

LAB ASSIGNMENT_14

NAME : T.Akanksha

ROLL NO : 2403A510D8

BATCH 01

COURSE : ASSISTED CODING

QUESTION

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 `<header>`, `<nav>`, `<footer>`.
- Clean indentation, proper tags, and comments.

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`:

Task Description #3 – JavaScript Interactivity

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.

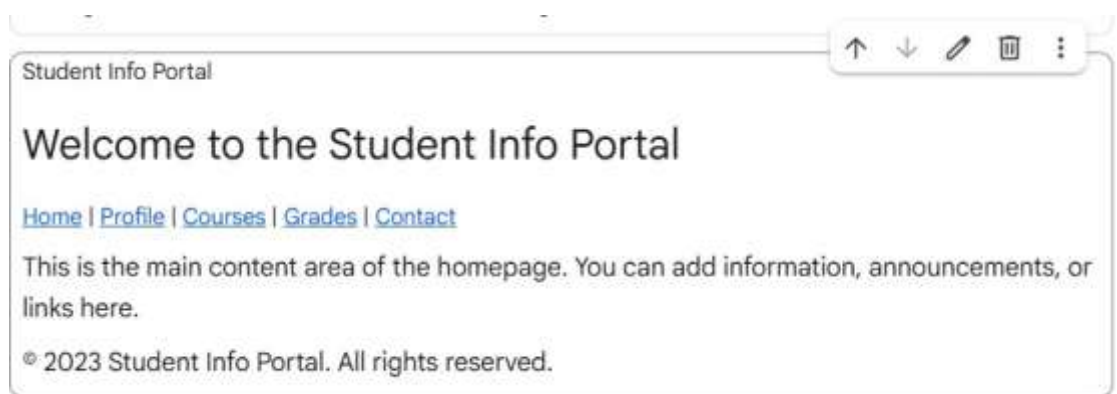
Task Description #4 – Python Backend Integration

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

TASK 1

PROMPT

Create a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer.
give me the code



CODE

▶ `from IPython.display import HTML`

```
html_content = """
<!DOCTYPE html>
<html>
<head>
<title>Student Info Portal</title>
<style>
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }
  header {
    background-color: #f2f2f2;
    padding: 10px;
    text-align: center;
  }
  nav {
    background-color: #e0e0e0;
    padding: 10px;
    text-align: center;
  }
  nav a {
    margin: 0 15px;
    text-decoration: none;
    color: #333;
  }
  footer {
    background-color: #f2f2f2;
```



```
background-color: #f2f2f2;
padding: 10px;
text-align: center;
position: fixed;
bottom: 0;
width: 100%;
}
</style>
</head>
<body>

<header>
  <h1>Student Info Portal</h1>
</header>

<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Courses</a>
  <a href="#">Contact</a>
</nav>

<main>
  <!-- Content goes here -->
</main>

<footer>
  <p>&copy; 2023 Student Info Portal</p>
</footer>

</body>
```

```
    </nav>

    <main>
      <!-- Content goes here -->
    </main>

    <footer>
      <p>&copy; 2023 Student Info Portal</p>
    </footer>

  </body>
</html>
"""

display(HTML(html_content))
```

TASK 2

PROMPT

Add CSS styling to the "Student Info Portal" homepage:
Make the navigation bar responsive. Center the content section. Give the footer a light gray background. Provide the full HTML + CSS together and explain how the CSS classes work.

1
0s



```
from IPython.display import HTML
```

```
html_content = """
<!DOCTYPE html>
<html>
<head>
<title>Student Info Portal</title>
<style>
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

.navbar {
    background-color: #333;
    overflow: hidden;
}

.navbar a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 20px;
    text-decoration: none;
}

```



```
- }

.navbar a:hover {
  background-color: #ddd;
  color: black;
}

.content {
  padding: 20px;
  text-align: center; /* Center the content */
  max-width: 800px; /* Optional: Limit content width */
  margin: 20px auto; /* Center the content block */
  background-color: #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.footer {
  background-color: #f0f0f0; /* Light gray background */
  color: #333;
  text-align: center;
  padding: 10px;
  position: fixed;
  left: 0;
  bottom: 0;
  width: 100%;
}

/* Responsive design for the navigation bar */
```




```
bottom: 0;
width: 100%;
}

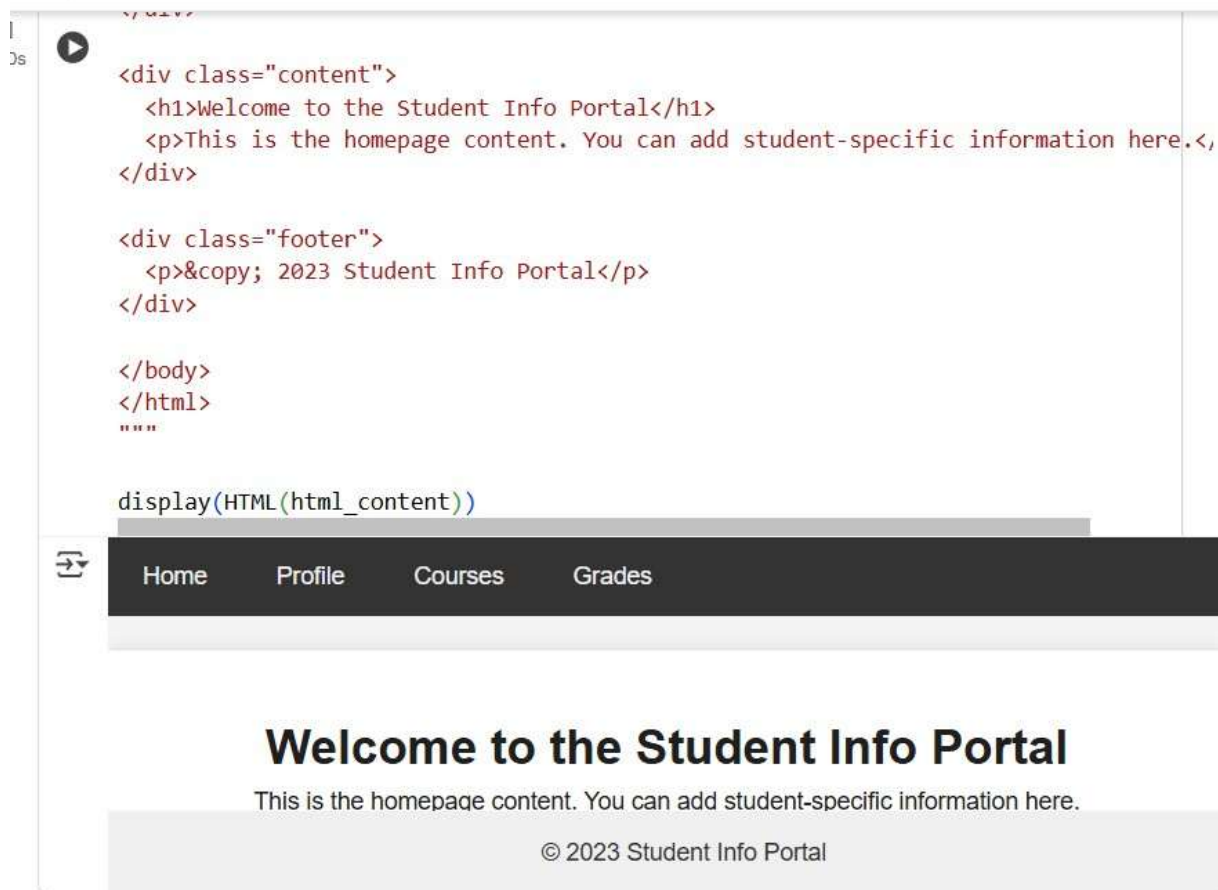
/* Responsive design for the navigation bar */
@media screen and (max-width: 600px) {
    .navbar a {
        float: none;
        display: block;
        text-align: left;
    }
}
</style>
</head>
<body>

<div class="navbar">
    <a href="#">Home</a>
    <a href="#">Profile</a>
    <a href="#">Courses</a>
    <a href="#">Grades</a>
</div>

<div class="content">
    <h1>Welcome to the Student Info Portal</h1>
    <p>This is the homepage content. You can add student-specific information</p>
</div>

<div class="footer">
    <n>&copy; 2023 Student Info Portal</n>
```





TASK 3

PROMPT

Create a JavaScript script to validate a simple login form. Check that username and password are not empty. Show clear error messages if any input is empty. Validation should work when the form is submitted.



```
from IPython.display import HTML
```

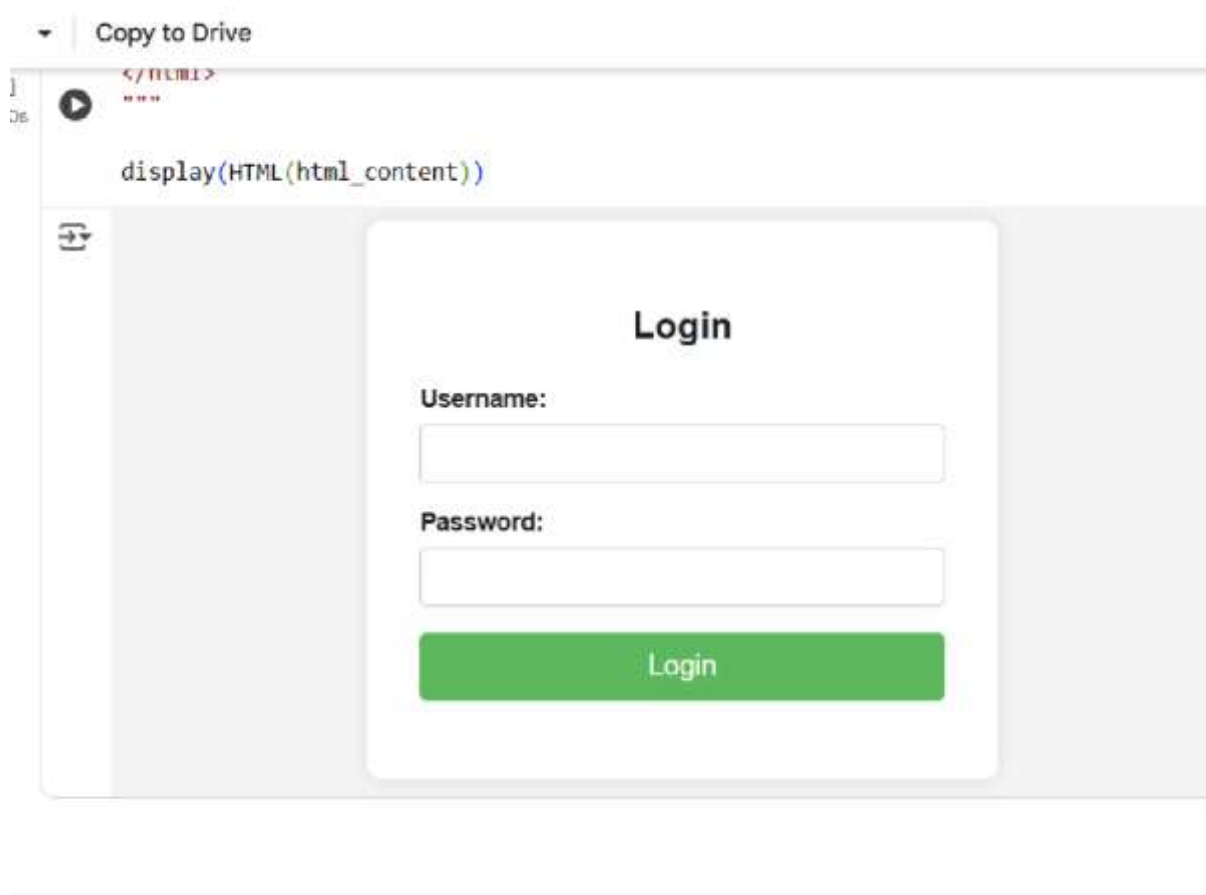
```
html_content = """
<!DOCTYPE html>
<html>
<head>
<title>Login Form</title>
<style>
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 80vh;
    background-color: #f4f4f4;
}
.login-container {
    background-color: #fff;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 300px;
}

```



```
7] }  
0s 0s ▶ .login-container h2 {  
      text-align: center;  
      margin-bottom: 20px;  
    }  
    .form-group {  
      margin-bottom: 15px;  
    }  
    .form-group label {  
      display: block;  
      margin-bottom: 5px;  
      font-weight: bold;  
    }  
    .form-group input {  
      width: 100%;  
      padding: 8px;  
      box-sizing: border-box;  
      border: 1px solid #ccc;  
      border-radius: 4px;  
    }  
    .form-group button {  
      width: 100%;  
      padding: 10px;  
      background-color: #5cb85c;  
      color: white;  
      border: none;  
      border-radius: 4px;  
      cursor: pointer;  
      font-size: 16px;
```

```
    }  
    .form-group button:hover {  
        background-color: #4cae4c;  
    }  
    </style>  
    </head>  
    <body>  
  
    <div class="login-container">  
        <h2>Login</h2>  
        <form id="loginForm">  
            <div class="form-group">  
                <label for="username">Username:</label>  
                <input type="text" id="username" name="username">  
            </div>  
            <div class="form-group">  
                <label for="password">Password:</label>  
                <input type="password" id="password" name="password">  
            </div>  
            <div class="form-group">  
                <button type="submit">Login</button>  
            </div>  
        </form>  
    </div>  
  
    </body>  
    </html>  
    """"
```



TASK 4

Prompt

Create a Flask app that serves the login form (from Task #3) and prints the username on successful login. GIVE ME IN ONE CELL and give fast

```
> from flask import Flask, request, render_template_string
```

```
app = Flask(__name__)
```

```
# HTML for the login form
```

```
login_form_html = """
```

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
    <title>Login</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Login</h1>
```

```
    <form method="post">
```

```
        <p><input type="text" name="username"></p>
```

```
        <p><input type="submit" value="Login"></p>
```

```
    </form>
```

```
</body>
```

```
</html>
```

```
"""
```

```
@app.route('/')
```

```
def index():
```

```
    return render_template_string(login_form_html)
```

```
@app.route('/', methods=['POST'])
```

```
def login():
```

Copy to Drive

```
<form method="post">
    <p><input type="text" name="username"></p>
    <p><input type="submit" value="Login"></p>
</form>
</body>
</html>
"""

@app.route('/')
def index():
    return render_template_string(login_form_html)

@app.route('/', methods=['POST'])
def login():
    username = request.form.get('username')
    if username:
        return f'Hello, {username}!'
    else:
        return 'Please enter a username.'

# To run the Flask app in Colab, you'll need to use a tool like ngrok
# You can install it and run it with:
# !pip install flask-ngrok
# from flask_ngrok import run_with_ngrok
# run_with_ngrok(app)

if __name__ == '__main__':
    # This will run the Flask development server locally.
    # In Colab, you'll typically want to use run_with_ngrok instead.
```


Copy to Drive

```
<form method="post">
    <p><input type="text" name="username"></p>
    <p><input type="submit" value="Login"></p>
</form>
</body>
</html>
"""

@app.route('/')
def index():
    return render_template_string(login_form_html)

@app.route('/', methods=['POST'])
def login():
    username = request.form.get('username')
    if username:
        return f'Hello, {username}!'
    else:
        return 'Please enter a username.'

# To run the Flask app in Colab, you'll need to use a tool like ngrok
# You can install it and run it with:
# !pip install flask-ngrok
# from flask_ngrok import run_with_ngrok
# run_with_ngrok(app)

if __name__ == '__main__':
    # This will run the Flask development server locally.
    # In Colab, you'll typically want to use run_with_ngrok instead.
```

```
!pip install flask-ngrok
# from flask_ngrok import run_with_ngrok
# run_with_ngrok(app)

if __name__ == '__main__':
    # This will run the Flask development server locally.
    # In Colab, you'll typically want to use run_with_ngrok instead.
    # app.run(debug=True)
    print("Flask app is running. To access it, use ngrok or a similar service in Colab.")
```

🔗 Flask app is running. To access it, use ngrok or a similar service in Colab.