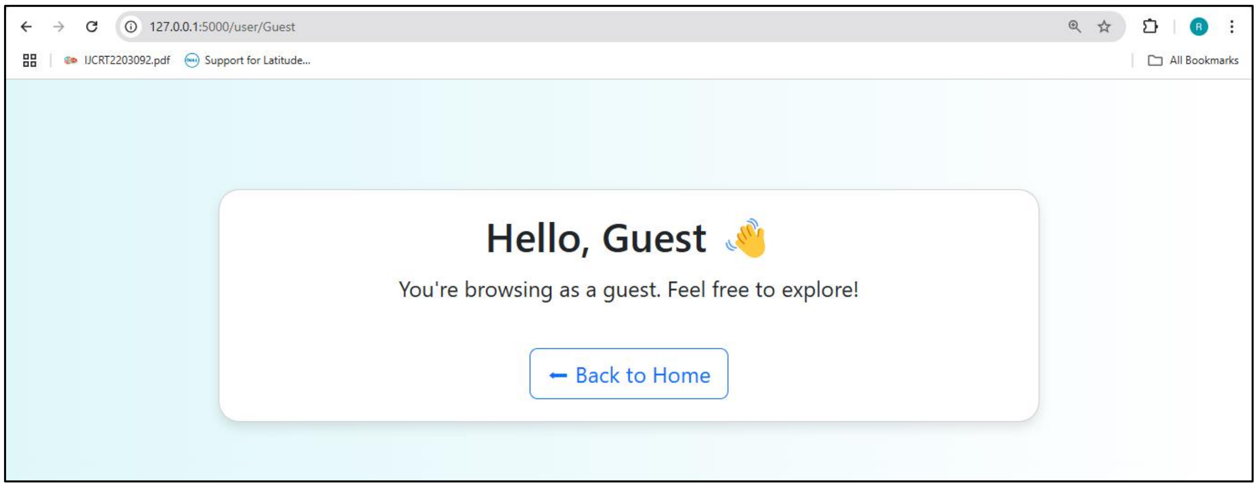
https://github.com/Rohan-Lalchandani08/WebX_Lab/tree/main/WebX_Exp5/flaskapp

OUTPUT

- **Homepage (`/`)**: The homepage displays a welcome message along with two links for user-specific pages (e.g., Guest's Page and Rohan's Page).

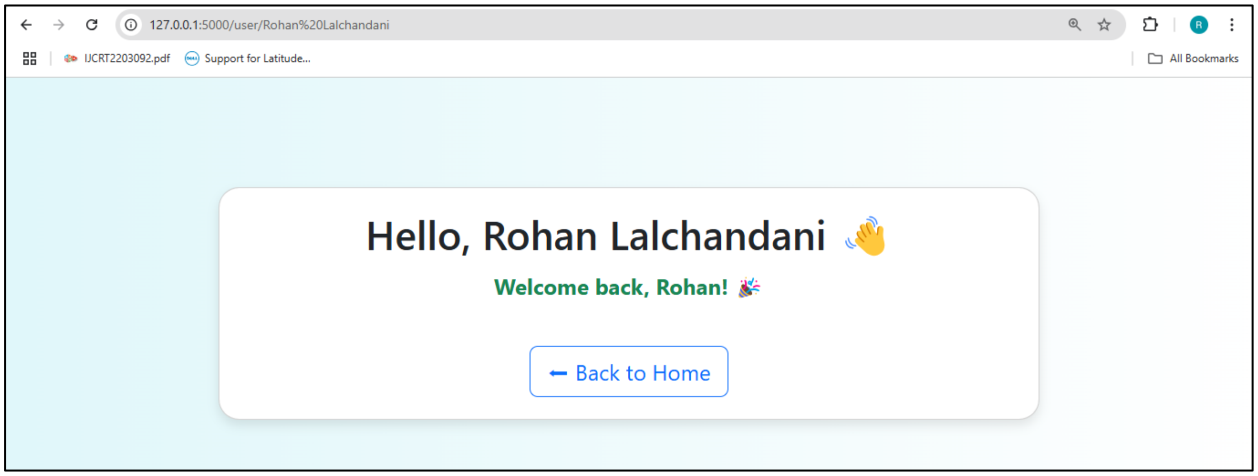

127.0.0.1:5000
IJCRT2203092.pdf
Support for Latitude...
All Bookmarks
Welcome to the Flask App 🌐
Choose how you'd like to continue:
Continue as Guest
Continue as Rohan Lalchandani

- **User Page (`/user/<username>`)**: When clicking on any of the user links, the app renders a personalized greeting with the username passed as a URL parameter.


127.0.0.1:5000/user/Guest

Hello, Guest 👋

You're browsing as a guest. Feel free to explore!


127.0.0.1:5000/user/Rohan%20Lalchandani

Hello, Rohan Lalchandani 👋

Welcome back, Rohan! 🎉

CONCLUSION

The experiment successfully demonstrated the use of the **render_template()** function in Flask to dynamically generate HTML content. A **homepage (/)** was created with links to user-specific pages, and a **dynamic route (/user/<username>)** was implemented to personalize greetings using Jinja2 templating.

This experiment highlighted key Flask concepts such as **template rendering, Jinja2 syntax, variable passing, and dynamic content generation**, showcasing how Flask efficiently separates business logic from presentation logic to create interactive web applications.