AI ASSISTED CODING

ASSIGNMENT 14.3

NAME: ALUWALA SHARANYA

HT NO: 2403A52013

BATCH NO: 02

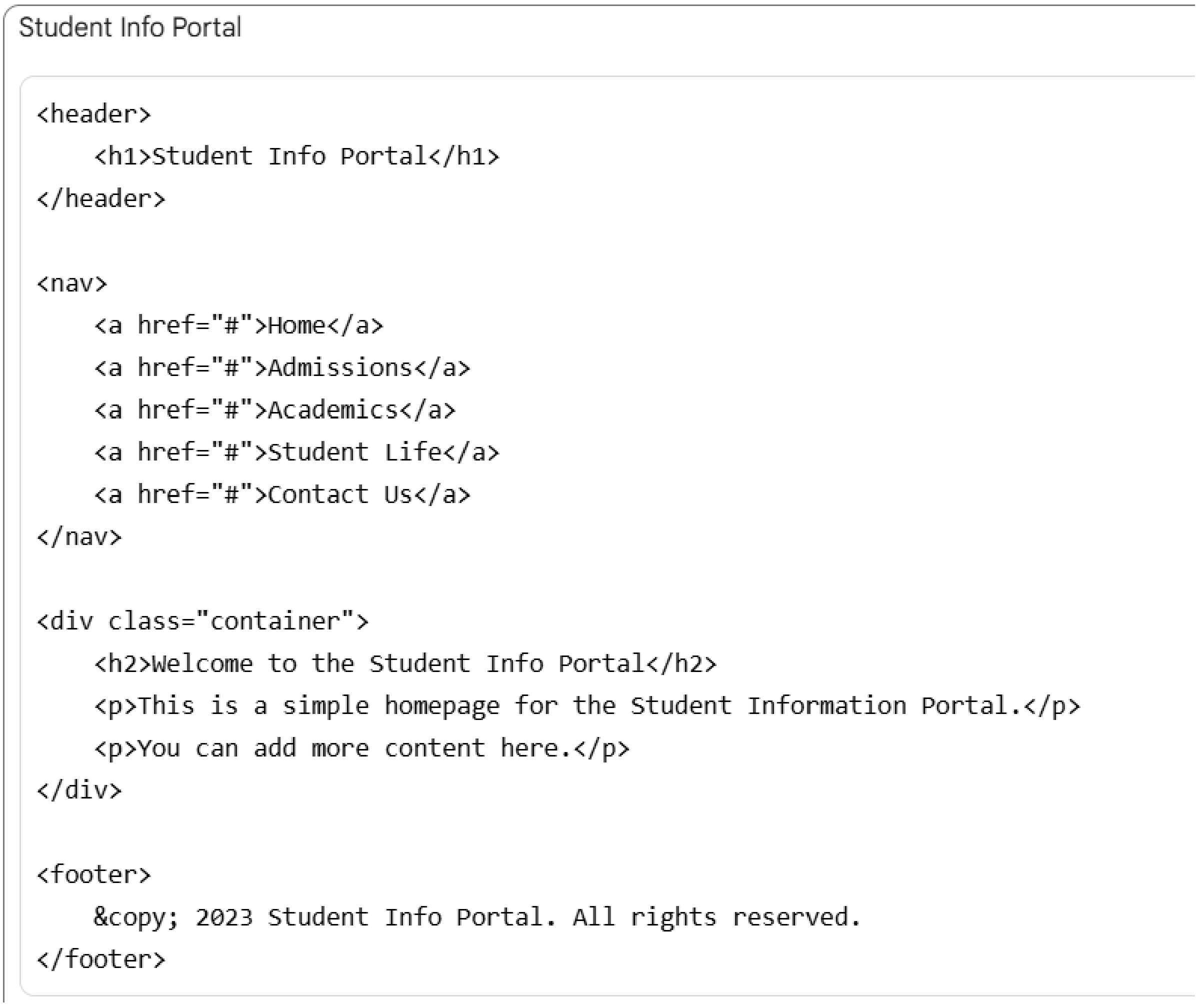
TASK 1~

Ask AI to generate a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer.

PROMPT:

Generate a simple HTML homepage for a 'Student Info Portal' with a header, navigation menu, and footer. The output should include HTML code with <header>, <nav>, and <footer> tags, clean indentation, proper tags, and comments."

CODE & OUTPUT:



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

<nav>
    <a href="#">Home</a>
    <a href="#">Admissions</a>
    <a href="#">Academics</a>
    <a href="#">Student Life</a>
    <a href="#">Contact Us</a>
</nav>

<div class="container">
    <h2>Welcome to the Student Info Portal</h2>
    <p>This is a simple homepage for the Student Information Portal.</p>
    <p>You can add more content here.</p>
</div>

<footer>
    &copy; 2023 Student Info Portal. All rights reserved.
</footer>
```

CODE EXPLANATION:

This is a simple HTML page for a "Student Info Portal." It includes a header with the portal title, a navigation bar with links, a main content area for information, and a footer with copyright details. Basic CSS is included to style the elements for better presentation.

TASK 2~

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 to the provided HTML homepage for a 'Student Info Portal' to include:

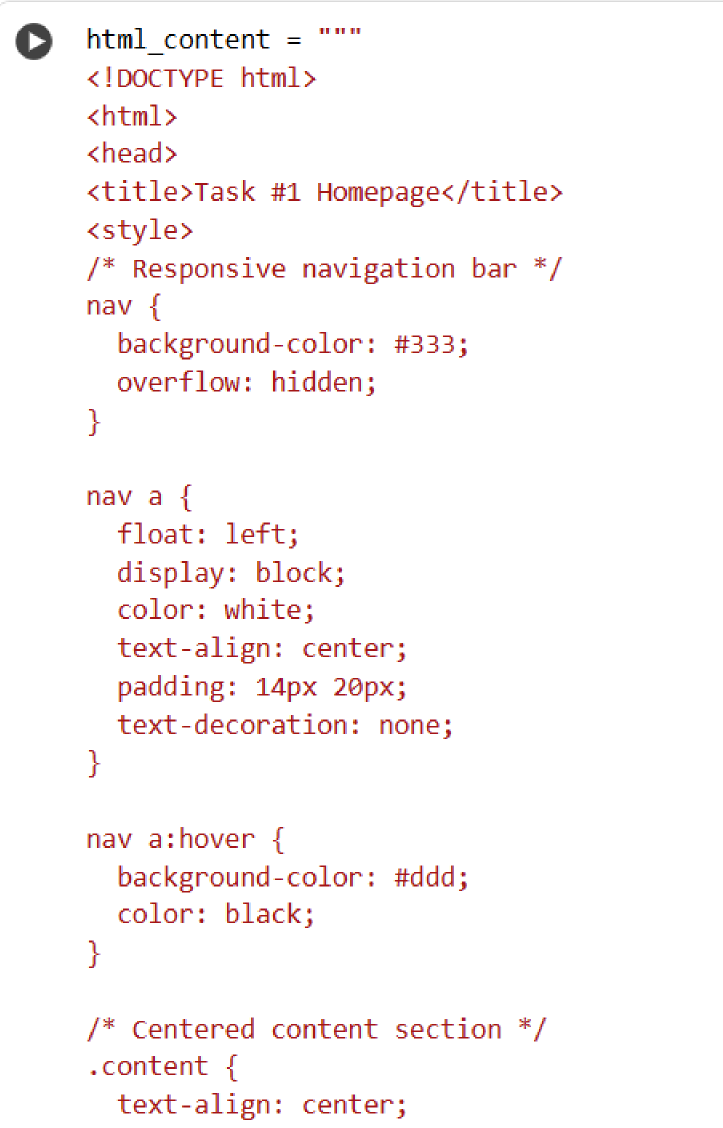
A responsive navigation bar.

A centered content section.

A footer with a light gray background.

The output should be the combined HTML and CSS code, and an explanation of how the CSS styles are applied."

CODE & OUTPUT:

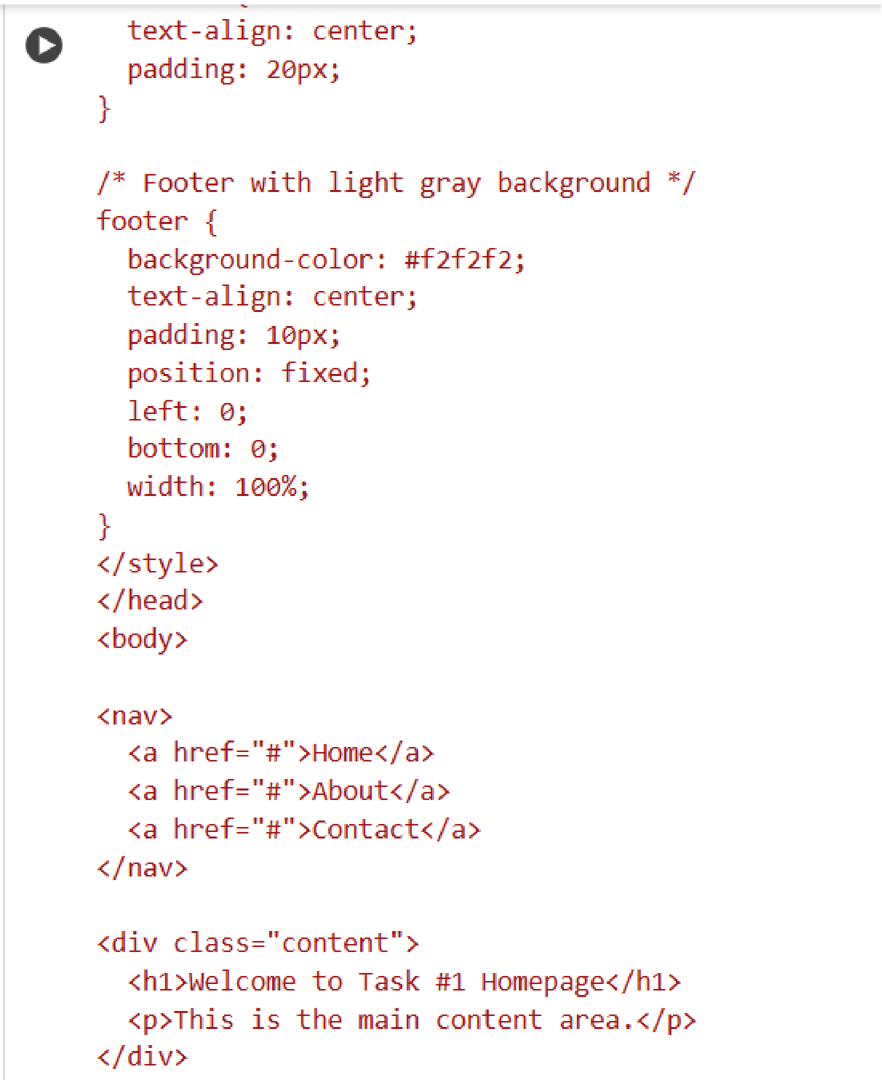


```
html_content = """
<!DOCTYPE html>
<html>
<head>
<title>Task #1 Homepage</title>
<style>
/* Responsive navigation bar */
nav {
  background-color: #333;
  overflow: hidden;
}

nav a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
}

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

/* Centered content section */
.content {
  text-align: center;
```

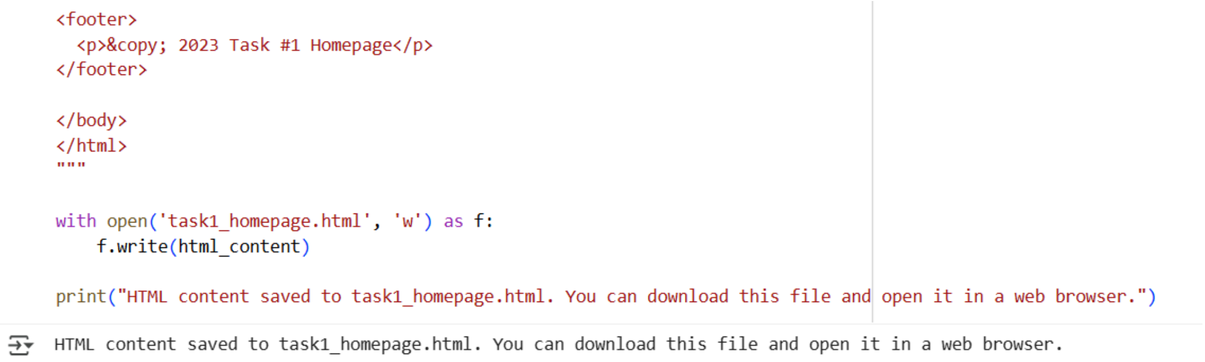


```
    text-align: center;
    padding: 20px;
  }

  /* Footer with light gray background */
  footer {
    background-color: #f2f2f2;
    text-align: center;
    padding: 10px;
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
  }
  </style>
  </head>
  <body>

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

  <div class="content">
    <h1>Welcome to Task #1 Homepage</h1>
    <p>This is the main content area.</p>
  </div>
```



```
<footer>
  <p>&copy; 2023 Task #1 Homepage</p>
</footer>

</body>
</html>
"""

with open('task1_homepage.html', 'w') as f:
    f.write(html_content)

print("HTML content saved to task1_homepage.html. You can download this file and open it in a web browser.")
```

```
HTML content saved to task1_homepage.html. You can download this file and open it in a web browser.
```

CODE EXPLANATION:

This code saves HTML content to a file named task1_homepage.html. It defines the HTML structure of a simple webpage with a navigation bar, a main content area, and a footer. The <style> tags contain CSS rules to style these elements, including making the navigation bar responsive, centering the content, and giving the footer a light gray background. Finally, it writes this HTML to the specified file.

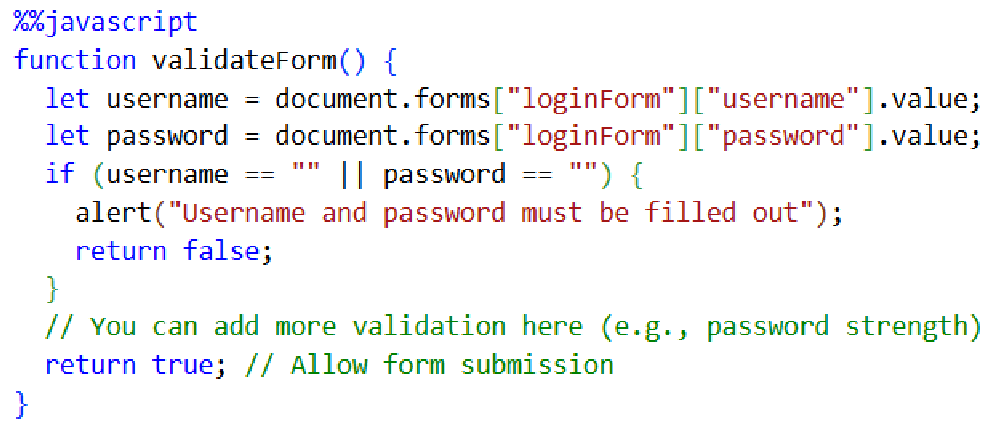
TASK 3~

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

PROMPT:

Generate a Javascript script that validates a simple login form. The script should check if the username and password fields are non-empty and provide clear error messages if they are empty. The expected output is a working Javascript validation script that can be used on form submission

CODE & OUTPUT:



```
%%javascript
function validateForm() {
  let username = document.forms["loginForm"]["username"].value;
  let password = document.forms["loginForm"]["password"].value;
  if (username == "" || password == "") {
    alert("Username and password must be filled out");
    return false;
  }
  // You can add more validation here (e.g., password strength)
  return true; // Allow form submission
}
```

CODE EXPLANATION:

This Javascript code, run with %%javascript, defines a function validateForm. It checks if the 'username' and 'password' fields in a form named 'loginForm' are empty. If either is empty, it shows an alert and returns false to prevent form submission. Otherwise, it returns true, allowing the form to submit. This ensures required fields are filled before processing.

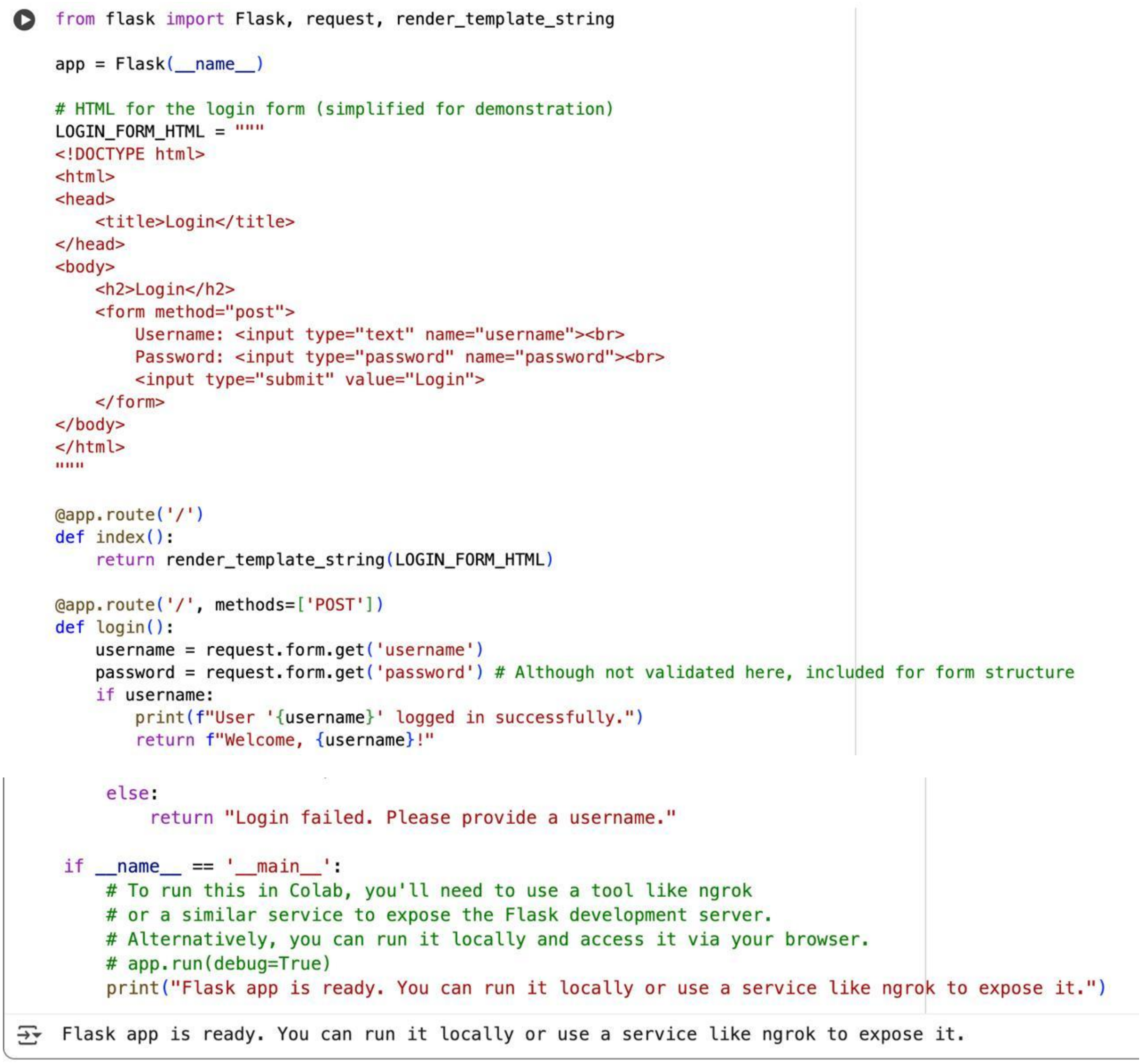
TASK 4~

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

PROMPT:

Generate a Flask application that serves an HTML form (similar to the one generated in Task #3) and prints the entered username to the console upon successful login. The application should handle both GET and POST requests for the login route.

CODE & OUTPUT:



```
from flask import Flask, request, render_template_string

app = Flask(__name__)

# HTML for the login form (simplified for demonstration)
LOGIN_FORM_HTML = """
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </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')
    password = request.form.get('password') # Although not validated here, included for form structure
    if username:
        print(f"User '{username}' logged in successfully.")
        return f"Welcome, {username}!"
    else:
        return "Login failed. Please provide a username."

if __name__ == '__main__':
    # To run this in Colab, you'll need to use a tool like ngrok
    # or a similar service to expose the Flask development server.
    # Alternatively, you can run it locally and access it via your browser.
    # app.run(debug=True)
    print("Flask app is ready. You can run it locally or use a service like ngrok to expose it.")
```

```
Flask app is ready. You can run it locally or use a service like ngrok to expose it.
```

CODE EXPLANATION:

This Python code uses the Flask library to create a simple web app. It displays a login form. When you submit the form, the app takes the username you entered and prints it in the console, then shows a welcome message in your browser. It's set up to run as a web server, but you'd need extra steps (like using ngrok) to access it from outside of where the code is running.