

LAB ASSIGNMENT 14.3

Program : B. Tech (CSE)
Specializa on : AIML
Course Title : AI Assisted coding
Semester : III
Academic Session : 2025-2026
Name of Student : Saini.Deepthi
Enrollment No : 2403a52014
Batch No. : 02
Date : 29-10-2025

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

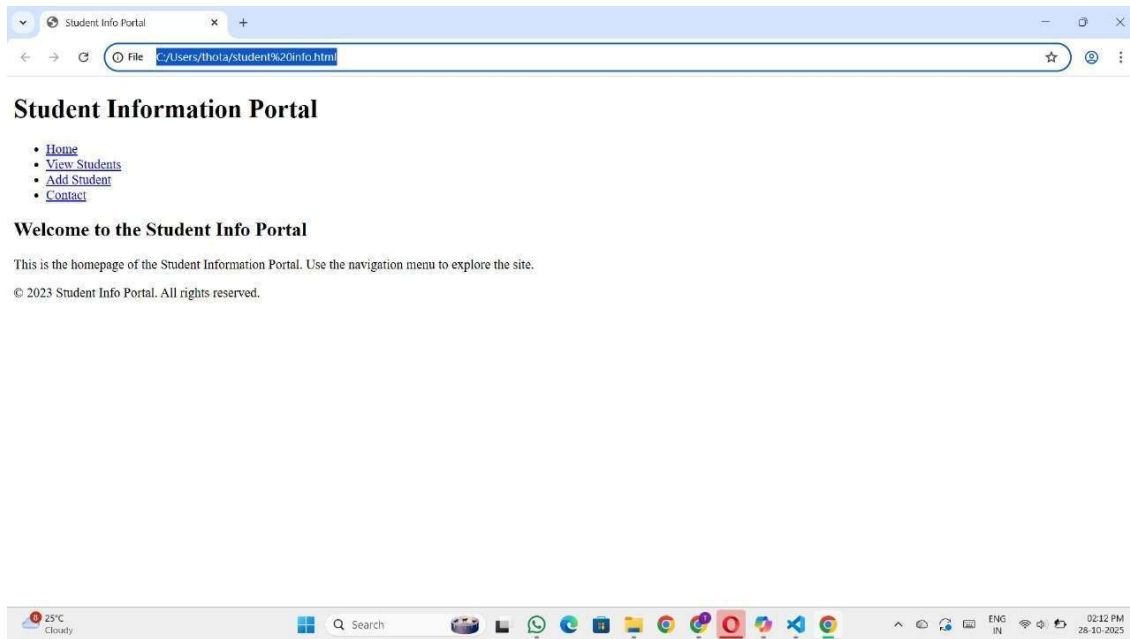
Prompt: Create a basic HTML layout for a "Student Info Portal" homepage. The structure should include a header section, a navigation menu, and a footer. Ensure the code is well-formatted with clean indentation, appropriate HTML tags, and helpful comments for clarity.

```

C: > Users > thota > <> student info.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Student Info Portal</title>
7      <!-- You can link a CSS file here for styling -->
8      <!-- <link rel="stylesheet" href="style.css"> -->
9  </head>
10 <body>
11
12     <!-- Header Section -->
13     <header>
14         <h1>Student Information Portal</h1>
15     </header>
16
17     <!-- Navigation Menu -->
18     <nav>
19         <ul>
20             <li><a href="#">Home</a></li>
21             <li><a href="#">View Students</a></li>
22             <li><a href="#">Add Student</a></li>
23             <li><a href="#">Contact</a></li>
24         </ul>
25     </nav>
26
27     <!-- Main Content Area (can be filled later) -->
28     <main>
29         <h2>Welcome to the Student Info Portal</h2>
30         <p>This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.</p>
31         <!-- Content for the homepage goes here -->
32     </main>
33
34     <!-- Footer Section -->
35     <footer>
36         <p>&copy; 2023 Student Info Portal. All rights reserved.</p>
37     </footer>
38
39 </body>
40 </html>

```

Output:



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.
 - ``: An unordered list, used here for the navigation links.
 - `...`: 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>© 2023 Student Info Portal. All rights reserved.</p>**: A paragraph containing copyright information. The **©** 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 `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:

```
C: > Users > thota > <> css.html > <img alt="html icon" data-bbox="338 208 355 220"/> html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Student Info Portal</title>
7      <style>
8          /* Basic styling for the body and overall layout */
9          body {
10              font-family: sans-serif;
11              margin: 0;
12              padding: 0;
13              line-height: 1.6;
14          }
15
16          /* Header styling */
17          header {
18              background: #f4f4f4;
19              padding: 1rem;
20              text-align: center;
21          }
22
23          /* Navigation bar styling */
24          nav {
25              background: #333;
26              color: #fff;
27              padding: 0.5rem 0;
28          }
29
```

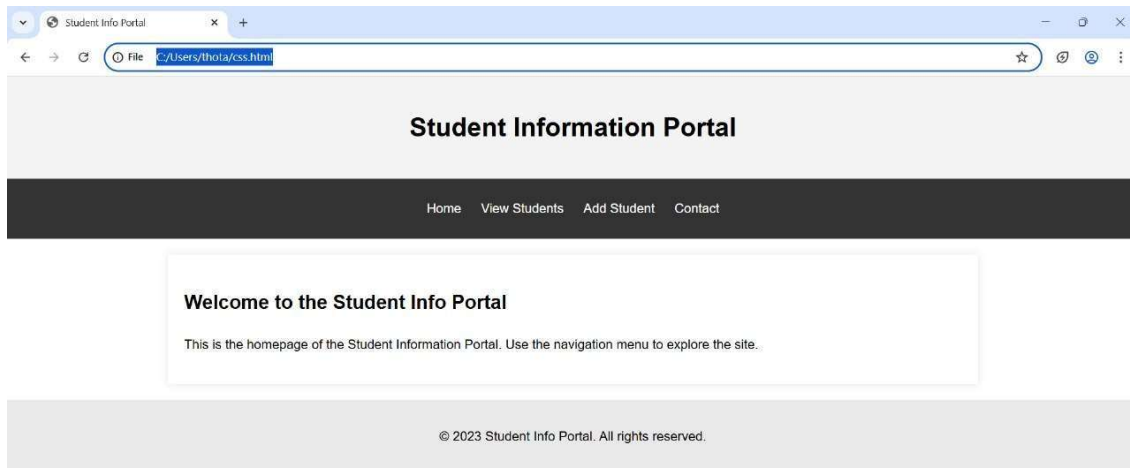
```
29
30     nav ul {
31         padding: 0;
32         list-style: none;
33         text-align: center;
34     }
35
36     nav ul li {
37         display: inline;
38         margin: 0 10px;
39     }
40
41     nav ul li a {
42         color: #ffff;
43         text-decoration: none;
44     }
45
46     /* Responsive navigation for smaller screens */
47     @media (max-width: 768px) {
48         nav ul li {
49             display: block;
50             margin-bottom: 5px;
51         }
52     }
53
```

```

54     /* Main content styling - centered */
55     main {
56         padding: 20px;
57         max-width: 960px; /* Limit content width */
58         margin: 20px auto; /* Center the content block */
59         background: #fff;
60         box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
61     }
62
63     /* Footer styling */
64     footer {
65         background: #e9e9e9; /* Light gray background */
66         text-align: center;
67         padding: 1rem;
68         margin-top: 20px; /* Add space above the footer */
69     }
70 </style>
71 </head>
72 <body>
73
74     <!-- Header Section -->
75     <header>
76         <h1>Student Information Portal</h1>
77     </header>
78
79     <!-- Navigation Menu -->
80     <nav>
81         <ul>
82             <li><a href="#">Home</a></li>
83             <li><a href="#">View Students</a></li>
84             <li><a href="#">Add Student</a></li>
85             <li><a href="#">Contact</a></li>
86         </ul>
87     </nav>
88
89     <!-- Main Content Area -->
90     <main>
91         <h2>Welcome to the Student Info Portal</h2>
92         <p>This is the homepage of the Student Information Portal. Use the navigation menu to explore the site.</p>
93         <!-- Content for the homepage goes here -->
94     </main>
95
96     <!-- Footer Section -->
97     <footer>
98         <p>&copy; 2023 Student Info Portal. All rights reserved.</p>
99     </footer>
100
101 </body>
102 </html>

```

Output:



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`).

In summary, this code provides a robust way to generate an HTML file programmatically, 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 validation. 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 validation. Clear error messages if inputs are empty.

Code:

```

<!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;

```

```

    }
    .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="loginForm" class="error"></div>

```

```

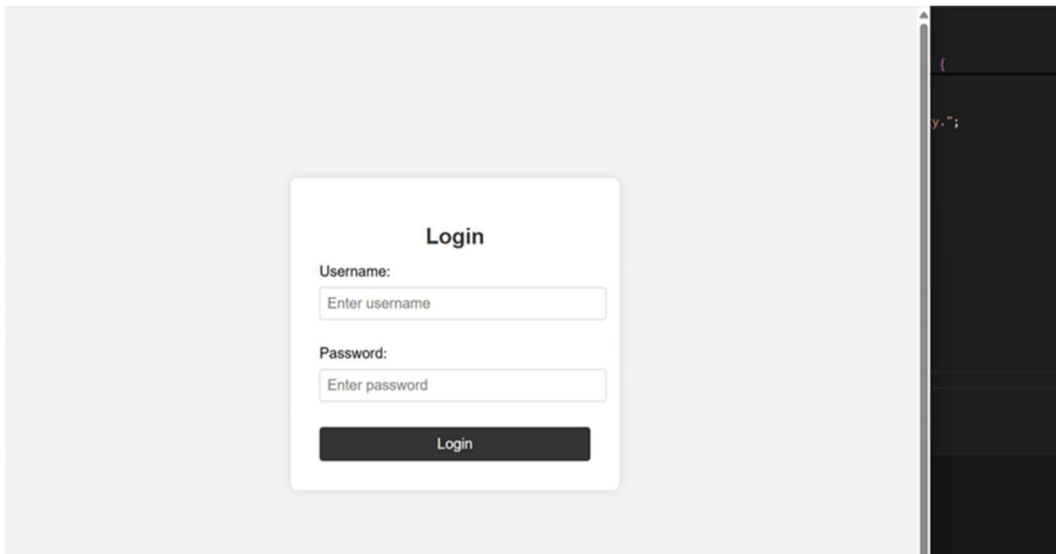
// Validate username
if (username === "") {
    document.getElementById("userError").textContent = "Username cannot be empty.";
    valid = false;
}

// 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 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 ID `passError` and class `error` to display password validation error messages.
- `<button type="submit">Login</button>`: The submit button for 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> jsmi > from flask import Flask, render_template.html > html > body > div.login-container

```
1 from flask import Flask, render_template_string, request
2
3 app = Flask(__name__)
4
5 # ===== HTML Template (from Task #3) =====
6 # Using render_template_string for simplicity
7 html_form = """
8 <!DOCTYPE html>
9 <html lang="en">
10 <head>
11   <meta charset="UTF-8">
12   <meta name="viewport" content="width=device-width, initial-scale=1.0">
13   <title>Login Form Validation</title>
14   <style>
15     body {
16       font-family: Arial, sans-serif;
17       display: flex;
18       justify-content: center;
19       align-items: center;
20       height: 100vh;
21       background-color: #f2f2f2;
22     }
23     .login-container {
24       background: white;
25       padding: 2rem;
26       border-radius: 8px;
27       box-shadow: 0 0 10px rgba(0,0,0,0.1);
28       width: 300px;
29     }
30     h2 {
31       text-align: center;
32       margin-bottom: 1rem;
33       color: #333;
34     }
35     input[type="text"], input[type="password"] {
36       width: 100%;
37       padding: 0.5rem;
```

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

```
9   <html lang="en">
10   <head>
11     <style>
12
13     h2 {
14       text-align: center;
15       margin-bottom: 1rem;
16       color: #333;
17     }
18     input[type="text"], input[type="password"] {
19       width: 100%;
20       padding: 0.5rem;
21       margin: 0.5rem 0;
22       border: 1px solid #ccc;
23       border-radius: 4px;
24       font-size: 1rem;
25     }
26     .error {
27       color: red;
28       font-size: 0.9rem;
29       margin-top: -0.25rem;
30       margin-bottom: 0.5rem;
31       height: 1rem;
32     }
33     button {
34       width: 100%;
35       padding: 0.6rem;
36       background-color: #333;
37       color: white;
38       border: none;
39       border-radius: 4px;
40       cursor: pointer;
41       font-size: 1rem;
42     }
43     button:hover {
44       background-color: #555;
45     }
46     .success {
```

Spaces: 4 UTF-8 () HTML

```

10 <head>
14   <style>
63     .success {
66       text-align: center;
67       margin-top: 1rem;
68       font-weight: bold;
69     }
70   </style>
71 </head>
72 <body>
73   <div class="login-container">
74     <h2>Login</h2>
75     <form method="POST">
76       <label for="username">Username:</label>
77       <input type="text" id="username" name="username" placeholder="Enter username" value="{{ username|default('') }}">
78       <div class="error">{{ user_error }}</div>
79
80       <label for="password">Password:</label>
81       <input type="password" id="password" name="password" placeholder="Enter password">
82       <div class="error">{{ pass_error }}</div>
83
84       <button type="submit">Login</button>
85     </form>
86   </div>

```

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:

{{ user_error }}

Password:

{{ pass_error }}

Login

```

value="{{ username|default('') }}"
password">

```

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`
-