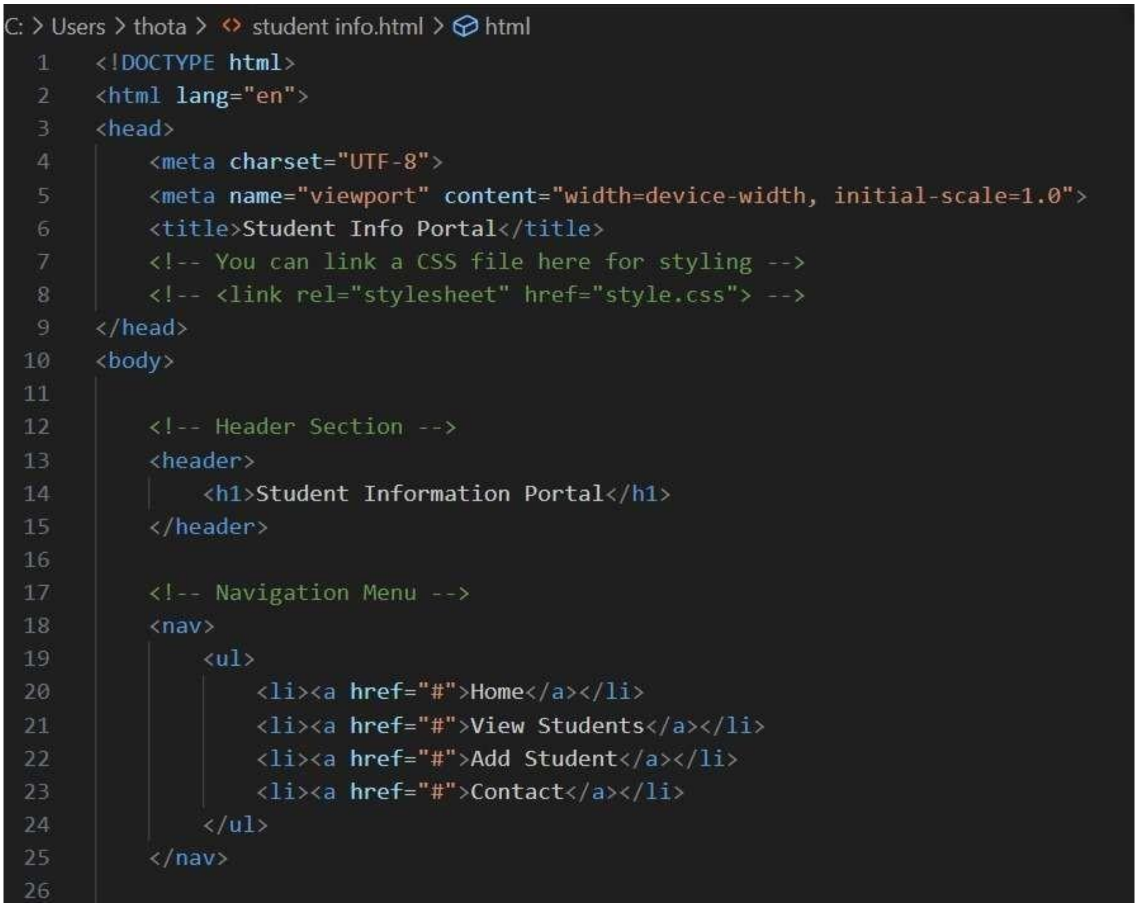
LAB ASSIGNMENT 14.3

Program : B. Tech (CSE)
Specialization : AIML
Course Title : AI Assisted coding
Semester : III
Academic Session : 2025-2026
Name of Student : Vangapandla Koushik
Enrollment No : 2403a52004
Batch No. : 01
Date : 28-10-2025

Task Description #1 – AI-generated HTML Page Task: Ask AI to generate a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer

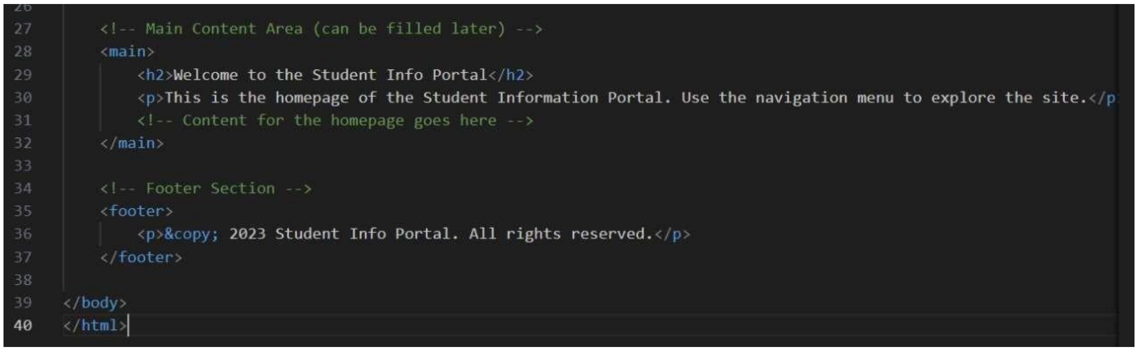
Expected Output: • HTML code with , , . • Clean indentation, proper tags, and comments

Prompt: Create a basic HTML layout for a "Student Info Portal" homepage.
The structure should include a header section, a navigation menu, and a footer. Ensure the code is well-formatted with clean indentation, appropriate HTML tags, and helpful comments for clarity.



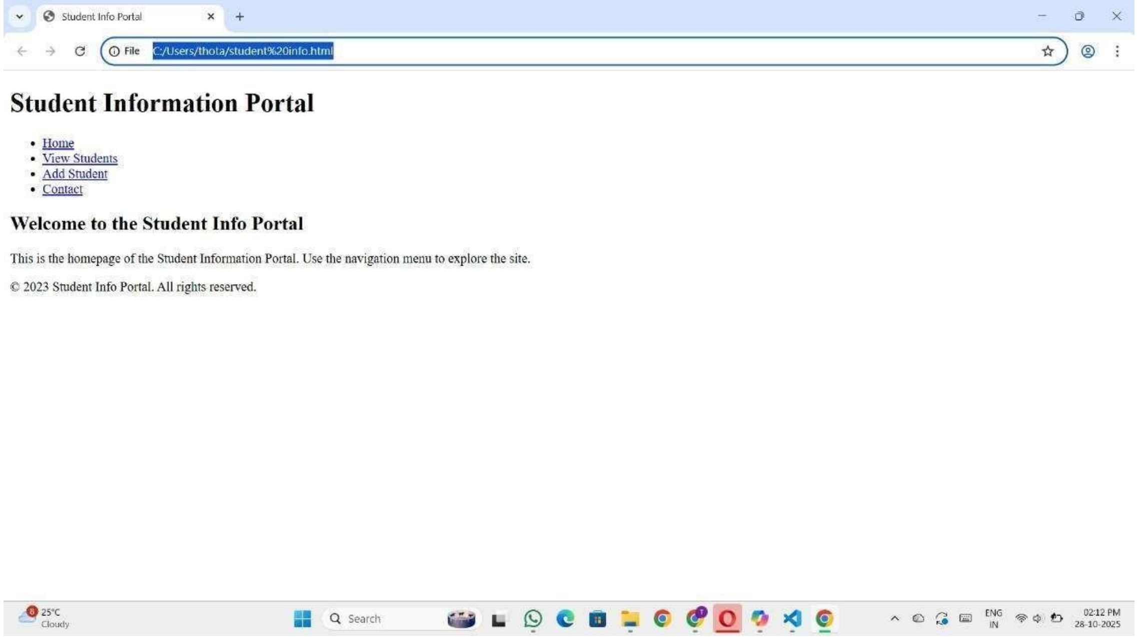
C: > Users > thota > student info.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Info Portal</title>
    <!-- You can link a CSS file here for styling -->
    <!-- <link rel="stylesheet" href="style.css"> -->
</head>
<body>

    <!-- Header Section -->
    <header>
        <h1>Student Information Portal</h1>
    </header>

    <!-- Navigation Menu -->
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">View Students</a></li>
            <li><a href="#">Add Student</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>

```



```html
    <!-- Main Content Area (can be filled later) -->
    <main>
        <h2>Welcome to the Student Info Portal</h2>
        <p>This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.</p
        <!-- Content for the homepage goes here -->
    </main>

    <!-- Footer Section -->
    <footer>
        <p>&copy; 2023 Student Info Portal. All rights reserved.</p>
    </footer>

</body>
</html>
```

Output:



Student Information Portal

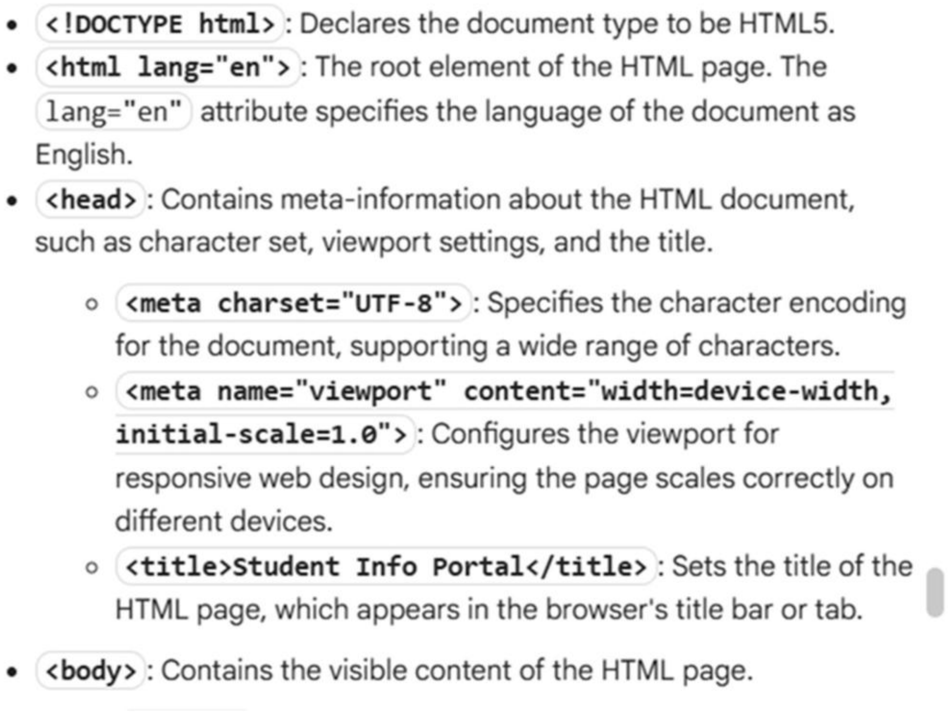
- Home
- View Students
- Add Student
- Contact

Welcome to the Student Info Portal

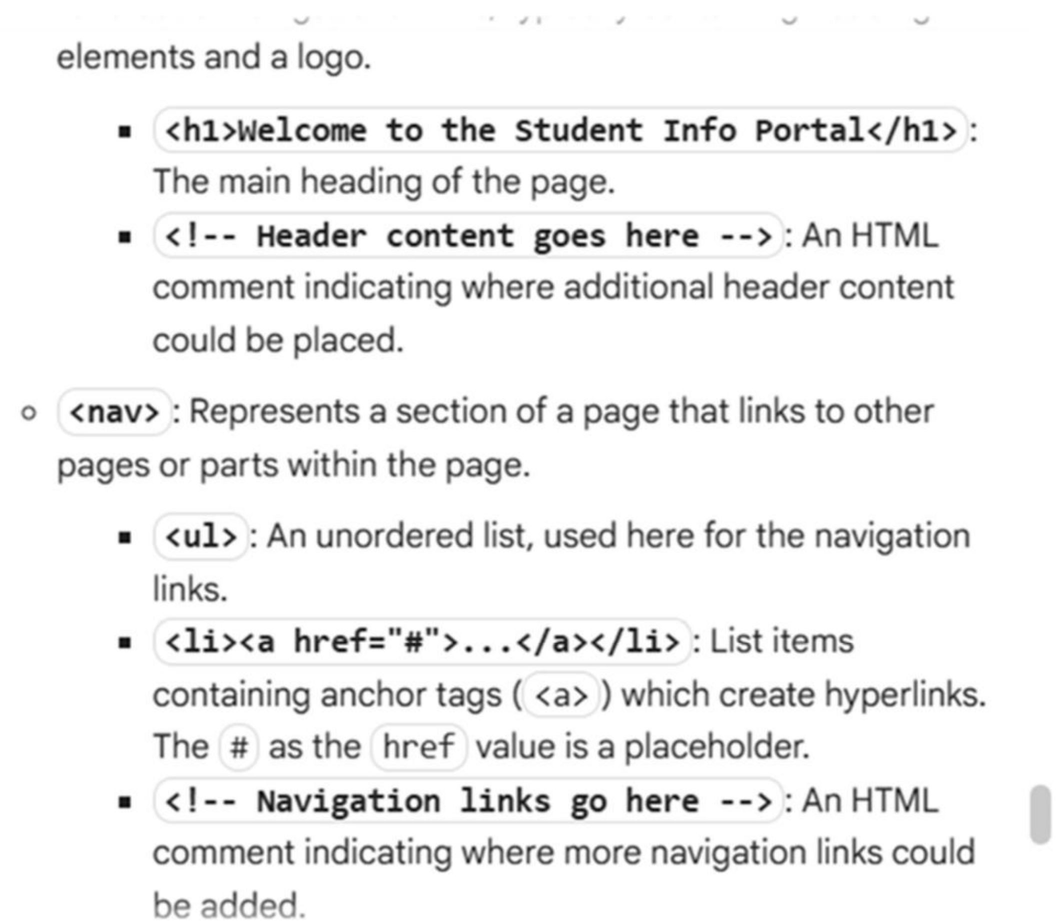
This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.

© 2023 Student Info Portal. All rights reserved.

Explanation of code:



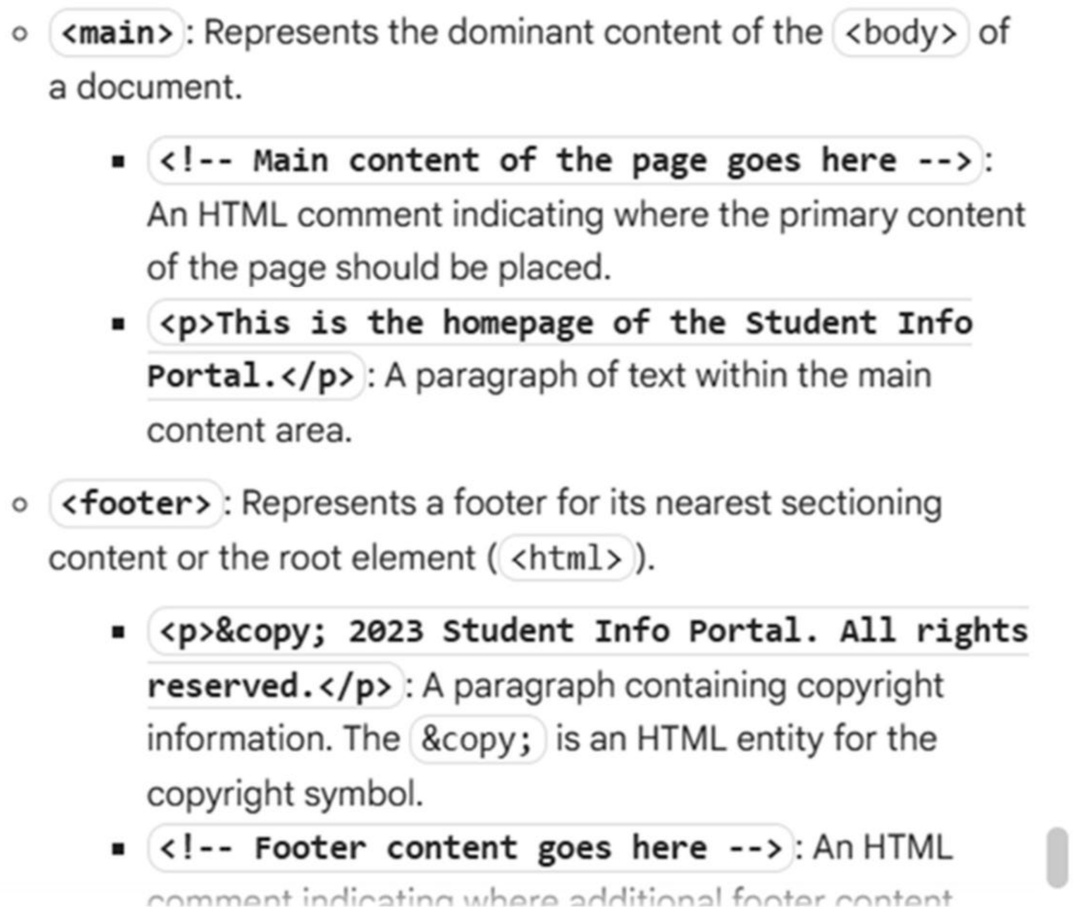
- `<!DOCTYPE html>`: Declares the document type to be HTML5.
- `<html lang="en">`: The root element of the HTML page. The `lang="en"` attribute specifies the language of the document as English.
- `<head>`: Contains meta-information about the HTML document, such as character set, viewport settings, and the title.
  - `<meta charset="UTF-8">`: Specifies the character encoding for the document, supporting a wide range of characters.
  - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Configures the viewport for responsive web design, ensuring the page scales correctly on different devices.
  - `<title>Student Info Portal</title>`: Sets the title of the HTML page, which appears in the browser's title bar or tab.
- `<body>`: Contains the visible content of the HTML page.



elements and a logo.

- `<h1>Welcome to the Student Info Portal</h1>`: The main heading of the page.
- `<!-- Header content goes here -->`: An HTML comment indicating where additional header content could be placed.

- `<nav>`: Represents a section of a page that links to other pages or parts within the page.
  - `<ul>`: An unordered list, used here for the navigation links.
  - `<li><a href="#">...</a></li>`: List items containing anchor tags (`<a>`) which create hyperlinks. The `#` as the `href` value is a placeholder.
  - `<!-- Navigation links go here -->`: An HTML comment indicating where more navigation links could be added.



- `<main>`: Represents the dominant content of the `<body>` of a document.
    - `<!-- Main content of the page goes here -->`: An HTML comment indicating where the primary content of the page should be placed.
    - `<p>This is the homepage of the Student Info Portal.</p>`: A paragraph of text within the main content area.
- `<footer>`: Represents a footer for its nearest sectioning content or the root element (`<html>`).
    - `<p>&copy; 2023 Student Info Portal. All rights reserved.</p>`: A paragraph containing copyright information. The `&copy;` is an HTML entity for the copyright symbol.
    - `<!-- Footer content goes here -->`: An HTML comment indicating where additional footer content

Task Description #2 – CSS Styling

Task: Use AI to add CSS styling to Task #1 homepage for: • Responsive navigation bar. • Centered content section. • Footer with light gray background.

Expected Output: • HTML + CSS combined. • AI explains how CSS classes apply Expected Output: AI refactors with with open() and try-except:

Prompt:

Develop the "Student Info Portal" homepage by integrating CSS styling directly into the HTML. The design should include:

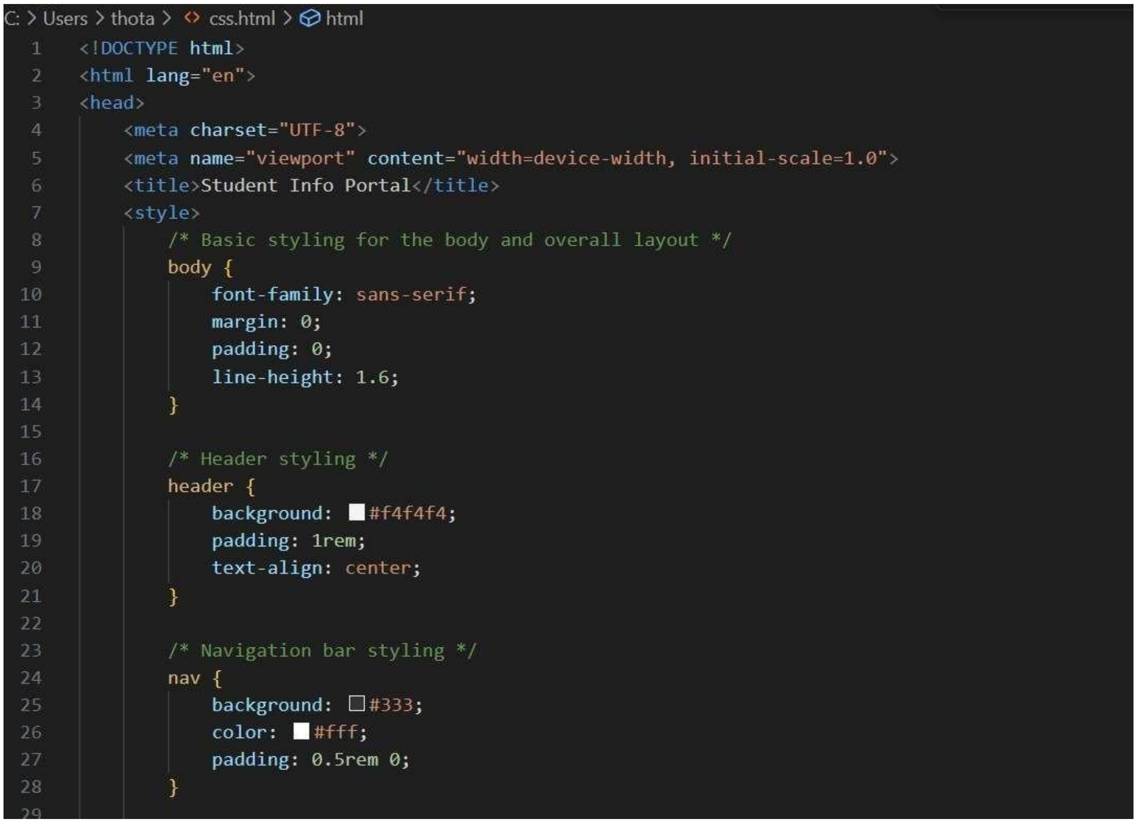
- A responsive navigation bar that adapts to different screen sizes

- A content section that is centered on the page

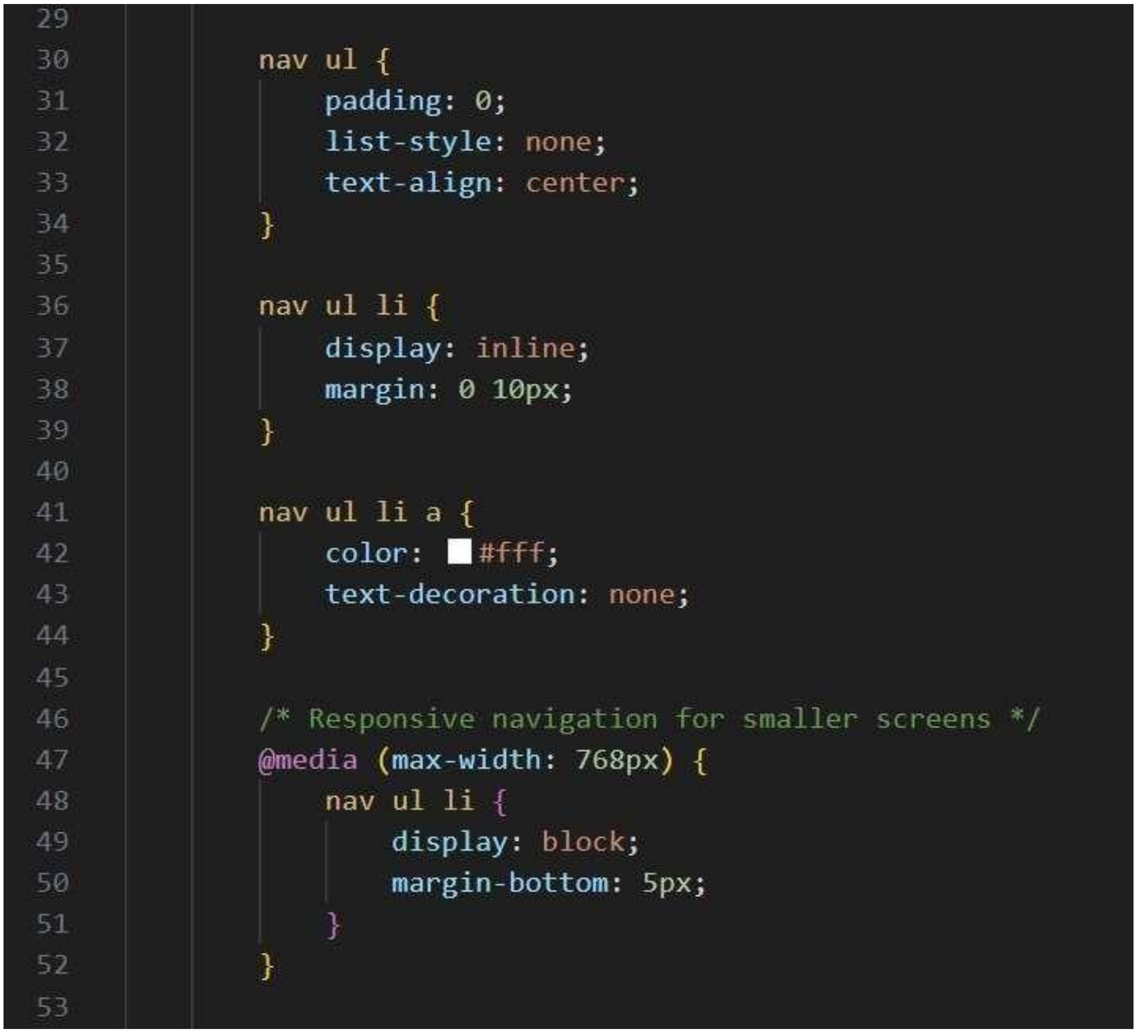
- A footer styled with a light gray background

Combine the HTML and CSS in a single file, and include comments or explanations that clarify how the CSS classes are applied to structure and style the layout.

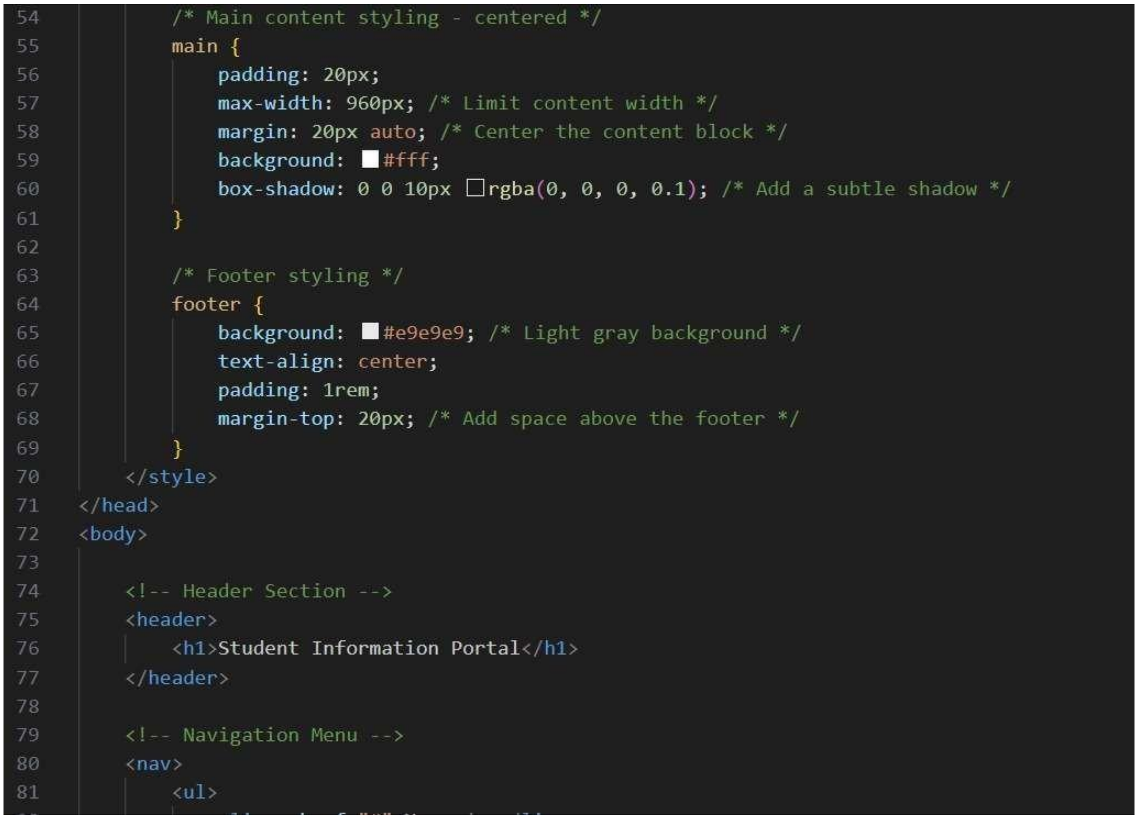
Code:



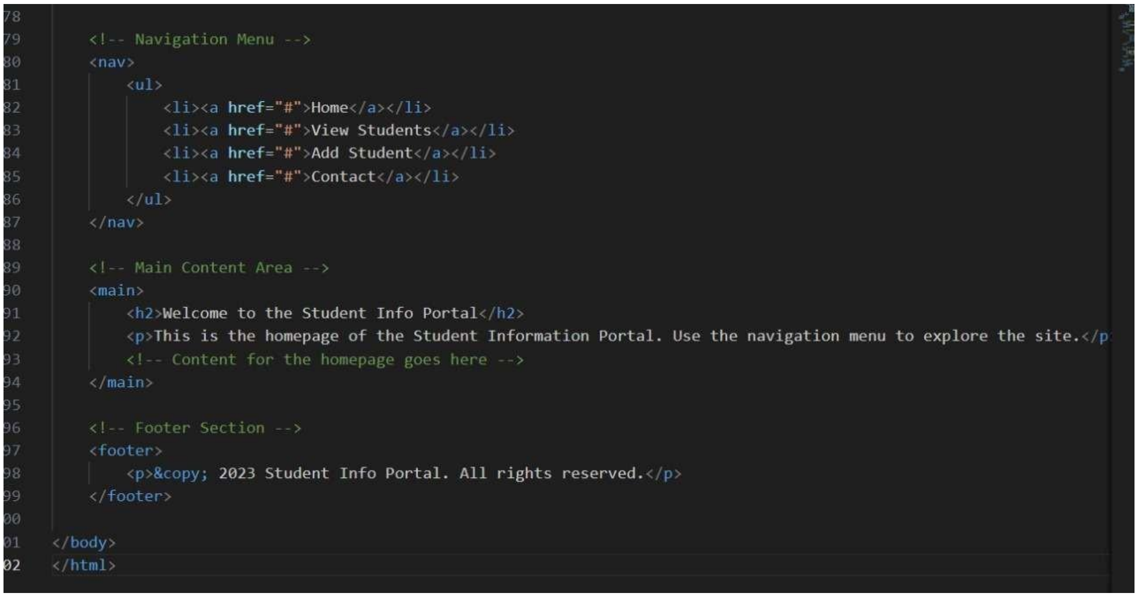
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Info Portal</title>
    <style>
        /* Basic styling for the body and overall layout */
        body {
            font-family: sans-serif;
            margin: 0;
            padding: 0;
            line-height: 1.6;
        }

        /* Header styling */
        header {
            background: #f4f4f4;
            padding: 1rem;
            text-align: center;
        }

        /* Navigation bar styling */
        nav {
            background: #333;
            color: #fff;
            padding: 0.5rem 0;
        }
```



```css

        nav ul {
            padding: 0;
            list-style: none;
            text-align: center;
        }

        nav ul li {
            display: inline;
            margin: 0 10px;
        }

        nav ul li a {
            color: #fff;
            text-decoration: none;
        }

        /* Responsive navigation for smaller screens */
        @media (max-width: 768px) {
            nav ul li {
                display: block;
                margin-bottom: 5px;
            }
        }

```

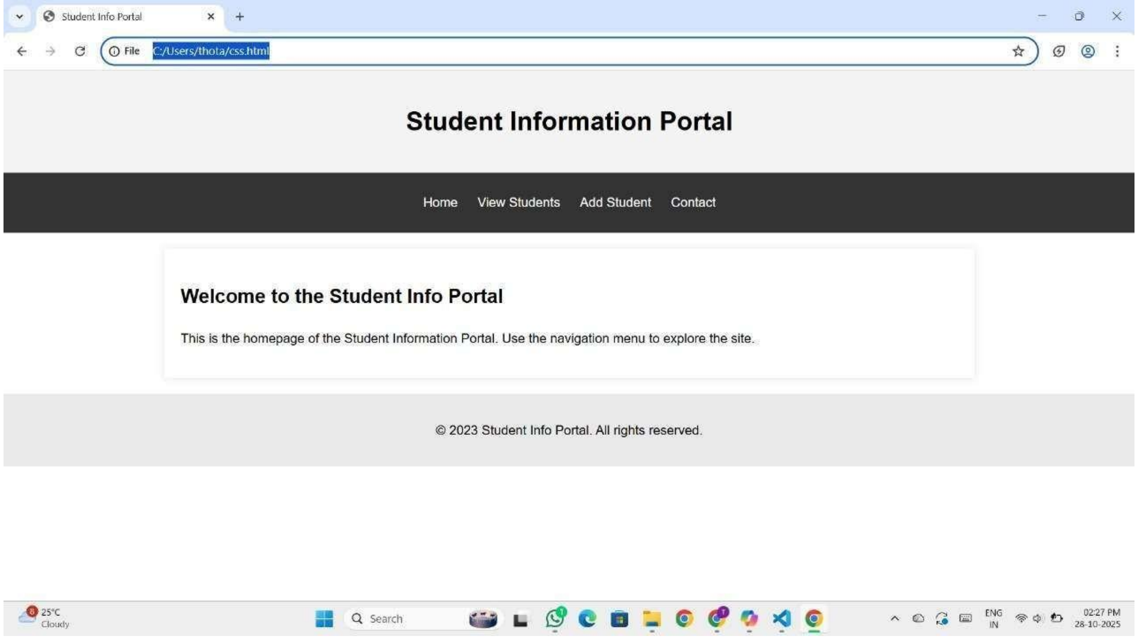


```
        /* Main content styling - centered */
        main {
            padding: 20px;
            max-width: 960px; /* Limit content width */
            margin: 20px auto; /* Center the content block */
            background: #fff;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
        }

        /* Footer styling */
        footer {
            background: #e9e9e9; /* Light gray background */
            text-align: center;
            padding: 1rem;
            margin-top: 20px; /* Add space above the footer */
        }
    </style>
</head>
<body>

    <!-- Header Section -->
    <header>
        <h1>Student Information Portal</h1>
    </header>

    <!-- Navigation Menu -->
    <nav>
        <ul>
```



```

    <!-- Navigation Menu -->
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">View Students</a></li>
            <li><a href="#">Add Student</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>

    <!-- Main Content Area -->
    <main>
        <h2>Welcome to the Student Info Portal</h2>
        <p>This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.</p
        <!-- Content for the homepage goes here -->
    </main>

    <!-- Footer Section -->
    <footer>
        <p>&copy; 2023 Student Info Portal. All rights reserved.</p>
    </footer>

</body>
</html>
```

Output:



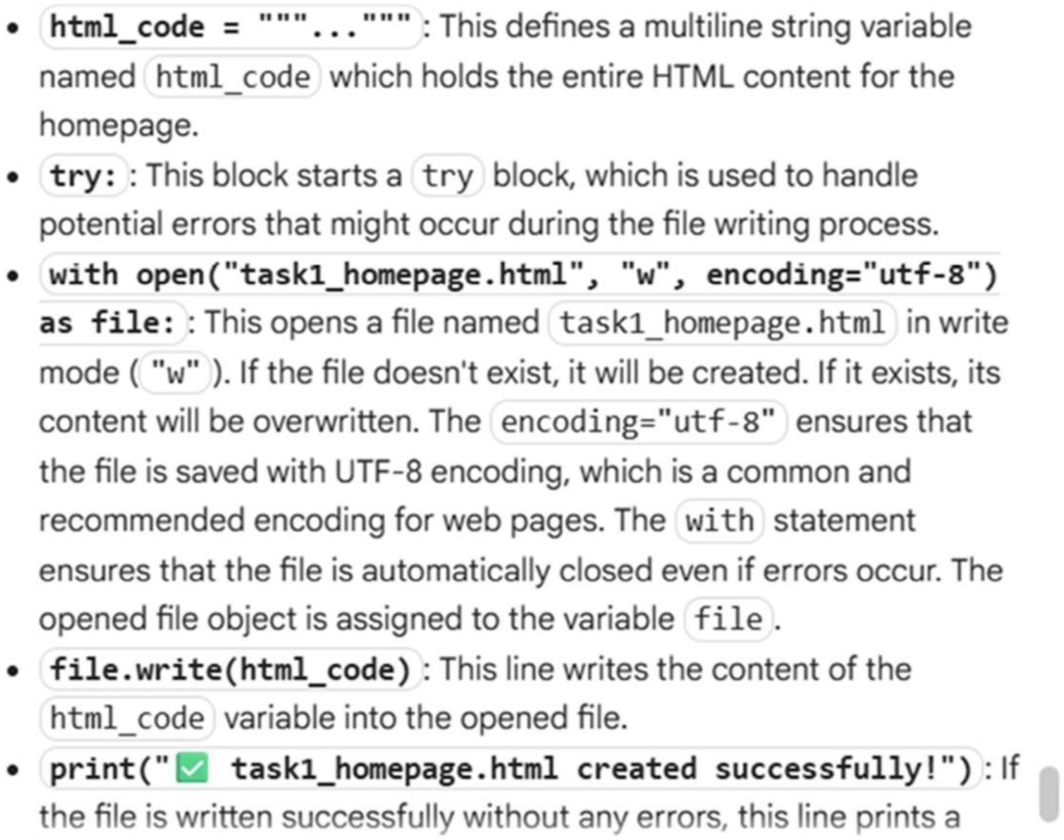
Student Information Portal

Welcome to the Student Info Portal

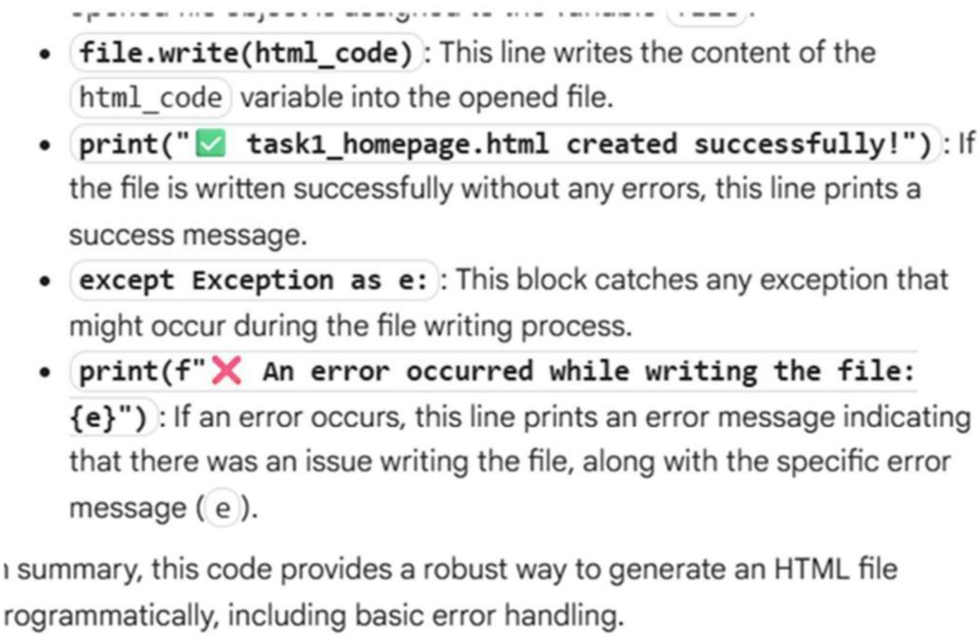
This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.

© 2023 Student Info Portal. All rights reserved.

Explanation of code:



- **`html_code = """..."""`**: This defines a multiline string variable named `html_code` which holds the entire HTML content for the homepage.
- **`try:`**: This block starts a `try` block, which is used to handle potential errors that might occur during the file writing process.
- **`with open("task1_homepage.html", "w", encoding="utf-8") as file:`**: This opens a file named `task1_homepage.html` in write mode (`"w"`). If the file doesn't exist, it will be created. If it exists, its content will be overwritten. The `encoding="utf-8"` ensures that the file is saved with UTF-8 encoding, which is a common and recommended encoding for web pages. The `with` statement ensures that the file is automatically closed even if errors occur. The opened file object is assigned to the variable `file`.
- **`file.write(html_code)`**: This line writes the content of the `html_code` variable into the opened file.
- **`print("✅ task1_homepage.html created successfully!")`**: If the file is written successfully without any errors, this line prints a



- `file.write(html_code)`: This line writes the content of the `html_code` variable into the opened file.
- `print("✅ task1_homepage.html created successfully!")`: If the file is written successfully without any errors, this line prints a success message.
- `except Exception as e:`: This block catches any exception that might occur during the file writing process.
- `print(f"❌ An error occurred while writing the file: {e}")`: If an error occurs, this line prints an error message indicating that there was an issue writing the file, along with the specific error message (`e`).

summary, this code provides a robust way to generate an HTML file rogrammatically, including basic error handling.

Task Description #3 – JavaScript Interactivity

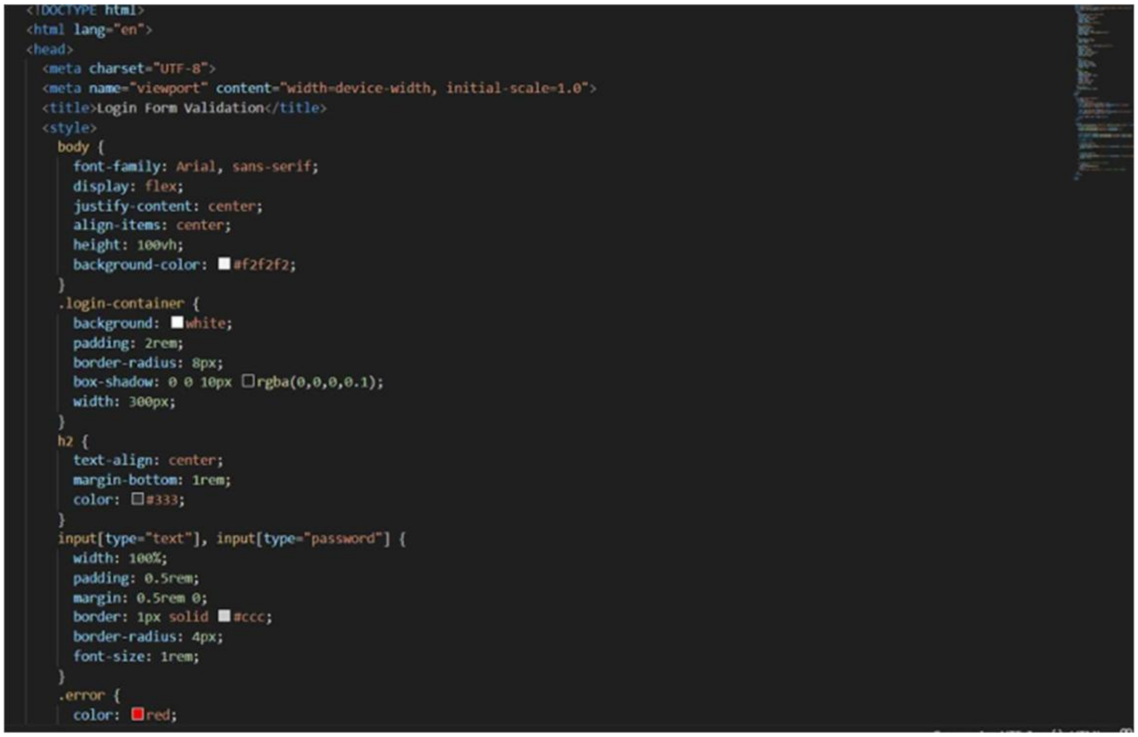
Task: Prompt AI to generate a JS script that validates a simple login form (non-empty username/password)

Expected Output: Working on submit JS valida on. Clear error messages if inputs are empty.

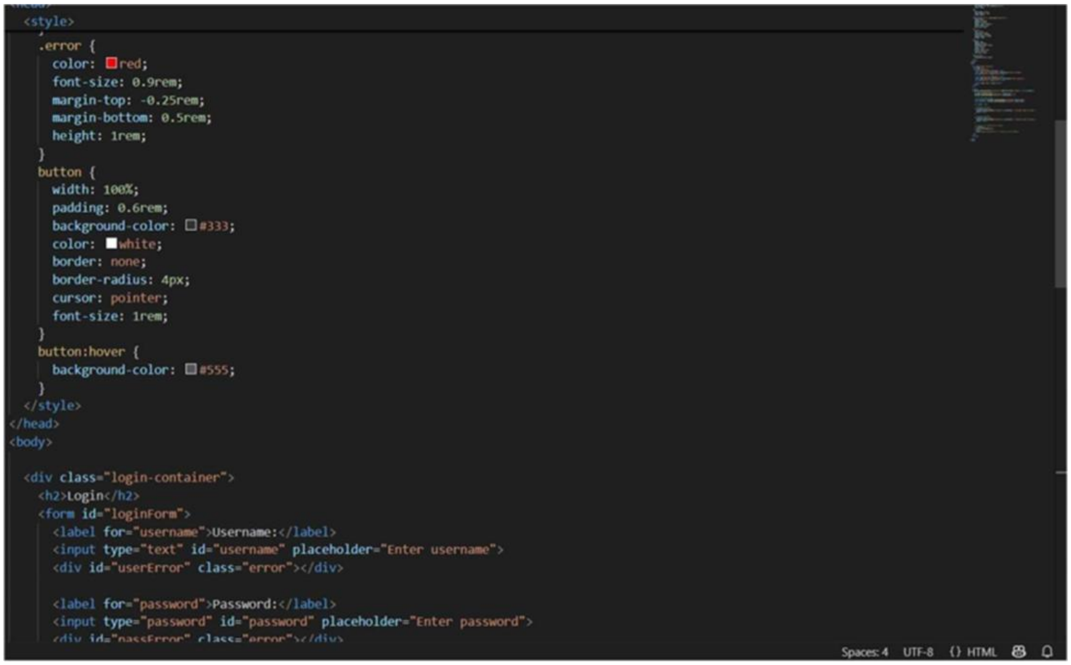
Prompt:

Develop a code to generate Task: Prompt AI to generate a JS script that validates a simple login form (non-empty username/password). Expected Output: Working on submit JS valida on. Clear error messages if inputs are empty.

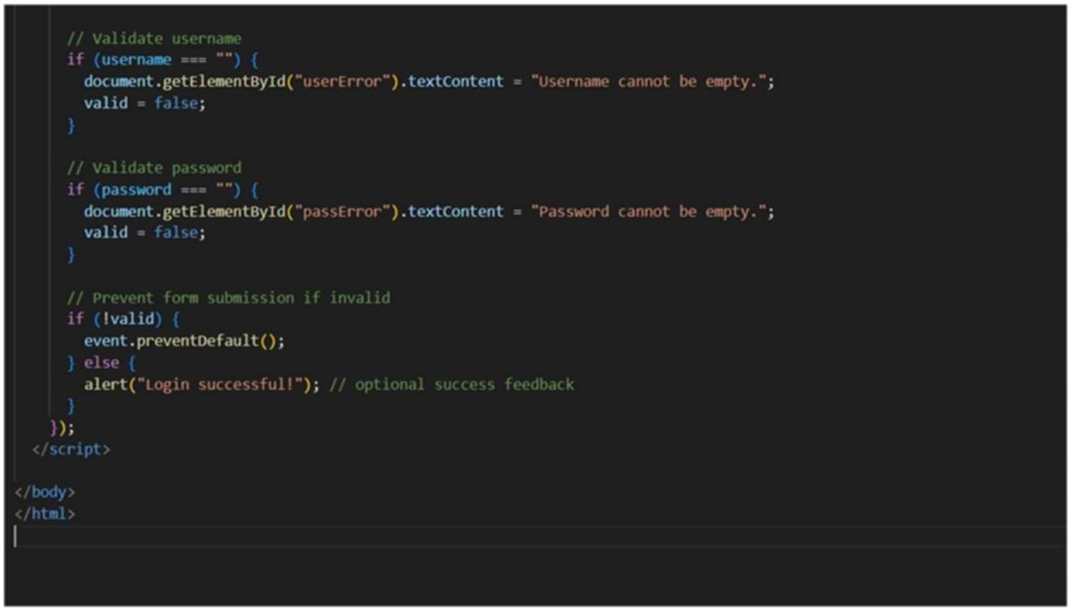
Code:



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Form Validation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f2f2f2;
    }
    .login-container {
      background: white;
      padding: 2rem;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      width: 300px;
    }
    h2 {
      text-align: center;
      margin-bottom: 1rem;
      color: #333;
    }
    input[type="text"], input[type="password"] {
      width: 100%;
      padding: 0.5rem;
      margin: 0.5rem 0;
      border: 1px solid #ccc;
      border-radius: 4px;
      font-size: 1rem;
    }
    .error {
      color: red;
```

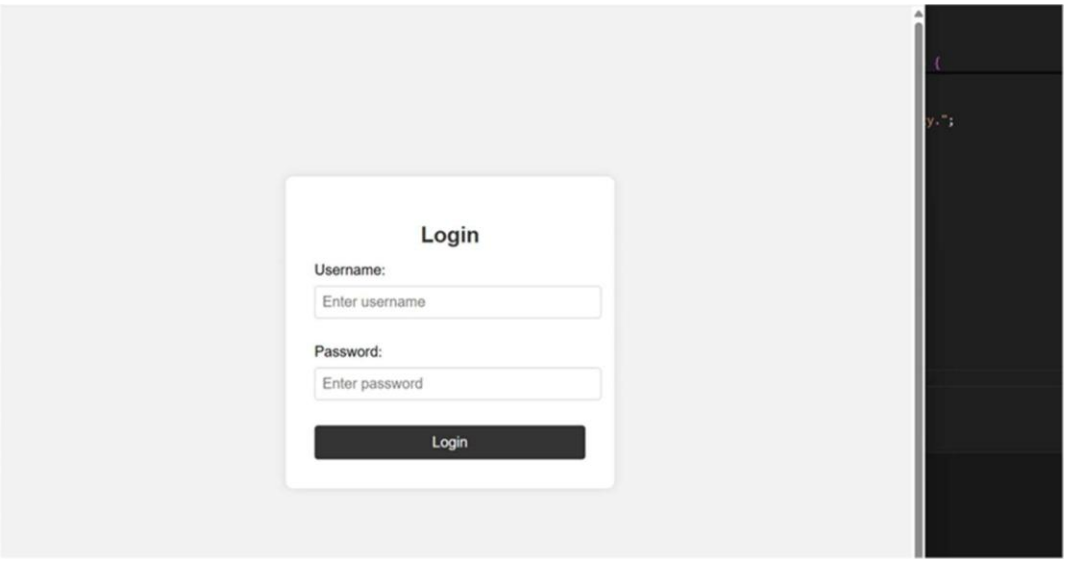


```html
  <style>
    .error {
      color: red;
      font-size: 0.9rem;
      margin-top: -0.25rem;
      margin-bottom: 0.5rem;
      height: 1rem;
    }
    button {
      width: 100%;
      padding: 0.6rem;
      background-color: #333;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 1rem;
    }
    button:hover {
      background-color: #555;
    }
  </style>
</head>
<body>

  <div class="login-container">
    <h2>Login</h2>
    <form id="loginForm">
      <label for="username">Username:</label>
      <input type="text" id="username" placeholder="Enter username">
      <div id="userError" class="error"></div>

      <label for="password">Password:</label>
      <input type="password" id="password" placeholder="Enter password">
      <div id="passError" class="error"></div>
```

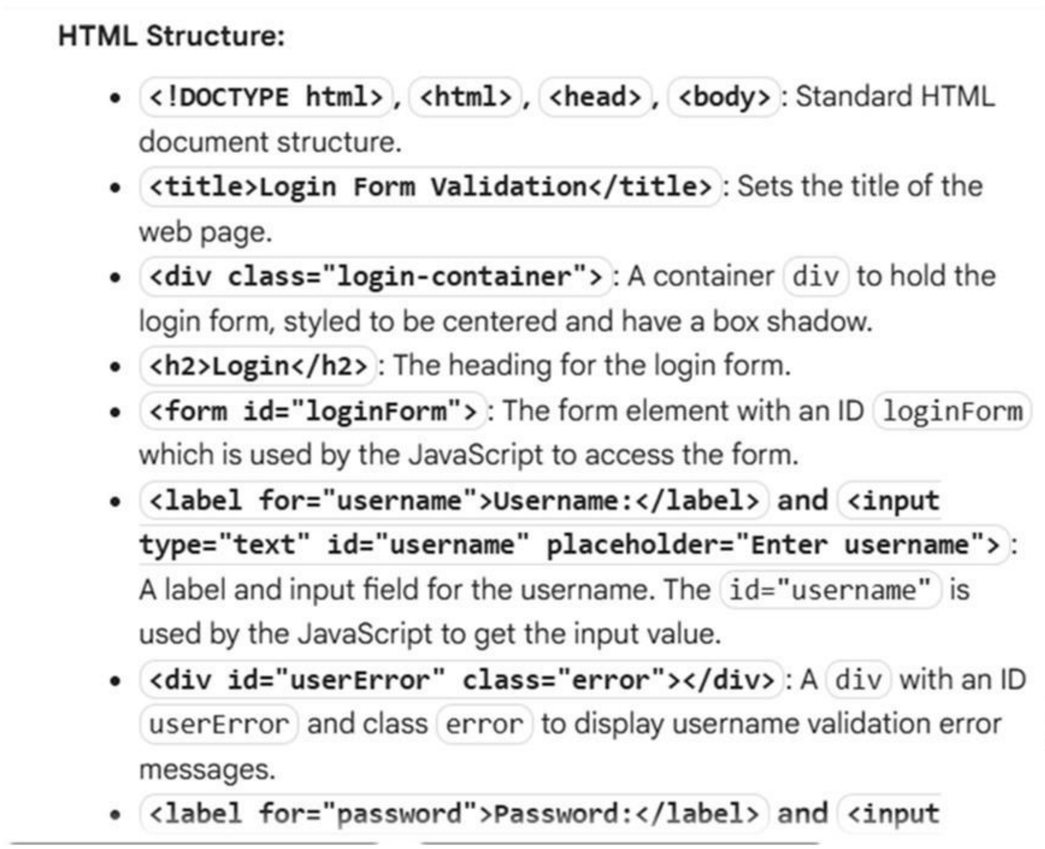


```
      // Validate username
      if (username === "") {
        document.getElementById("userError").textContent = "Username cannot be empty.";
        valid = false;
      }

      // Validate password
      if (password === "") {
        document.getElementById("passError").textContent = "Password cannot be empty.";
        valid = false;
      }

      // Prevent form submission if invalid
      if (!valid) {
        event.preventDefault();
      } else {
        alert("Login successful!"); // optional success feedback
      }
    });
  </script>

</body>
</html>
```

Output:

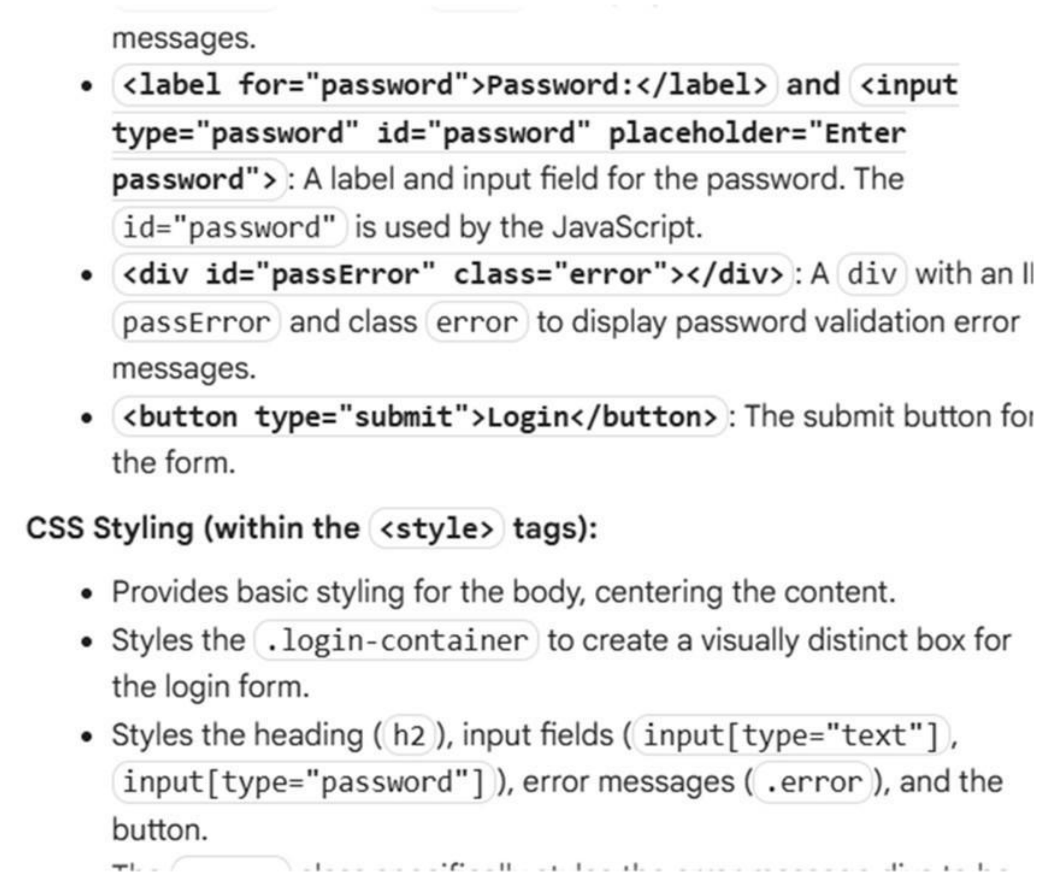


Explanation of code:



HTML Structure:

- `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`: Standard HTML document structure.
- `<title>Login Form Validation</title>`: Sets the title of the web page.
- `<div class="login-container">`: A container `div` to hold the login form, styled to be centered and have a box shadow.
- `<h2>Login</h2>`: The heading for the login form.
- `<form id="loginForm">`: The form element with an ID `loginForm` which is used by the JavaScript to access the form.
- `<label for="username">Username:</label>` and `<input type="text" id="username" placeholder="Enter username">`: A label and input field for the username. The `id="username"` is used by the JavaScript to get the input value.
- `<div id="userError" class="error"></div>`: A `div` with an ID `userError` and class `error` to display username validation error messages.
- `<label for="password">Password:</label>` and `<input`



messages.

- `<label for="password">Password:</label>` and `<input type="password" id="password" placeholder="Enter password">`: A label and input field for the password. The `id="password"` is used by the JavaScript.
- `<div id="passError" class="error"></div>`: A `div` with an I `passError` and class `error` to display password validation error messages.
- `<button type="submit">Login</button>`: The submit button fo the form.

CSS Styling (within the `<style>` tags):

- Provides basic styling for the body, centering the content.
- Styles the `.login-container` to create a visually distinct box for the login form.
- Styles the heading (`h2`), input fields (`input[type="text"]`, `input[type="password"]`), error messages (`.error`), and the button.

red and have a fixed height to prevent layout shifts.

JavaScript (within the `<script>` tags):

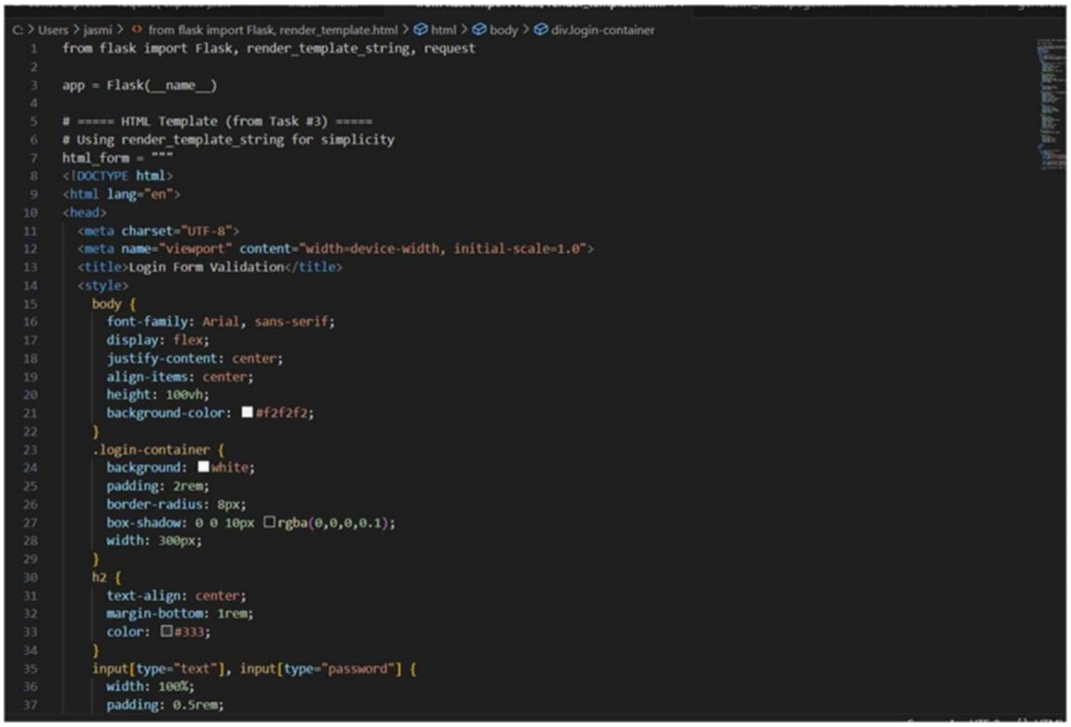
- `document.getElementById("loginForm").addEventListener("submit", function(event) { ... });`: This attaches an event listener to the form with the ID `loginForm`. When the form is submitted, the function inside the event listener is executed. The `event` object is passed to the function.
- `document.getElementById("userError").textContent = "";` and `document.getElementById("passError").textContent = "";`: These lines clear any previously displayed error messages when the form is submitted again.
- `const username = document.getElementById("username").value.trim();` and `const password = document.getElementById("password").value.trim();`: These lines get the values entered in the username and password input fields, and the `.trim()` method removes any leading or trailing whitespace.

Task Description #4 – Python Backend Integration Task: Ask AI to generate a Flask app that serves the HTML form (Task #3) and prints the username on successful login.

Prompt:

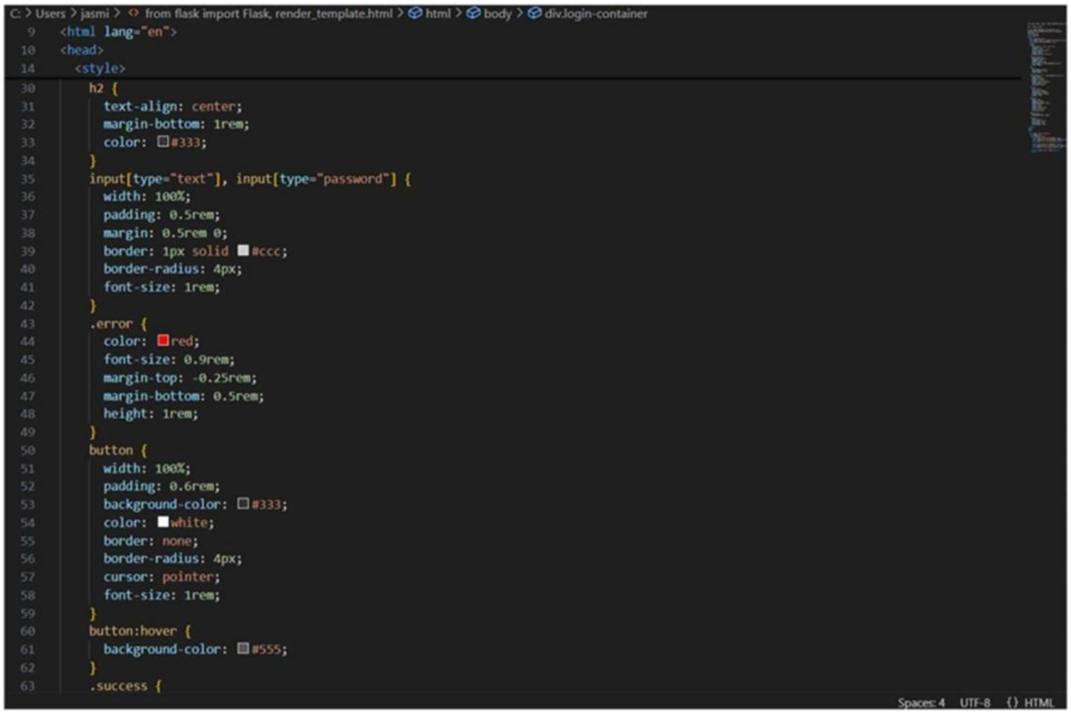
Generate a code to develop Task: Ask AI to generate a Flask app that serves the HTML form (Task #3) and prints the username on successful login.

Code:



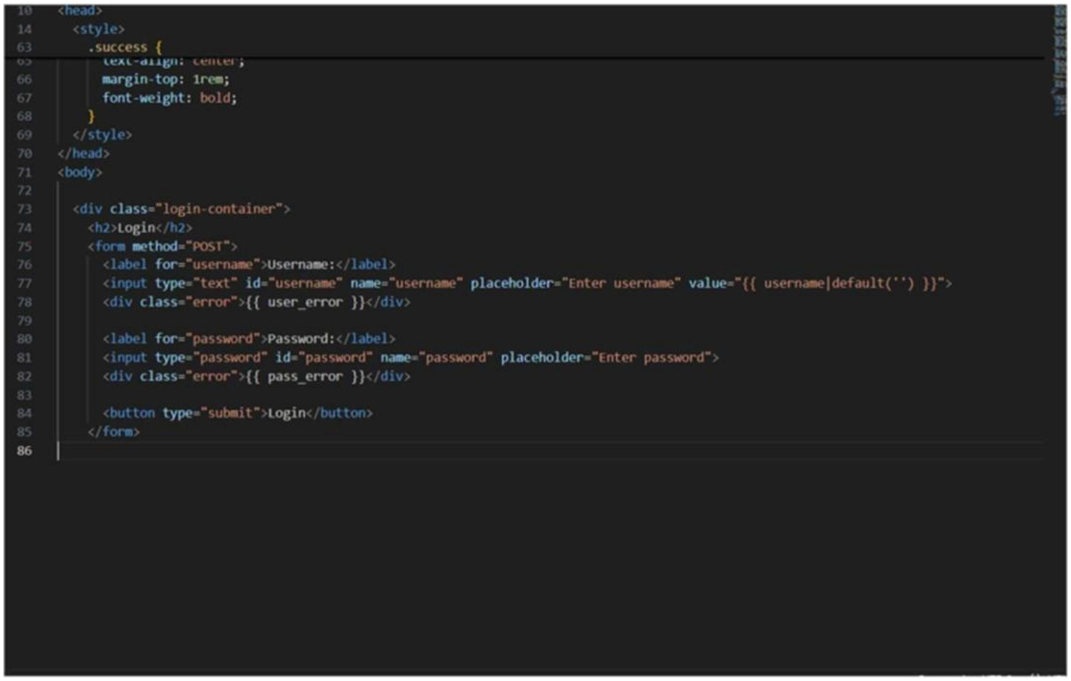
C: > Users > jasmi > from flask import Flask, render_template.html > html > body > div.login-container

```
from flask import Flask, render_template_string, request

app = Flask(__name__)

# ===== HTML Template (from Task #3) =====
# Using render_template_string for simplicity
html_form = """
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Form Validation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f2f2f2;
    }
    .login-container {
      background: white;
      padding: 2rem;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      width: 300px;
    }
    h2 {
      text-align: center;
      margin-bottom: 1rem;
      color: #333;
    }
    input[type="text"], input[type="password"] {
      width: 100%;
      padding: 0.5rem;
```



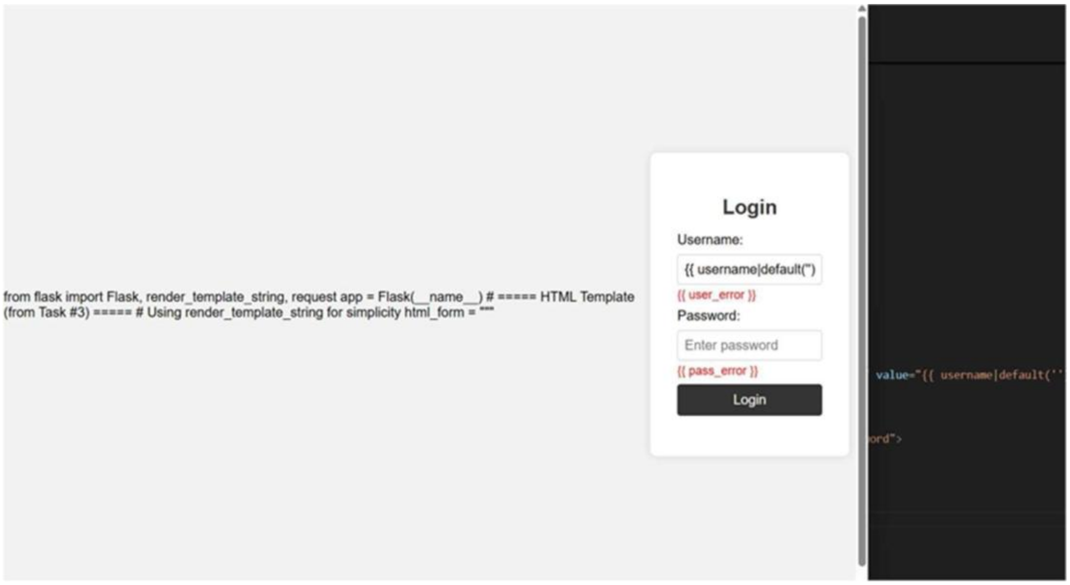
C: > Users > jasmi > from flask import Flask, render_template.html > html > body > div.login-container

```
<html lang="en">
<head>
  <style>
    h2 {
      text-align: center;
      margin-bottom: 1rem;
      color: #333;
    }
    input[type="text"], input[type="password"] {
      width: 100%;
      padding: 0.5rem;
      margin: 0.5rem 0;
      border: 1px solid #ccc;
      border-radius: 4px;
      font-size: 1rem;
    }
    .error {
      color: red;
      font-size: 0.9rem;
      margin-top: -0.25rem;
      margin-bottom: 0.5rem;
      height: 1rem;
    }
    button {
      width: 100%;
      padding: 0.6rem;
      background-color: #333;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 1rem;
    }
    button:hover {
      background-color: #555;
    }
    .success {
```



```
<head>
  <style>
    .success {
      text-align: center;
      margin-top: 1rem;
      font-weight: bold;
    }
  </style>
</head>
<body>

  <div class="login-container">
    <h2>Login</h2>
    <form method="POST">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" placeholder="Enter username" value="{{ username|default('') }}">
      <div class="error">{{ user_error }}</div>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password" placeholder="Enter password">
      <div class="error">{{ pass_error }}</div>

      <button type="submit">Login</button>
    </form>

```

Output:



```
from flask import Flask, render_template_string, request app = Flask(__name__) # ===== HTML Template
(from Task #3) ===== # Using render_template_string for simplicity html_form = """
```

Login

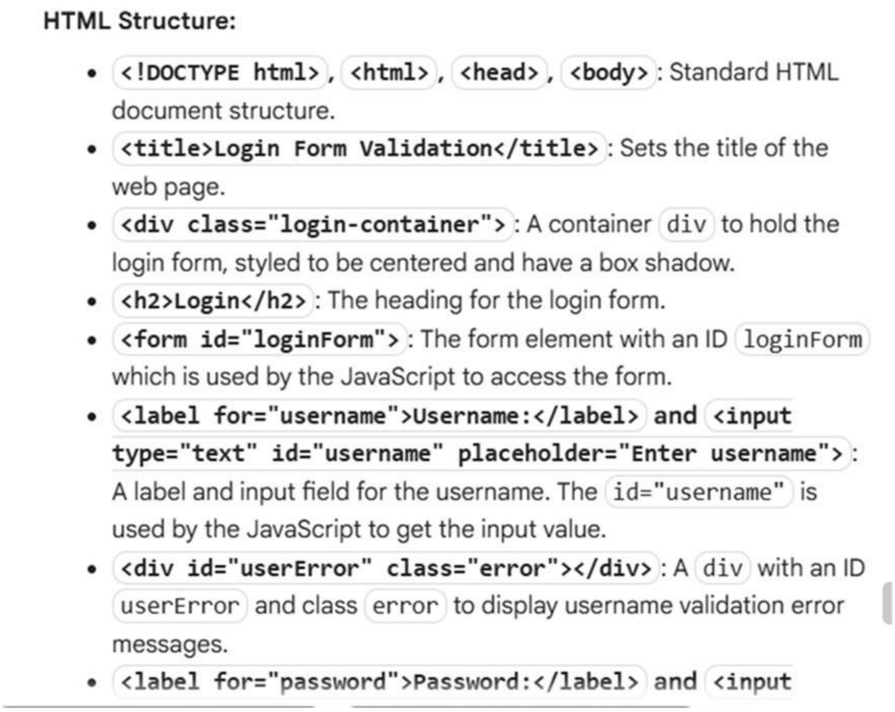
Username: {{ username|default('') }}

{{ user_error }}

Password: Enter password

{{ pass_error }}

Explanation of code:



HTML Structure:

- `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`: Standard HTML document structure.
- `<title>Login Form Validation</title>`: Sets the title of the web page.
- `<div class="login-container">`: A container `div` to hold the login form, styled to be centered and have a box shadow.
- `<h2>Login</h2>`: The heading for the login form.
- `<form id="loginForm">`: The form element with an ID `loginForm` which is used by the JavaScript to access the form.
- `<label for="username">Username:</label>` **and** `<input type="text" id="username" placeholder="Enter username">`: A label and input field for the username. The `id="username"` is used by the JavaScript to get the input value.
- `<div id="userError" class="error"></div>`: A `div` with an ID `userError` and class `error` to display username validation error messages.
- `<label for="password">Password:</label>` **and** `<input`