

Dương Ngọc Linh Đan – 2374802010091

**Result:**

### **3 Practical Exercises**

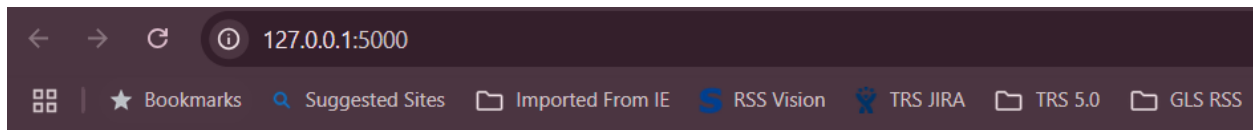
#### **3.1 Exercise 1:**

Create a file named hello.py with the following content

Sample folder structure: the following is the sample folder structure for this exercise (and similarly for the next exercises):

hello.py

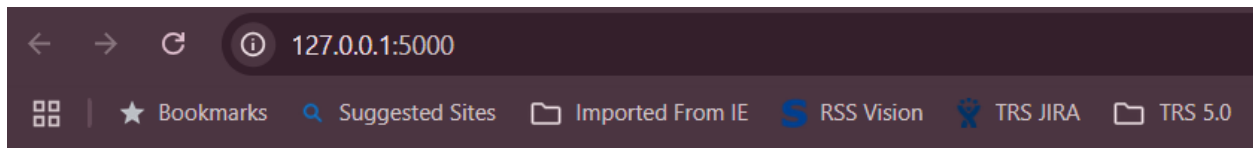
templates/ hello.html



# **Hello geeksforgeeks!**

#### **3.2 Exercise 2:**

Create a file named homepage.py with the following content



# Welcome to My Website

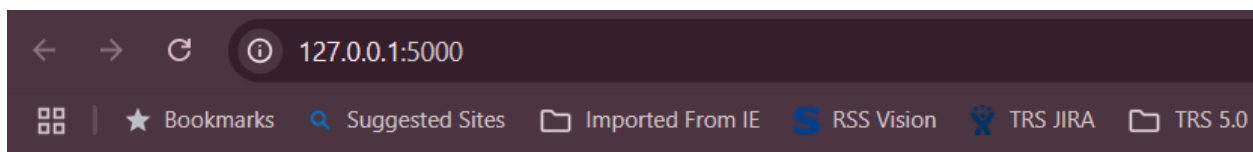
[Home](#) | [About](#)

## Home Page

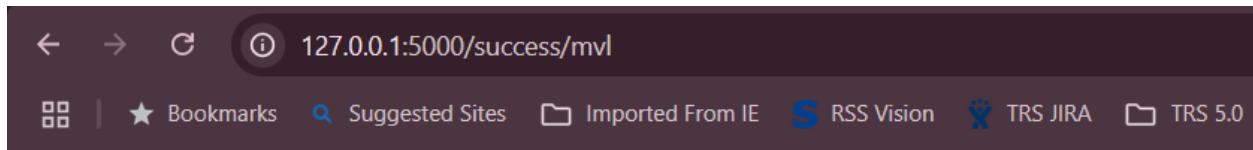
Welcom to homepage of our Flask app!

### 3.3 Exercise 3:

Create a file named name.py with the following content



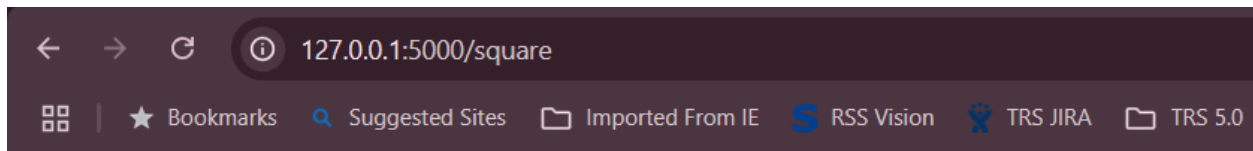
Enter Name:



Welcome mvl

### 3.4 Exercise 4:

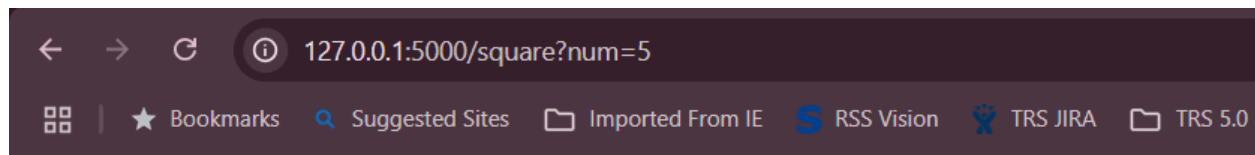
Create a file named math.py with the following content



## Welcome to the Maths Page!

Enter a number to find its square:

Enter a number:



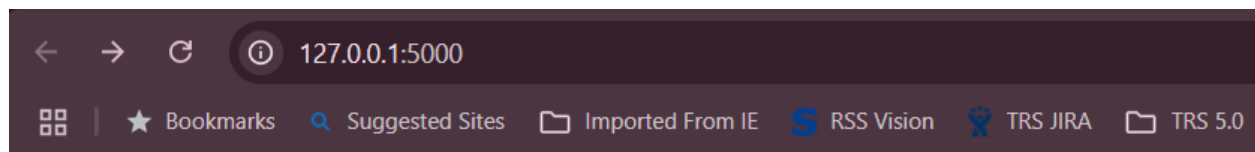
# Maths Calculation Result

The square of 5 is: 25

[Try another number](#)

## 3.5 Exercise 5:

Create a file named score.py with the following content

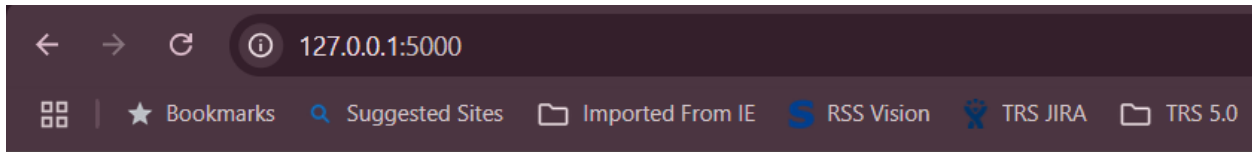


Name

Physics

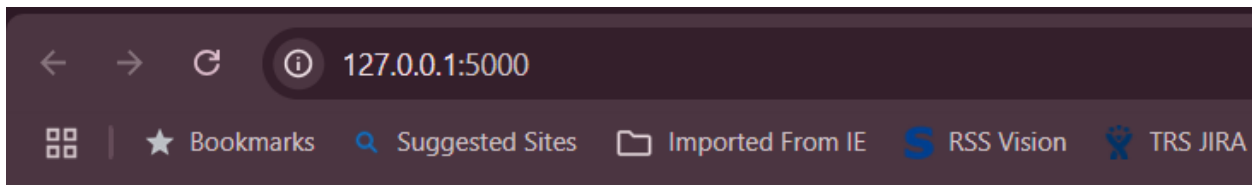
Chemistry

Maths

	
<b>Mathematics</b>	9.2
<b>chemistry</b>	9
<b>Physics</b>	8.5
<b>Name</b>	Nguyen Thi B

### 3.6 Exercise 6:

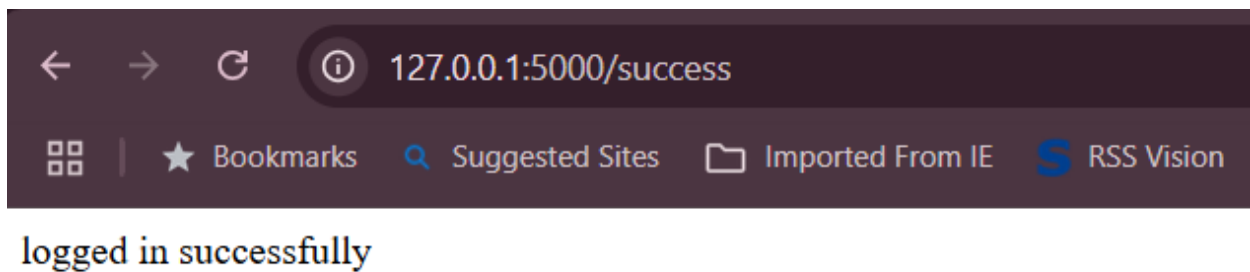
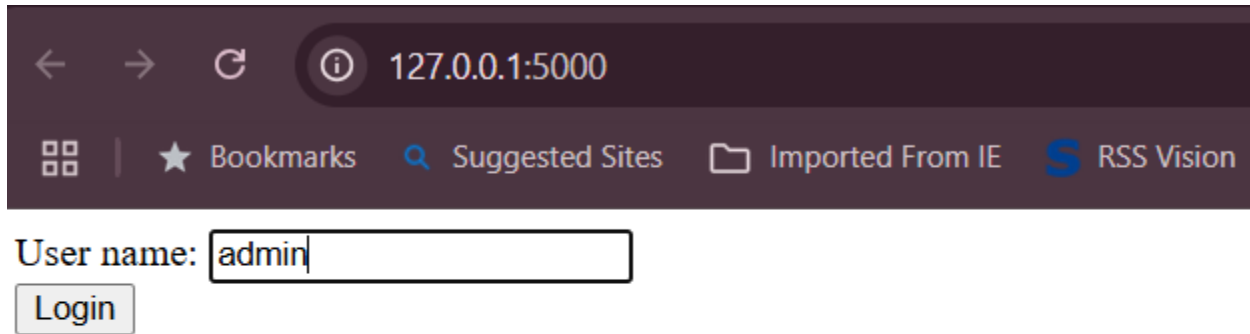
Create a file named newfile.py with the following content

	
<input type="button" value="Choose File"/>	trigger in T-sql.jpg <input type="button" value="Submit"/>

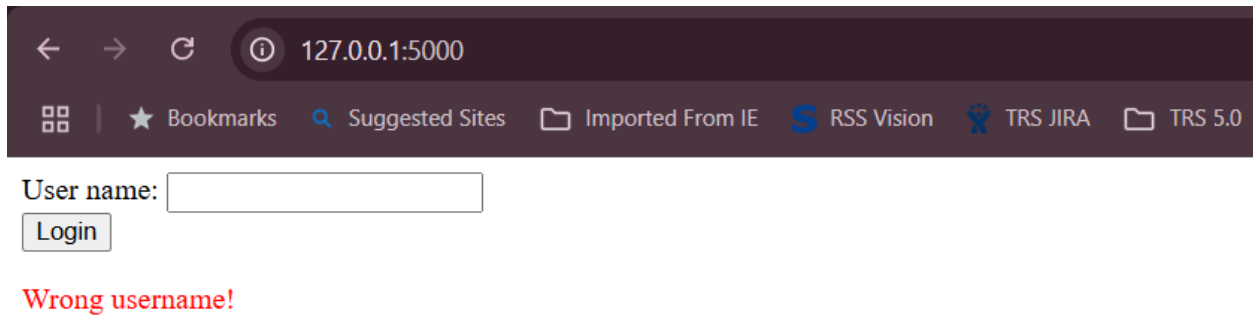
### 3.7 Exercise 7:

Create a file named login.py with the following content

- When I enter “admin”, it shows “logged in successfully”



- But when I enter another name such as “dan”, it shows message: “Wrong username!”



User name:

Login

Wrong username!

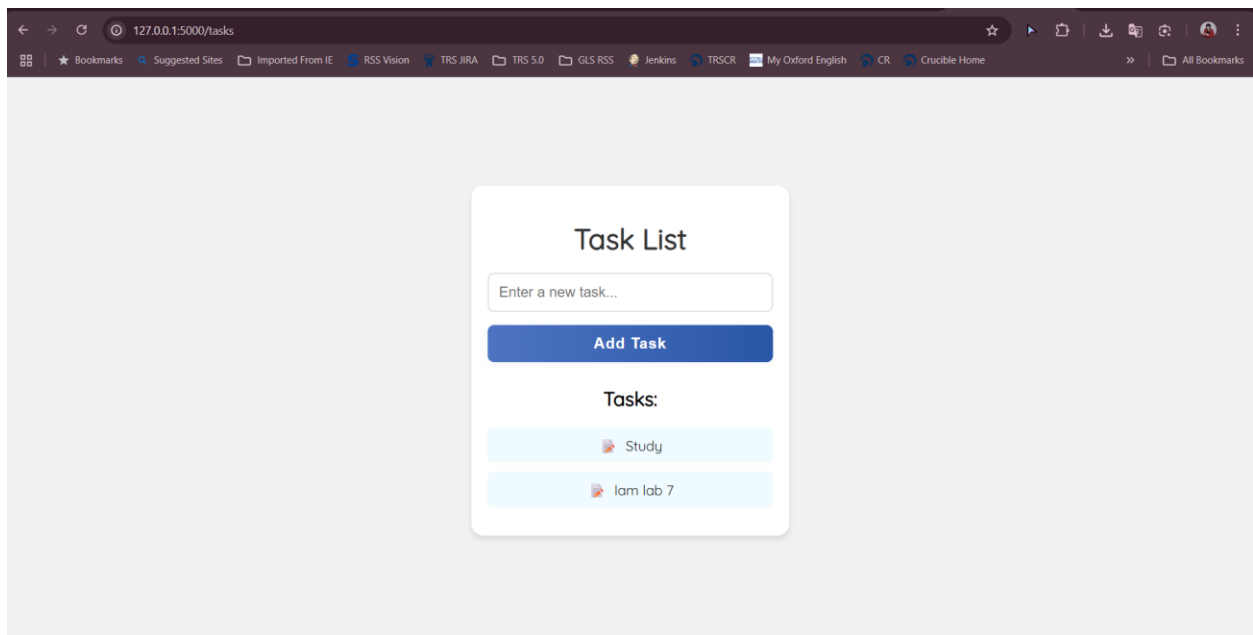
### 3.8 Exercise 8: Simple Task List

Create a file named `tasklist.py` that implements a basic task list application. Sample folder structure:

`tasklist.py`

`templates/ tasklist.html`

Create a route `/tasks` to display a list of tasks stored in a Python list. Include a form in `tasklist.html` to add tasks via POST. Append new tasks and refresh to show the updated list. Display tasks in an unordered list.



Task List

Enter a new task...

Add Task

Tasks:

- Study
- Iam lab 7

### 3.9 Exercise 9: User Profile Page

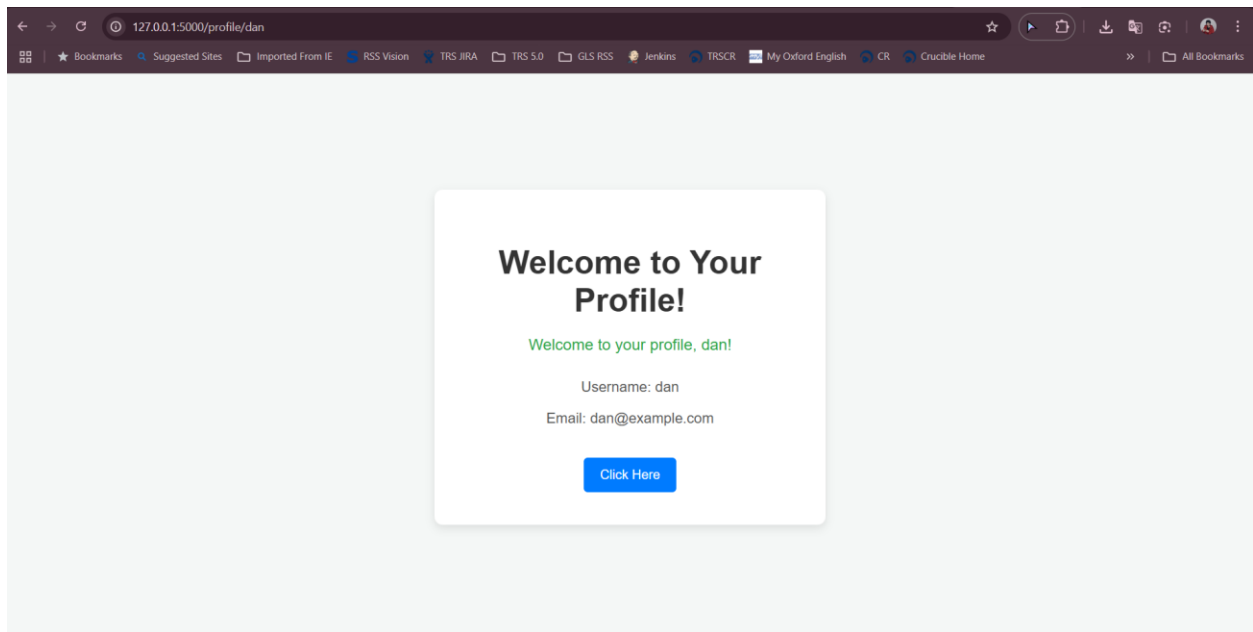
Create a file named `profile.py` that displays a user profile based on a username. Sample folder structure:

`profile.py`

`templates/ profile.html`

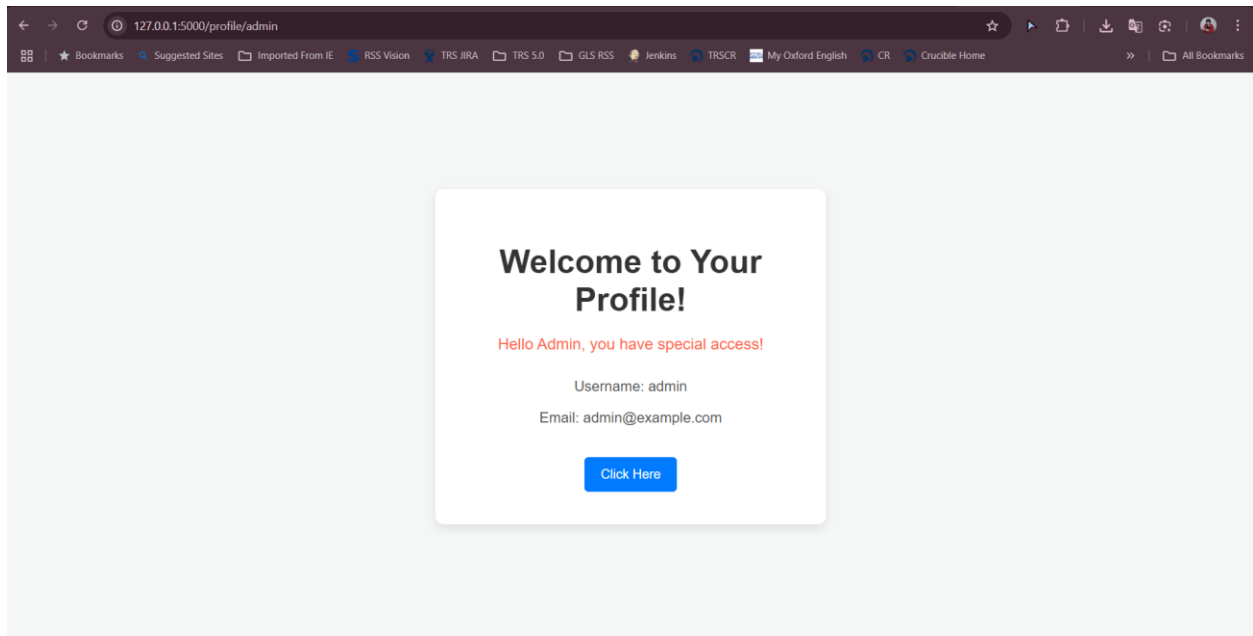
Create a route `/profile/<username>` accepting a username. Render `profile.html` with the username in a heading. Show "Welcome to your profile, [username]!". If username is "admin", display "Hello Admin, you have special access!".

- Show "Welcome to your profile, [username]!".



- If username is "admin", display "Hello Admin, you have special access!".





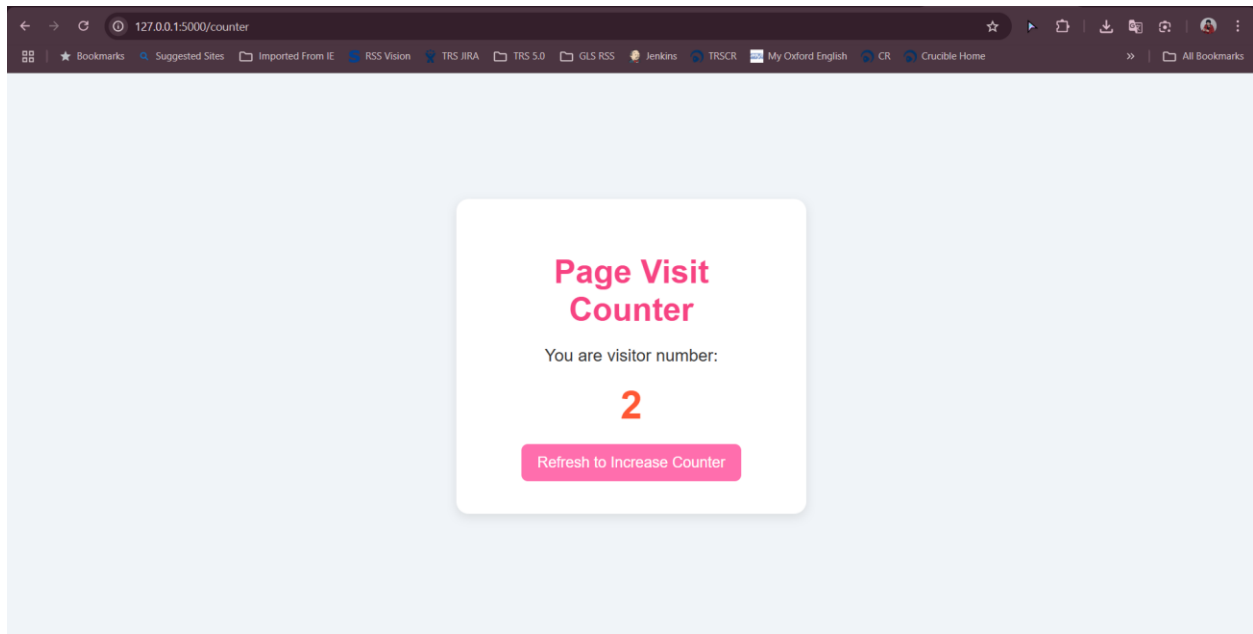
### 3.10 Exercise 10: Simple Counter

Create a file named `counter.py` that tracks page visits. Sample folder structure:

`counter.py`

`templates/ counter.html`

Create a route `/counter` using a global variable to track visits. Render `counter.html` with the count in a heading. Apply basic CSS (center text, background color)



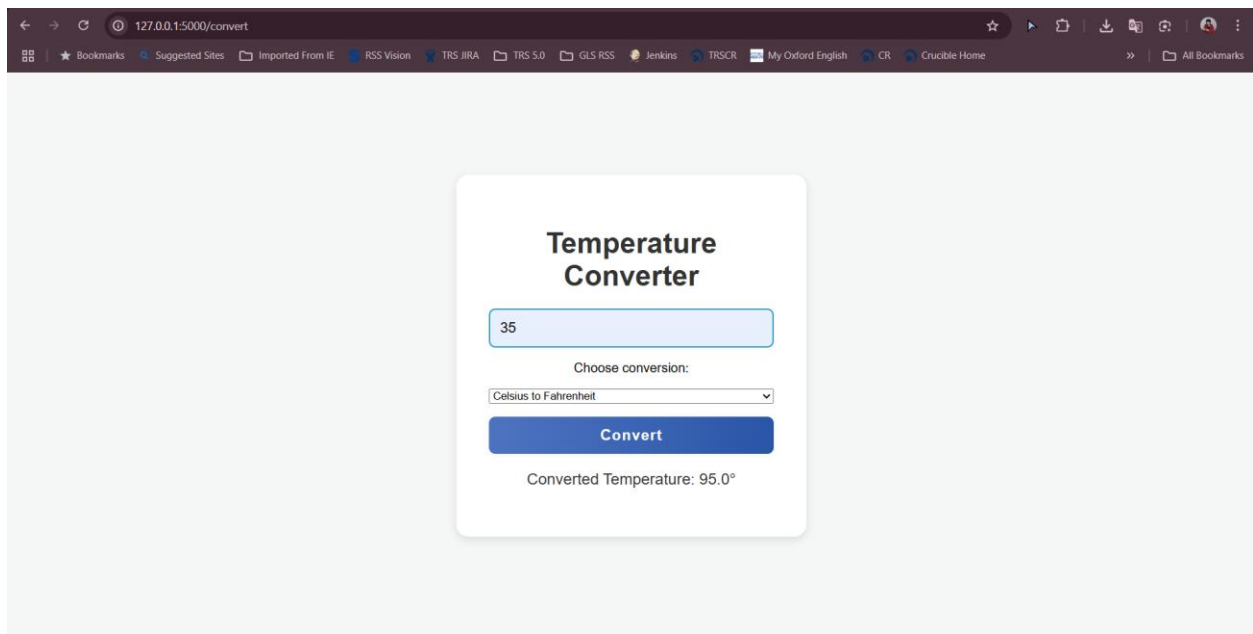
### 3.11 Exercise 11: Temperature Converter

Create a file named temperature.py for temperature conversion. Sample folder structure:

temperature.py

templates/ temperature.html

Create a route /convert with a form for temperature and conversion type (Celsius to Fahrenheit or vice versa). Handle POST to convert (C to F:  $F = C * 9/5 + 32$ ; F to C:  $C = (F - 32) * 5/9$ ). Display result below the form.



### 3.12 Exercise 12: Dynamic Blog Post

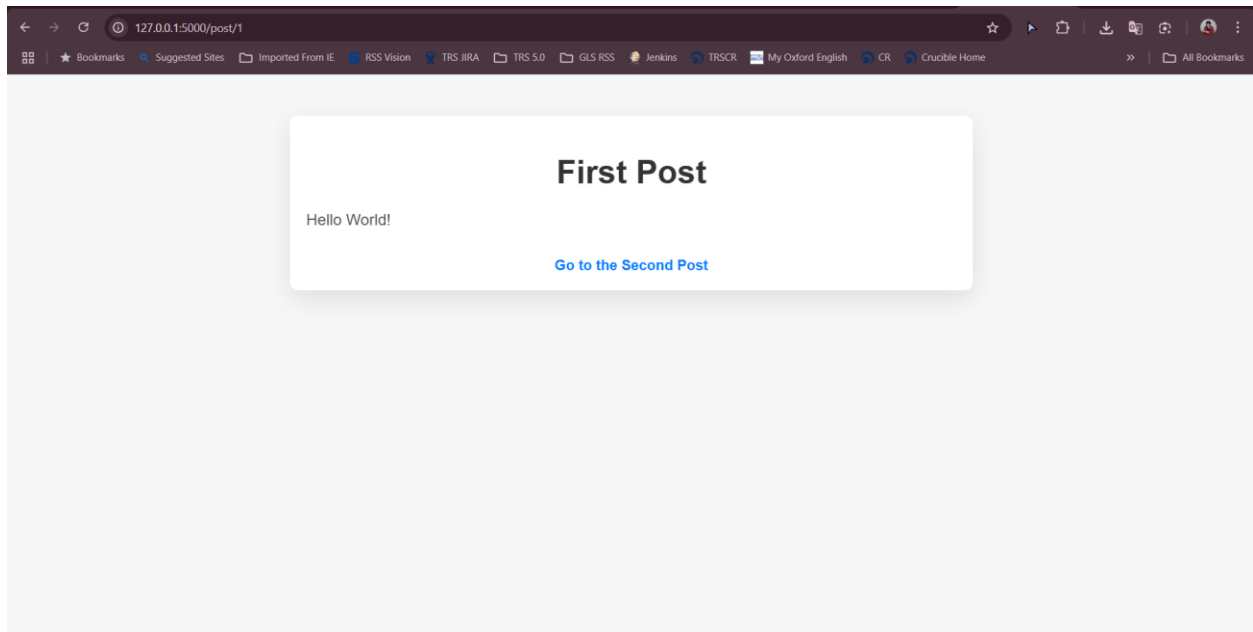
Create a file named `blog.py` to display blog posts. Sample folder structure:

`blog.py`

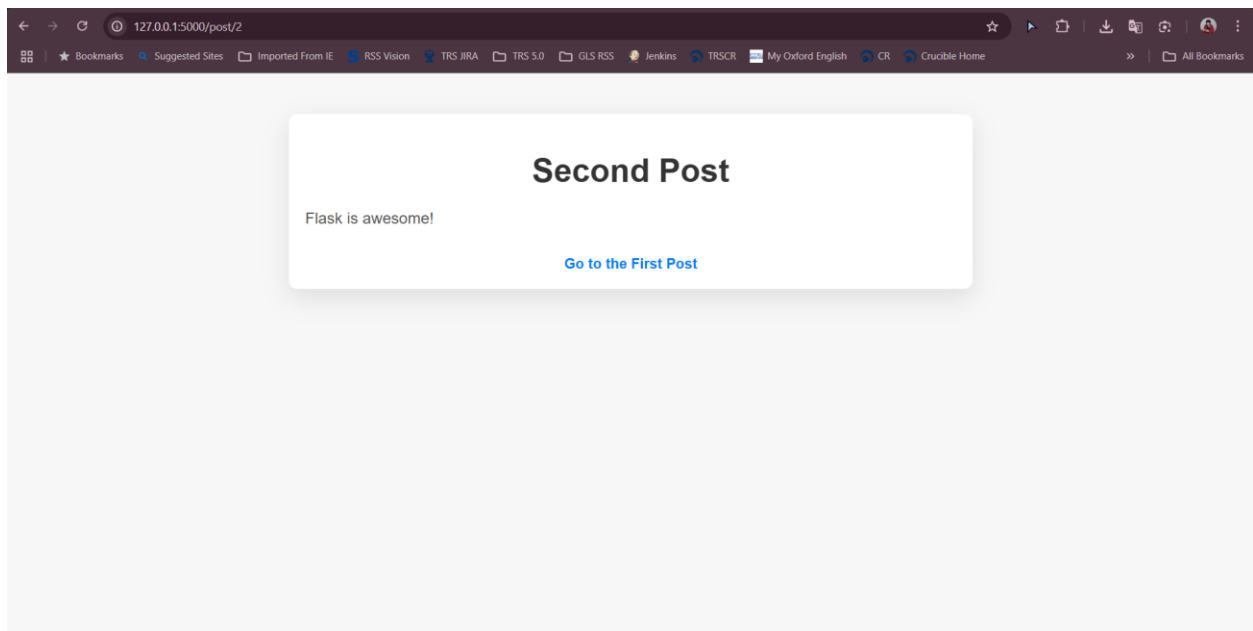
`templates/ blog.html`

Create a route `/post/<int:post_id>` for integer post IDs. Use a dictionary for posts (e.g., `{1: {"title": "First Post", "content": "Hello, world!"}}`). Render `blog.html` with title and content. Return 404 for invalid IDs: "Post not found".

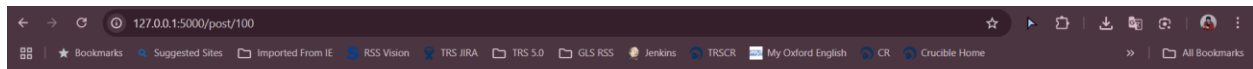
- POST 1:



- POST 2:



- POST “Not Found”:



## Not Found

Post not found

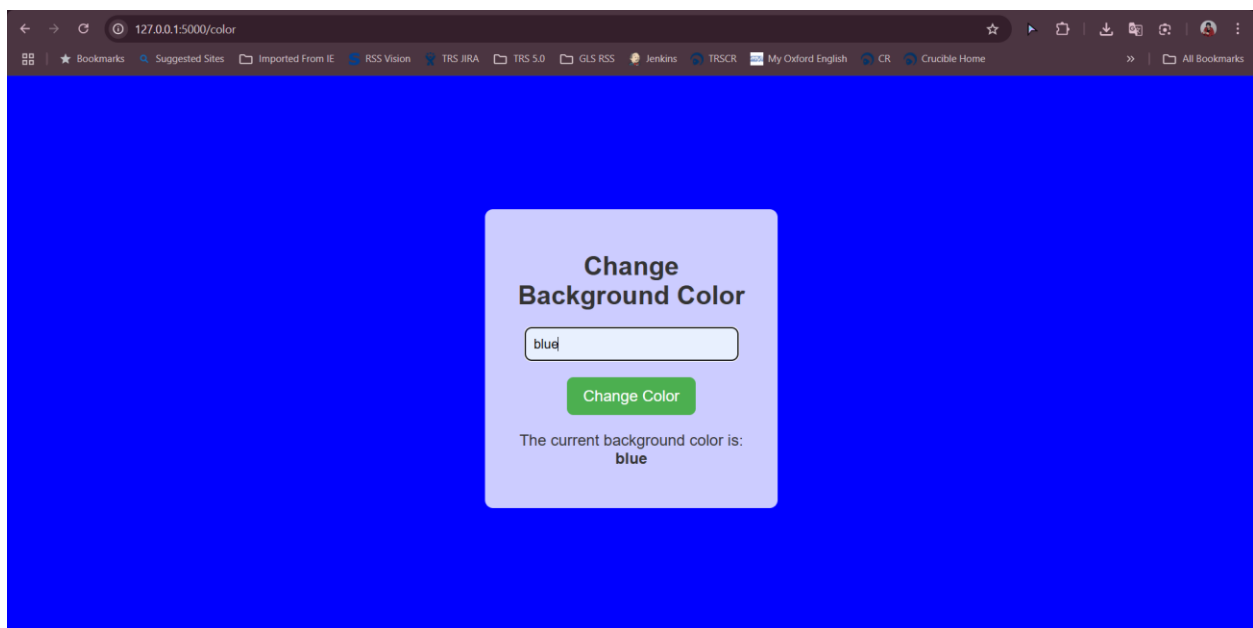
### 3.13 Exercise 13: Color Changer

Create a file named `color.py` to change the page background color. Sample folder structure:

`color.py`

`templates/ color.html`

Create a route `/color` with a form to input a color (e.g., "red", "#FF0000"). Handle POST to set the background color of `color.html`. Default color is white if no input.



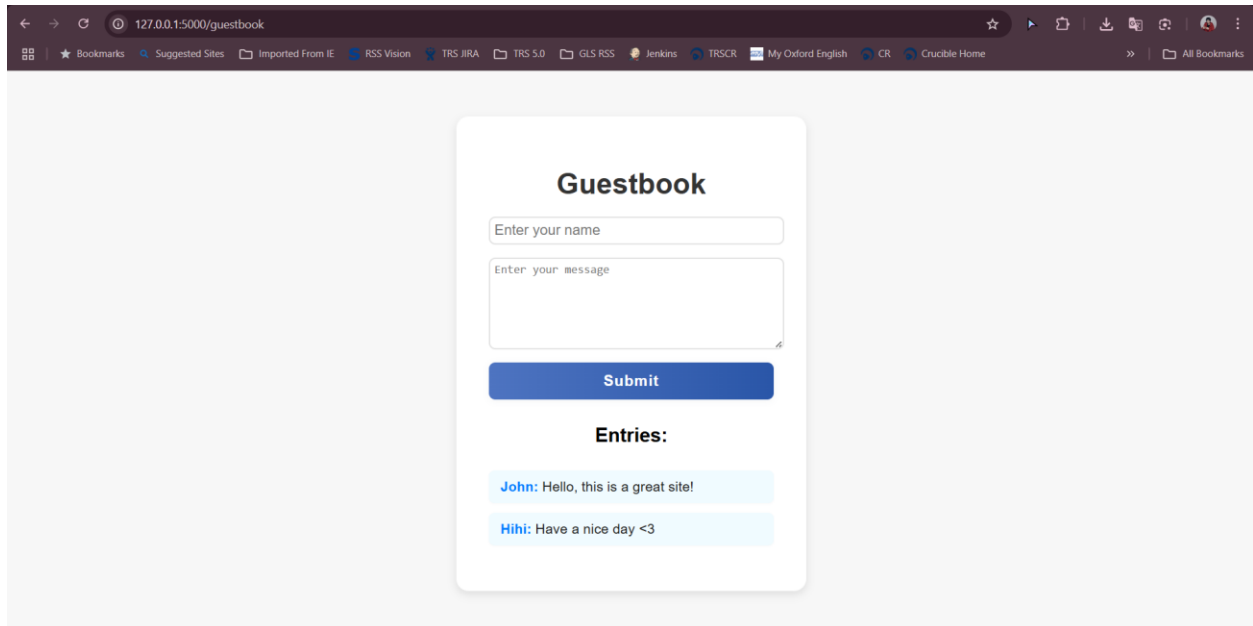
### 3.14 Exercise 14: Guestbook

Create a file named `guestbook.py` for a guestbook application. Sample folder structure:

`guestbook.py`

`templates/ guestbook.html`

Create a route `/guestbook` to display entries (name and message) in a list. Include a form to submit new entries via POST. Store entries in a list and display in `guestbook.html`.



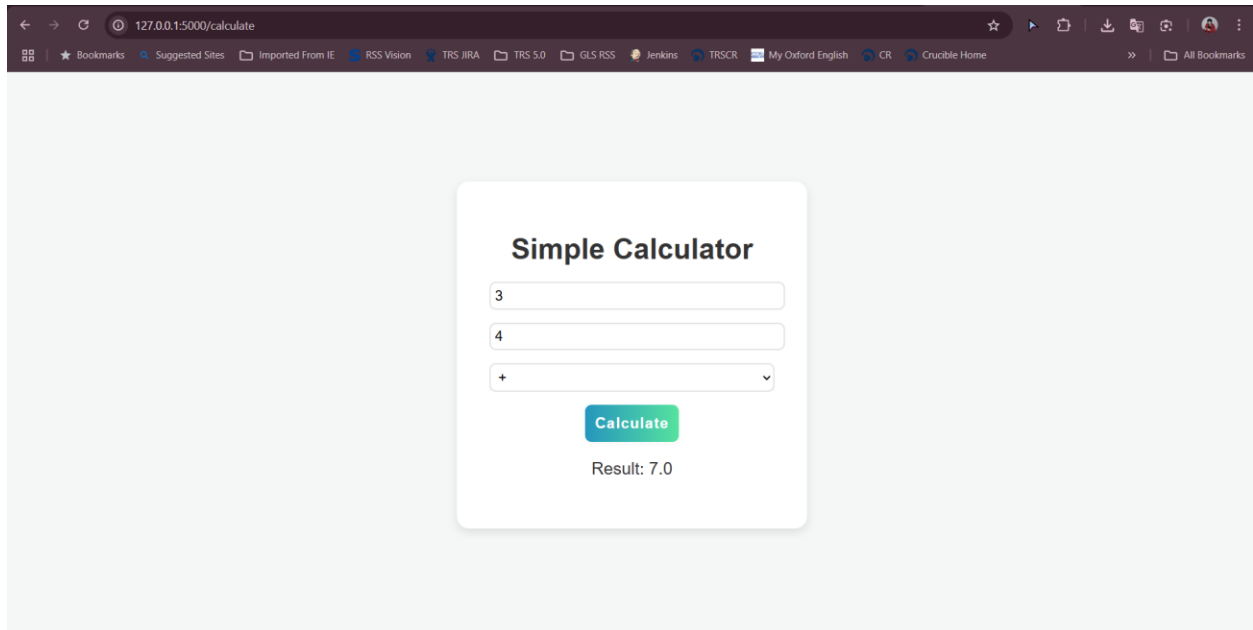
### 3.15 Exercise 15: Simple Calculator

Create a file named `calculator.py` for basic arithmetic. Sample folder structure:

`calculator.py`

`templates/ calculator.html`

Create a route `/calculate` with a form for two numbers and an operation (+, -, \*, /). Handle POST to compute and display the result. Handle division by zero with an error message.



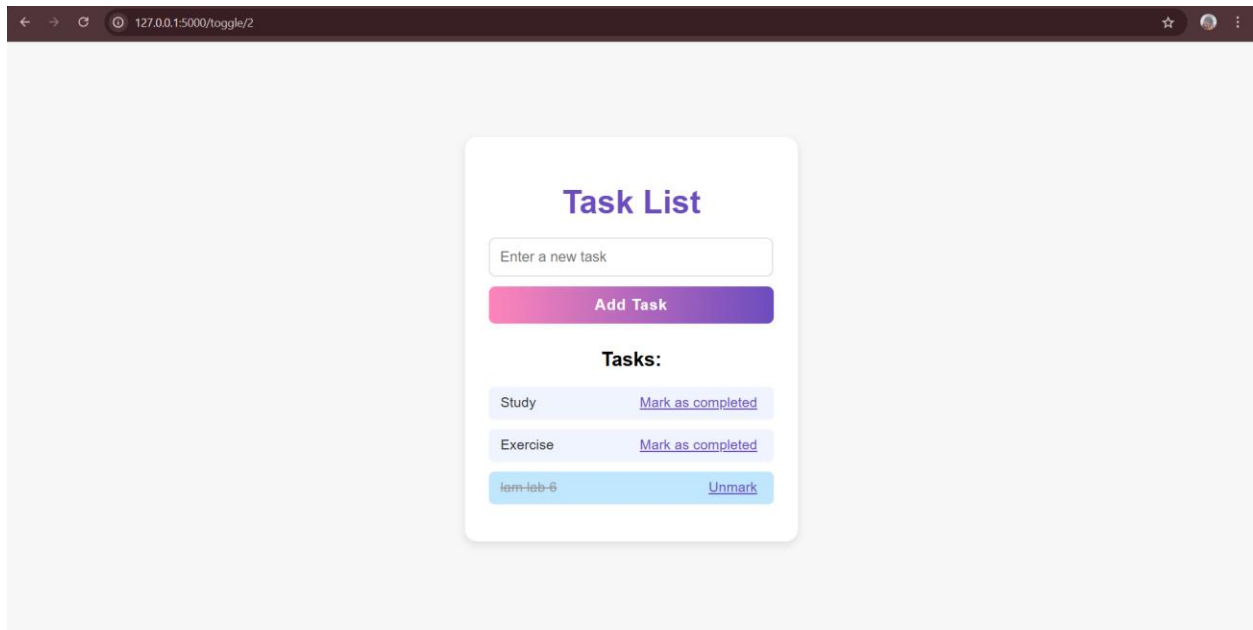
### 3.16 Exercise 16: To-Do List with Status

Create a file named `todolist.py` for a to-do list with completion status. Sample folder structure:

`todolist.py`

`templates/ todolist.html`

Create a route `/todo` to display tasks with a "completed" status. Include a form to add tasks and a button to toggle completion. Store tasks as a list of dictionaries (e.g., [{"task": "Study", "completed": False}]).



### 3.17 Exercise 17: Dynamic Menu

Create a file named `menu.py` to display a restaurant menu. Sample folder structure:

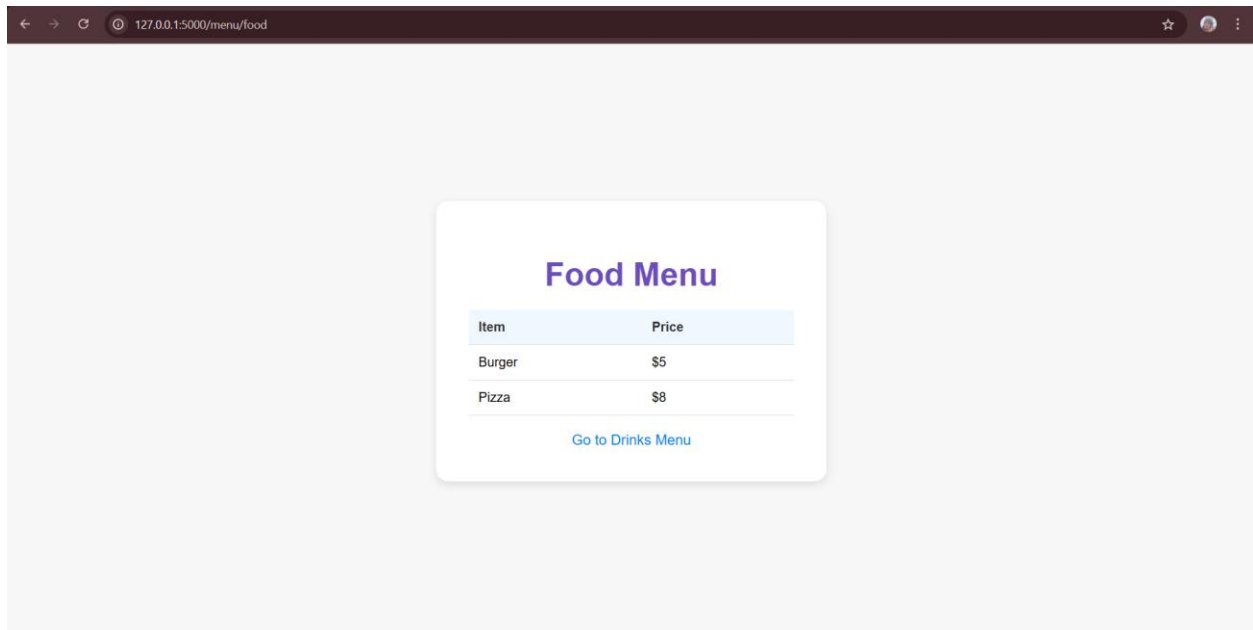
`menu.py`

`templates/ menu.html`

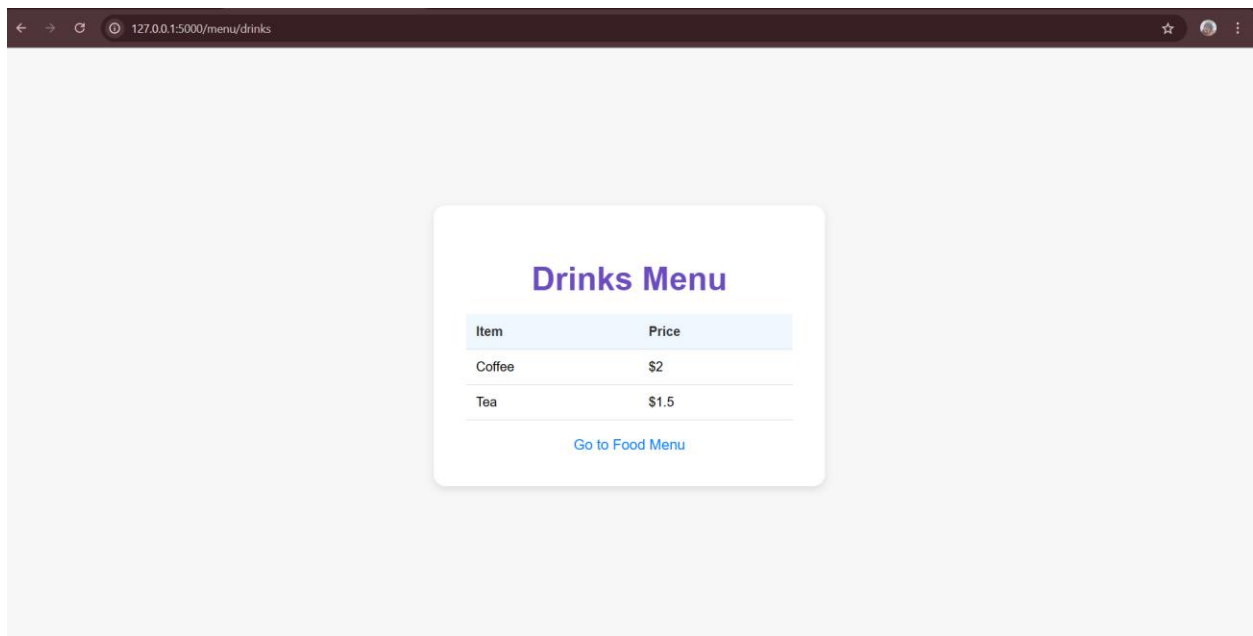
Create a route `/menu/<category>` for categories (e.g., "drinks", "food"). Use a dictionary for items (e.g., `{"drinks": [{"name": "Coffee", "price": 2}]}`). Render `menu.html` with items for the category. Return 404 for invalid categories.

- Food Menu





- Drinks Menu



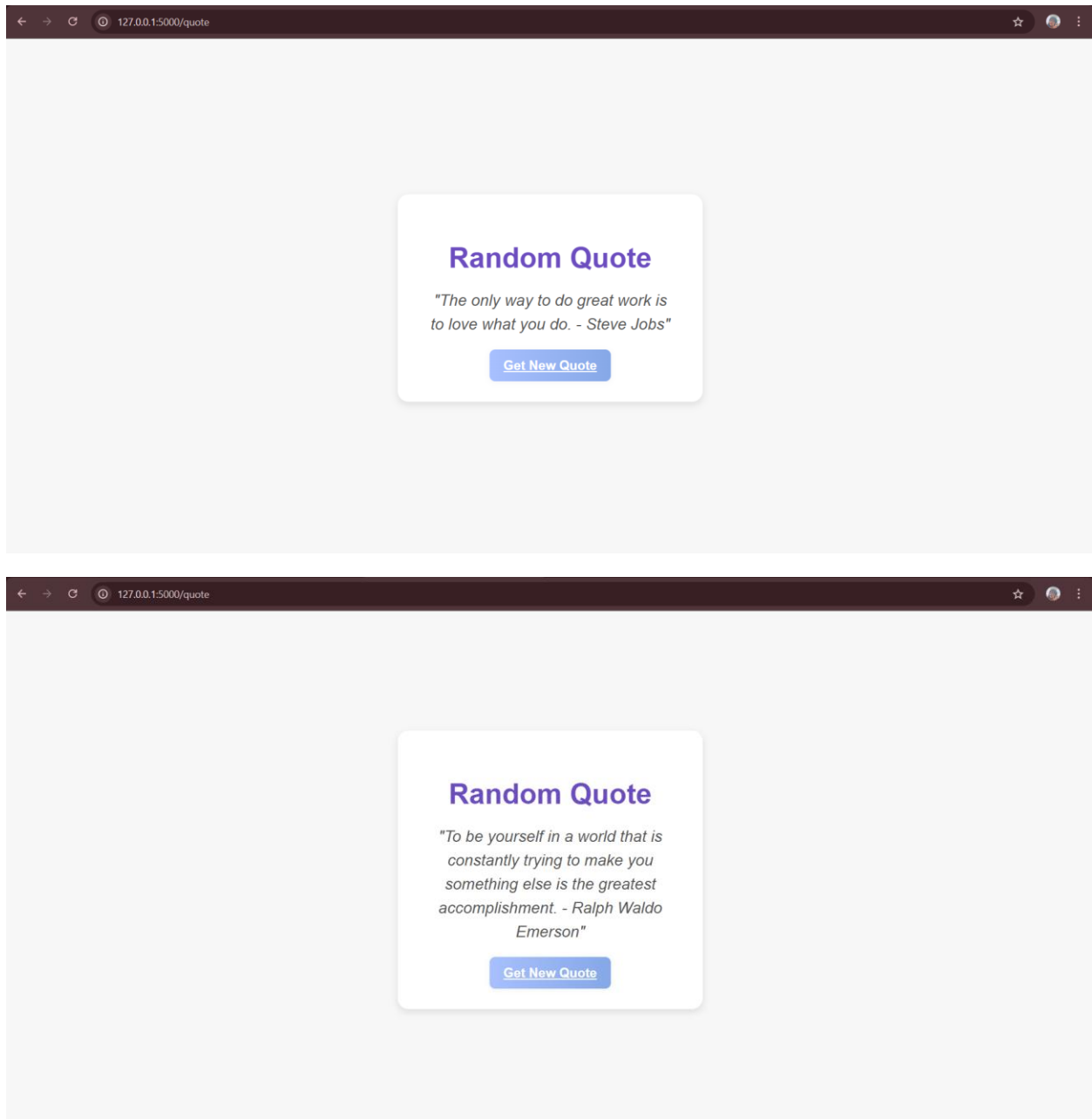
### 3.18 Exercise 18: Random Quote Generator

Create a file named `quote.py` to display random quotes. Sample folder structure:

`quote.py`

`templates/ quote.html`

Create a route /quote to display a random quote from a list. Use the random module to select a quote. Render quote.html with the quote and a "Get New Quote" button.



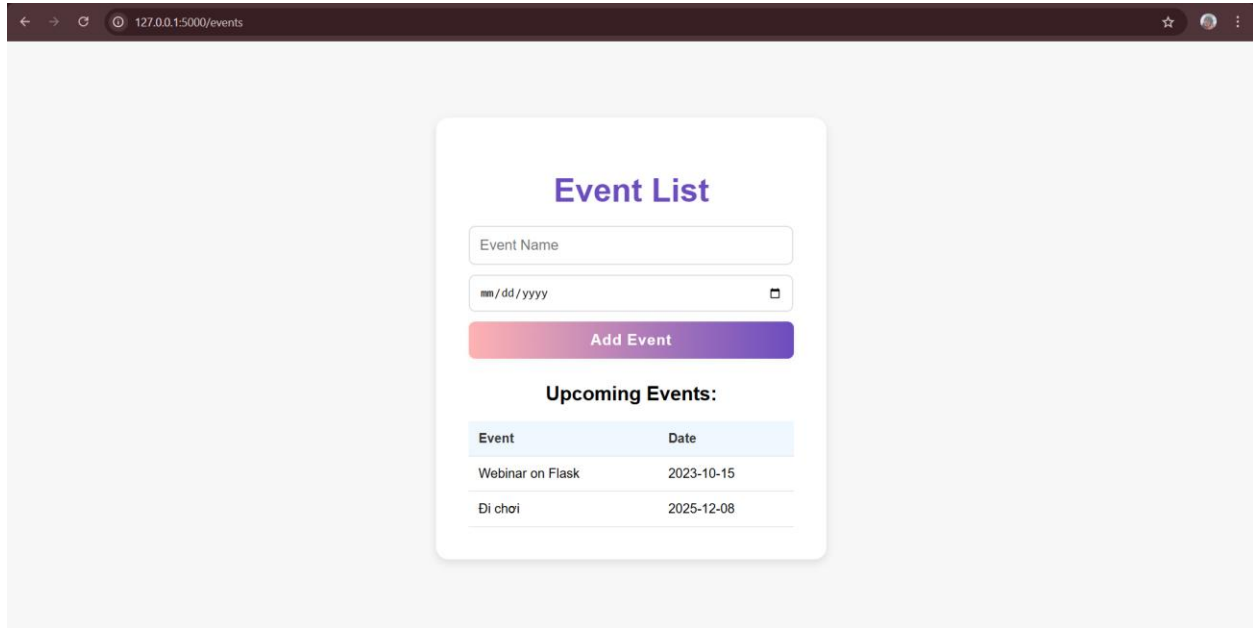
### 3.19 Exercise 19: Event List

Create a file named events.py for an event listing page. Sample folder structure:

events.py

templates/ events.html

Create a route /events to display events (name, date) in a table. Include a form to add events via POST. Store events in a list and render in events.html.



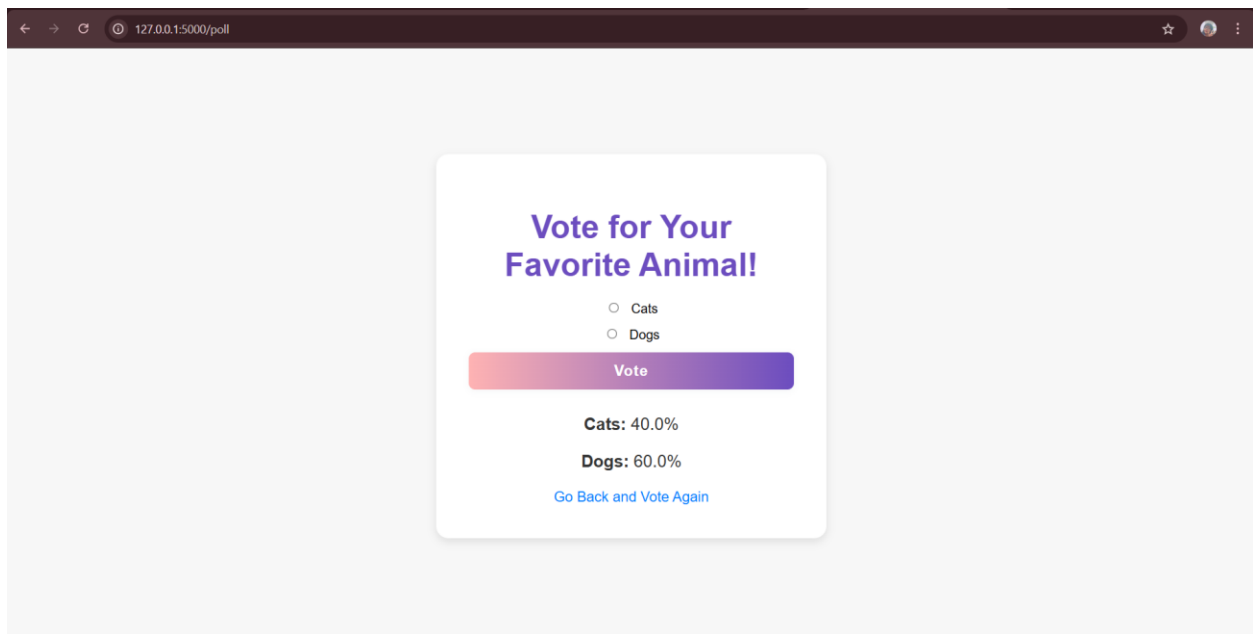
### 3.20 Exercise 20: Simple Poll

Create a file named poll.py for a voting poll. Sample folder structure:

poll.py

templates/ poll.html

Create a route /poll with a form to vote for options (e.g., "Cats", "Dogs"). Store votes in a dictionary and display results as a percentage. Render poll.html with the form and results.



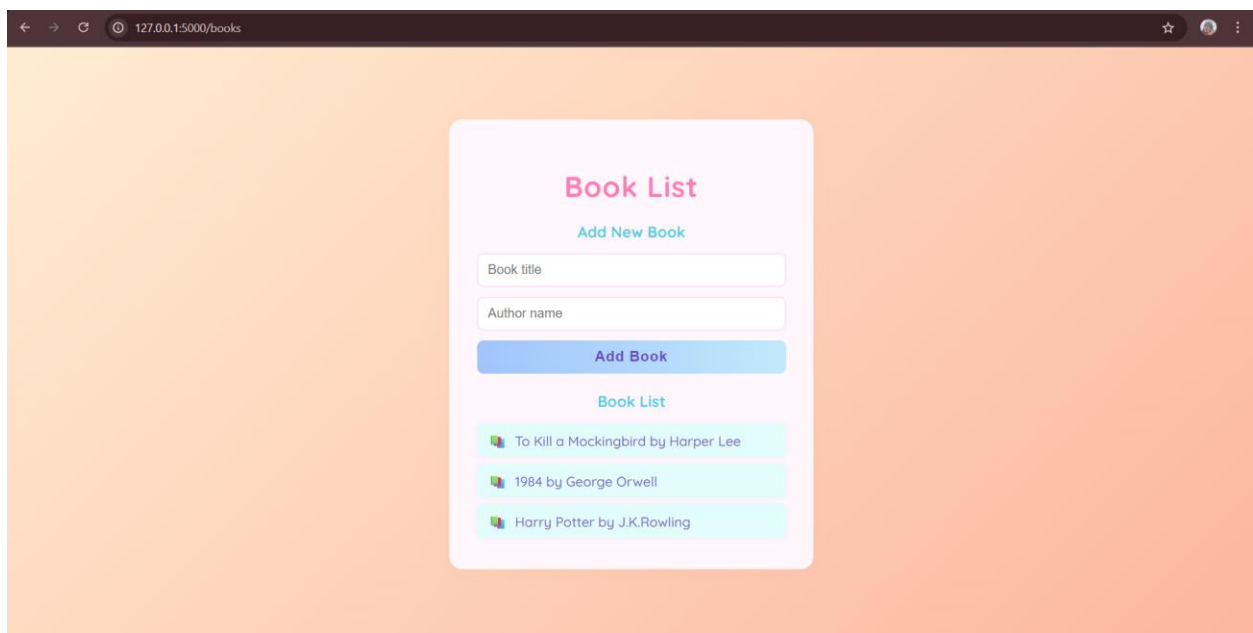
### 3.21 Exercise 21: Book List

Create a file named `books.py` to display a list of books. Sample folder structure:

`books.py`

`templates/ books.html`

Create a route `/books` to display books (title, author) in a list. Include a form to add books via POST. Store books in a list and render in `books.html`.



### 3.22 Exercise 22: Dynamic FAQ

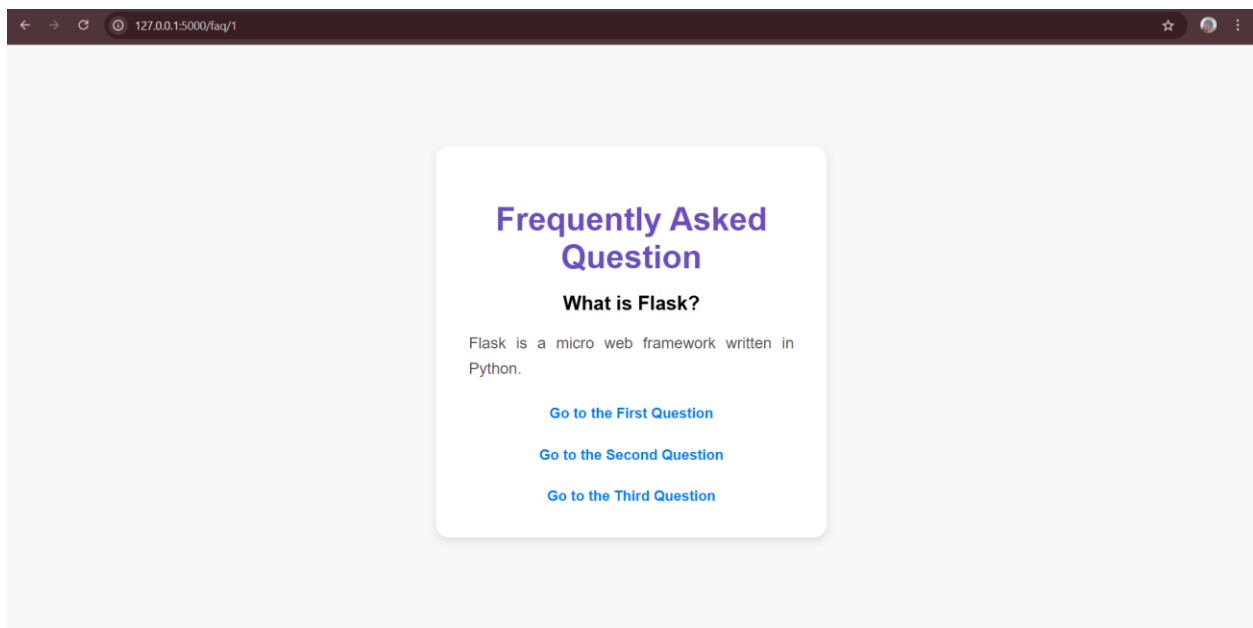
Create a file named `faq.py` for a FAQ page. Sample folder structure:

`faq.py`

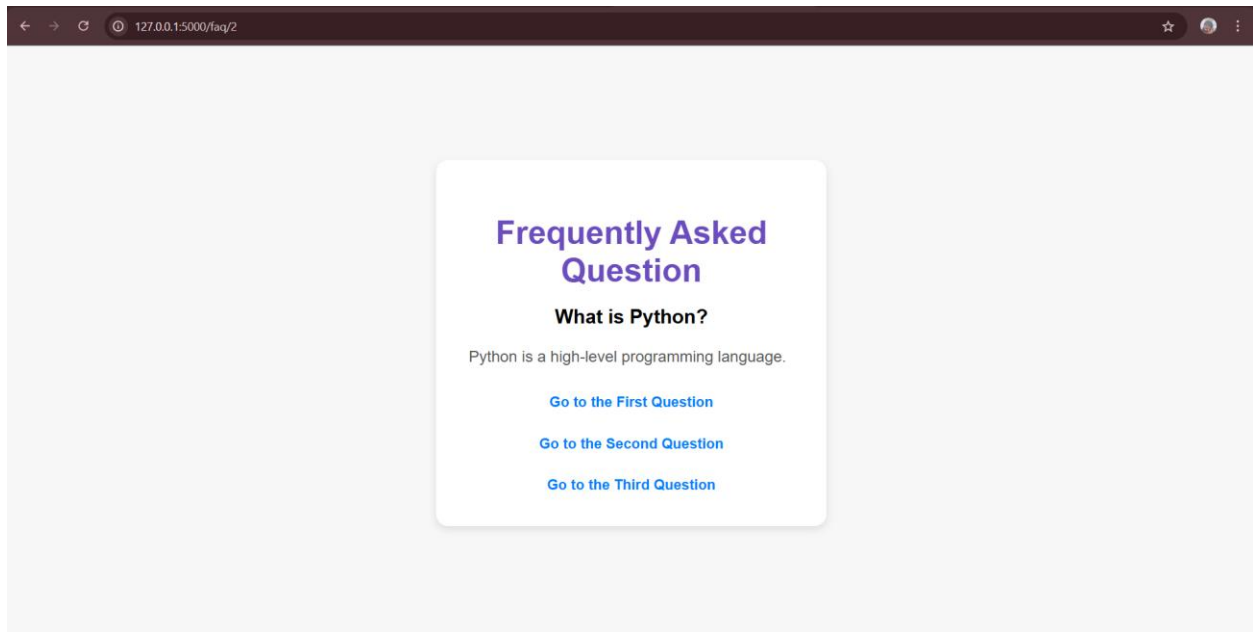
`templates/faq.html`

Create a route `/faq/<int:question_id>` for FAQ entries. Use a dictionary for FAQs (e.g., `{1: {"question": "What is Flask?", "answer": "A web framework"}}`). Render `faq.html` with the question and answer. Return 404 for invalid IDs.

- FAQ – Question 1:



- FAQ – Question 2:



- FAQ – Question 3:

