

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.

Prompt: give a python code for a simple html homepage for a student info portal with header, navigation menu, and footer

```
html_content = """
<!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>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      display: flex;
      flex-direction: column;
      min-height: 100vh;
    }
    header {
      background-color: #4CAF50;
      color: white;
      padding: 1em 0;
      text-align: center;
    }
    nav {
      background-color: #333;
      overflow: hidden;
    }
    nav a {
      float: left;
      display: block;
      color: white;
      text-align: center;
      padding: 14px 16px;
      text-decoration: none;
    }
    nav a:hover {
      background-color: #ddd;
      color: black;
    }
    .container {
      flex: 1;
      padding: 20px;
    }
    footer {
      background-color: #f1f1f1;
      padding: 1em 0;
      text-align: center;
      margin-top: auto;
    }
  </style>
</head>
<body>

  <header>
    <h1>Welcome to the Student Info Portal</h1>
  </header>

  <nav>
    <a href="#home">Home</a>
    <a href="#profile">Profile</a>
    <a href="#courses">Courses</a>
    <a href="#contact">Contact</a>
  </nav>

  <div class="container">
    <h2>Homepage Content</h2>
    <p>This is the main content area of the student info portal.</p>
    <p>Add information and links here relevant to students.</p>
  </div>

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

```

```
</body>
</html>
"""

# You can now use the 'html_content' variable.
# For example, you could save it to an HTML file:
# with open("index.html", "w") as f:
#     f.write(html_content)

# Or display it directly in the notebook (requires IPython.display.HTML)
from IPython.display import display, HTML
display(HTML(html_content))
```

Welcome to the Student Info Portal

[Home](#) [Profile](#) [Courses](#) [Contact](#)

Homepage Content

This is the main content area of the student info portal.
Add information and links here relevant to students.

© 2023 Student Info Portal

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.

Prompt: add css styling for the above student info portals homepage for responsive navigation bar, centered content section, footer with light grey in another cell

```
html_content_styled = """
<!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>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      display: flex;
      flex-direction: column;
      min-height: 100vh;
    }
    header {
      background-color: #4CAF50;
      color: white;
      padding: 1em 0;
      text-align: center;
    }
    nav {
      background-color: #333;
      overflow: hidden;
    }
    nav a {
      float: left;
      display: block;
      color: white;
      text-align: center;
      padding: 14px 16px;
      text-decoration: none;
    }
    nav a:hover {
      background-color: #ddd;
      color: black;
    }
    .container {
```

```

        flex: 1;
        padding: 20px;
        max-width: 800px; /* Limit content width */
        margin: 20px auto; /* Center the container */
        background-color: #fff; /* White background for content */
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
    }
    footer {
        background-color: #f1f1f1; /* Light gray background */
        padding: 1em 0;
        text-align: center;
        margin-top: auto;
    }

    /* Responsive Navigation Bar */
    @media screen and (max-width: 600px) {
        nav a {
            float: none;
            display: block;
            text-align: left;
        }
    }
</style>
</head>
<body>

    <header>
        <h1>Welcome to the Student Info Portal</h1>
    </header>

    <nav>
        <a href="#home">Home</a>
        <a href="#profile">Profile</a>
        <a href="#courses">Courses</a>
        <a href="#contact">Contact</a>
    </nav>

    <div class="container">
        <h2>Homepage Content</h2>
        <p>This is the main content area of the student info portal.</p>
        <p>Add information and links here relevant to students.</p>
    </div>

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

</body>
</html>
"""

# You can now use the 'html_content_styled' variable.
# For example, you could save it to an HTML file:
# with open("index_styled.html", "w") as f:
#     f.write(html_content_styled)

# Or display it directly in the notebook (requires IPython.display.HTML)
from IPython.display import display, HTML
display(HTML(html_content_styled))

```

Welcome to the Student Info Portal

[Home](#) [Profile](#) [Courses](#) [Contact](#)

Homepage Content

This is the main content area of the student info portal.
Add information and links here relevant to students.

© 2023 Student Info Portal

Task Description #3 – JavaScript Interactivity

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

Prompt:

```
js_script = ""
<script>
function validateForm() {
  var username = document.forms["loginForm"]["username"].value;
  var password = document.forms["loginForm"]["password"].value;
  if (username == "" || password == "") {
    alert("Username and password must be filled out");
    return false;
  }
  return true; // Allow form submission if validation passes
}
</script>

<form name="loginForm" onsubmit="return validateForm()" method="post">
  Username: <input type="text" name="username"><br>
  Password: <input type="password" name="password"><br>
  <input type="submit" value="Submit">
</form>
""
```

```
from IPython.display import display, HTML
display(HTML(js_script))
```

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.

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

# HTML content including the form and the JS script
html_content_with_form = """
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Portal</title>
  <style>
    body { font-family: Arial, sans-serif; }
    .container { margin-top: 50px; text-align: center; }
    input { margin: 5px; }
  </style>
  <script>
    function validateForm() {
      var username = document.forms["loginForm"]["username"].value;
      var password = document.forms["loginForm"]["password"].value;
      if (username == "" || password == "") {
        alert("Username and password must be filled out");
        return false;
      }
      return true; // Allow form submission if validation passes
    }
  </script>
</head>
<body>
  <div class="container">
    <h2>Login</h2>
    <form name="loginForm" onsubmit="return validateForm()" method="post">
      Username: <input type="text" name="username"><br>
      Password: <input type="password" name="password"><br>
      <input type="submit" value="Submit">
    </form>
  </div>
</body>
</html>
"""

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

@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    password = request.form.get('password')
    # In a real application, you would validate the password here
    print(f"Login successful for user: {username}")
    return f"Welcome, {username}!"

if __name__ == '__main__':
    # To run this in Colab, you might need to use a tool like ngrok
    # See instructions on running Flask apps in Colab:
    # https://colab.research.google.com/github/googlecolab/colabtools/blob/main/notebooks/misc/run\_flask\_in\_colab.ipynb
    # app.run(debug=True)

    # For a simple demonstration within Colab's output frame:
    # This won't run the Flask server in the background, but shows the HTML.
    from IPython.display import display, HTML
    display(HTML(html_content_with_form))
```

Login