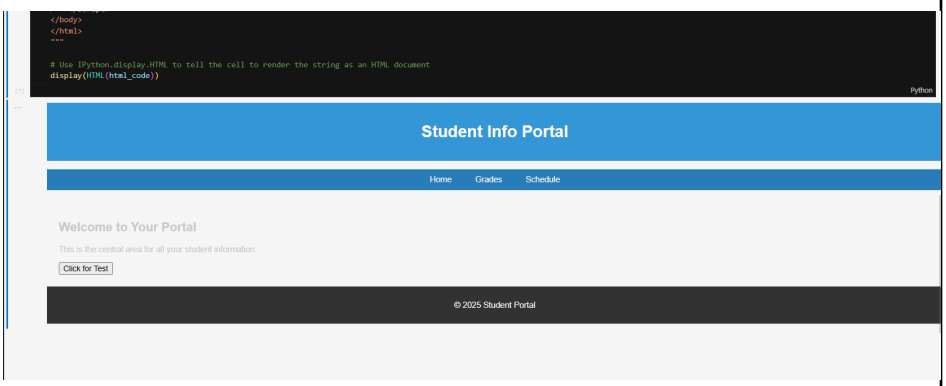
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M. Tech/MCA/MSC		**Assignment Type: Lab**	**AcademicYear:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		**Course Title**	AI Assisted Problem Solving Using Python
Year/Sem	II/I	**Regulation**	R25
Date and Day of Assignment	Week5 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	**M**. Tech/MCA/MSC
AssignmentNumber:14.3(Presentassignmentnumber)/**24**(Totalnumberofassignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 14 – Web Frontend Development: AI-assisted HTML/CSS/JS with Python **Lab Objectives** • To understand how AI can generate HTML/CSS/JS templates. • To practice integrating frontend and backend (Python) for small apps. • To evaluate AI-generated code for readability, reusability, and responsiveness. **Learning Outcomes** After completing this lab, students will be able to: 1. Generate HTML/CSS layouts using AI tools. 2. Add JavaScript interactivity with AI suggestions. 3. Integrate basic Python (Flask/Streamlit) backend to serve frontend. 4. Evaluate AI-generated web code for responsiveness and usability. **5.** Debug and refine AI-generated frontend code. **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.	Week5 - Tuesday

```
from IPython.display import HTML, display

# Define the HTML content as a multiline Python string
html_code = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Portal - Fixed</title>

    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px; /* Line 11 is where the error appears when incorrectly parsed */
            background-color: #f5f5f5;
        }

        header {
            background-color: #3498db;
            color: white;
            padding: 15px;
            text-align: center;
        }

        nav ul {
            list-style: none;
            padding: 0;
            display: flex;
            justify-content: center;
            background-color: #2980b9;
        }

        nav ul li a {
            display: block;
            padding: 10px 20px;
            color: white;
            text-decoration: none;
        }
```



Task Description #2 – CSS Styling

Task:

Use AI to add **CSS styling** to Task #1 homepage for:

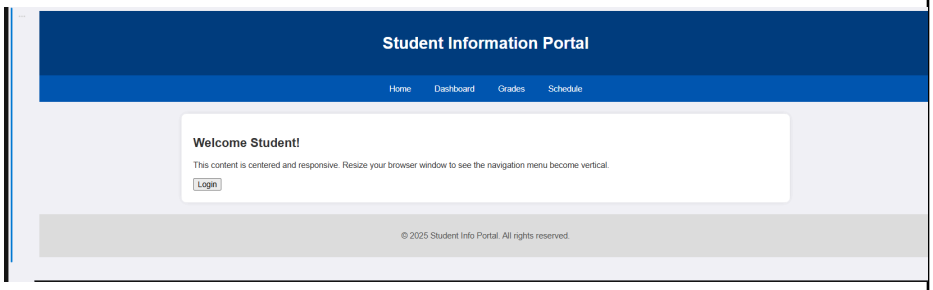
- Responsive navigation bar.
- Centered content section.
- Footer with light gray background.

Expected Output:

- HTML + CSS combined.
- AI explains how CSS classes apply.

Expected Output: AI refactors with with open() and try-except:

```
from IPython.display import HTML, display

# 1. Define the HTML content as a multiline Python string (using triple quotes)
# This content contains the responsive CSS and structure you requested.
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 - Stylized</title>
    <style>
        /* BASE STYLES */
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f9;
            color: #333;
            display: flex;
            flex-direction: column;
            min-height: 100vh; /* <-- LINE 17 in the Python string (Safe as string) */
        }

        /* HEADER STYLES */
        header { background-color: #004080; color: white; padding: 20px; text-align: center; }

        /* NAVIGATION BAR - RESPONSIVE STYLES */
        .nav-menu { list-style: none; padding: 0; margin: 0; background-color: #0056b3; display: flex; justify-content: center; }
        .nav-menu li a { display: block; color: white; text-decoration: none; padding: 15px 20px; transition: background-color 0.3s; }
        .nav-menu li a:hover { background-color: #007bff; }
        @media (max-width: 768px) {
            .nav-menu { flex-direction: column; }
            .nav-menu li { border-bottom: 1px solid #004080; }
            .nav-menu li:last-child { border-bottom: none; }
        }

        /* MAIN CONTENT SECTION - CENTERED */
        .content-container {
            flex: 1; padding: 20px; max-width: 1000px;
            margin: 20px auto; /* CENTERS the block horizontally */
```



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:

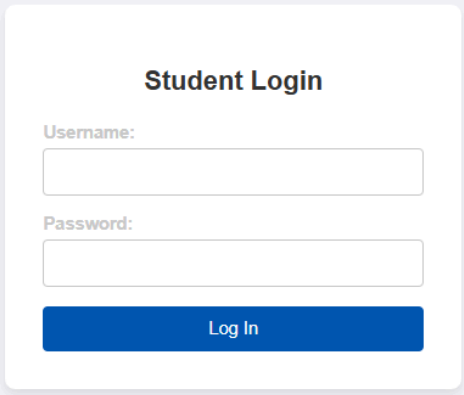
Working on submit JS validation.

Clear error messages if inputs are empty.

```
from IPython.display import HTML, display

# Store the complete HTML/CSS/JS code as a Python string
HTML_CONTENT = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Validation</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            display: flex;
            justify-content: center;
            align-items: center;
            /* Using a fixed height here to ensure rendering in a small Jupyter output area */
            margin: 0;
        }
        .login-container {
            background-color: white;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            width: 300px;
        }
        h2 { text-align: center; color: #333; margin-bottom: 20px; }
        .form-group { margin-bottom: 15px; }
        label { display: block; margin-bottom: 5px; font-weight: bold; }
        input[type="text"], input[type="password"] {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #0056b3;
```

```
                alert('Validation successful! (Form submission prevented for demo.)');
            }

            return false; // Always return false to prevent Jupyter cell reload
        }
    </script>
</body>
</html>
"""

# The essential step for running HTML/CSS/JS code in Jupyter:
display(HTML(HTML_CONTENT))
```

[1]



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.

```python
import threading
from flask import Flask, request, render_template_string

# Create Flask app
app = Flask(__name__)

# HTML form
login_form = """
<!DOCTYPE html>
<html>
<head><title>Login Form</title></head>
<body>
    <h2>Login</h2>
    <form method="POST" action="/login">
        <label>Username:</label>
        <input type="text" name="username" required><br><br>
        <label>Password:</label>
        <input type="password" name="password" required><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
"""

@app.route('/')
def home():
    return render_template_string(login_form)

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    return f"<h3>Login successful! Welcome, {username}.</h3>"

# Function to run Flask in background
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```