|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.   **Prompt:** Generate a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer.  **Code:**          **Code Explanation:**   1. The HTML builds a student info form and an empty card area where the student details will be displayed. 2. The CSS styles the page, form, inputs, button, and the student card for a clean UI. 3. JavaScript dynamically creates form fields using an array of field objects. 4. When the form is submitted, it collects all input values and checks if any field is empty. 5. If valid, it generates and displays a formatted student information card using the entered data.   **Output:**    **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:  **Prompt:** Add responsive navbar, centered content, and a light-gray footer to my Task #1 homepage using combined HTML+CSS, explain how the CSS classes work, and refactor the Python code using open() with proper try-except handling.  **Code:**          **Code Explanation:**   1. The responsive navigation bar uses CSS flexbox so the menu adjusts automatically on different screen sizes. 2. The main content section is centered using margin auto and a max-width to keep it readable on all devices. 3. The footer is styled with a light-gray background and padding to separate it visually from the page content. 4. All HTML and CSS are combined so styles directly apply to the structure without needing external files. 5. The Python refactor uses open() inside a try-except block to safely handle file reading or writing errors such as missing files or permission issues.   **Output:**    **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:** Generate a JavaScript script that validates a login form by checking non-empty username and password fields, shows clear error messages on submit, and ensures the form works correctly.  **Code:**      **Code Explanation:**   1. The Flask app serves an HTML login form that accepts a username and password through POST. 2. The embedded JavaScript prevents default form submission and checks if both fields are non-empty. 3. If either input is empty, it shows an inline error message inside the <div id="e">. 4. If both fields are filled, JavaScript allows the form to submit normally to the Flask route. 5. On the server side, Flask re-checks the inputs; if valid, it prints the logged-in user and displays “Welcome, username!” in the page.   **Output:**    **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 Python Flask backend that serves an HTML login form, validates the input, and prints the username on successful login.  **Code:**    **Code Explanation:**   1. A Flask app is created and defines a route / that handles both GET and POST requests. 2. When the page is loaded with GET, it returns an HTML login form asking for username and password. 3. When the form is submitted (POST), Flask reads the username using request.form.get(). 4. The username is printed on the terminal to simulate backend processing. 5. The server responds with a welcome message showing the submitted username.   **Output:** | | | | | | Week5 - Tuesday |  |