

# SEM LAB EXAM

Name: P. Umesh Reddy

Roll.no: 2403A510F9

Batch: 06

Course: AI Assisted coding

1Q. Create responsive student portal

## **Task 1: Use AI to Generate HTML Structure**

### **Prompt:**

Generate an HTML structure for a responsive student portal that includes:

- A header with a logo and navigation links (Home, Courses, Schedule, Profile, Logout)
- A dashboard section showing student name, notifications, and shortcuts
- Main content area for displaying dynamic content like assignments, grades, or announcements
- A sidebar with additional links.  
(e.g., Settings, Help)
- A footer with copyright

### **Task 1 Code:**

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Student Portal</title>
7      <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10     <header>
11         <div class="logo">StudentPortal</div>
12         <nav>
13             <ul>
14                 <li><a href="#">Home</a></li>
15                 <li><a href="#">Courses</a></li>
16                 <li><a href="#">Schedule</a></li>
17                 <li><a href="#">Profile</a></li>
18                 <li><a href="#">Logout</a></li>
19             </ul>
20         </nav>
21     </header>
22     <div class="container">
23         <aside>
24             <ul>
25                 <li><a href="#">Settings</a></li>
26                 <li><a href="#">Help</a></li>
27             </ul>
28         </aside>
29
30         <main>
31             <section class="dashboard">
32                 <h2>Welcome, [Student Name]</h2>
33                 <div class="notifications">
34                     <h3>Notifications</h3>
35                     <ul>
36                         <li>No new notifications</li>
37                     </ul>
38                 </div>
39                 <div class="shortcuts">
40                     <button>Assignments</button>
41                     <button>Grades</button>
42                     <button>Announcements</button>
43                 </div>
44             </section>
45             <section class="dynamic-content">
46                 <!-- Dynamic content goes here -->
47             </section>
48         </main>
49     </div>
50     <footer>
51         &copy; 2025 Student Portal. All rights reserved.
52     </footer>
53 </body>
54 </html>

```

## Explanation

- **Header:** Contains a logo and navigation menu for primary sections.
- **Sidebar (aside):** Supports secondary links/settings.
- **Dashboard (main):** Shows user info, notifications, and quick links.

- **Dynamic Content Area:** Place for displaying assignments, grades, or other page-specific info.
- **Footer:** Standard legal and copyright.

## Task 2: Generate CSS Grid Layout

### Prompt

Generate a CSS layout using grid for the portal above that:

- Makes the sidebar and main area responsive (collapses on mobile, sidebar on top or hidden)
- Ensures the header and footer are fixed at top/bottom, when possible
- Uses modern, clean design

### Task 2 Code:

```

1  /* styles.css */
2  :root {
3      --primary-color: #0044cc;
4      --secondary-color: #fff;
5      --accent-color: #f4f4f9;
6      --text-color: #222;
7  }
8
9  * {
10     box-sizing: border-box;
11     margin: 0;
12     padding: 0;
13 }
14
15 body, html {
16     height: 100%;
17     font-family: Arial, sans-serif;
18     background: var(--accent-color);
19     color: var(--text-color);
20 }
21
22 header, footer {
23     background: var(--primary-color);
24     color: var(--secondary-color);
25     padding: 1rem;
26 }
27
28 .logo {
29     font-weight: bold;
30     font-size: 1.5rem;
31     display: inline-block;
32 }

```

styles.css > ...

```
33
34   nav ul {
35     display: flex;
36     list-style: none;
37     gap: 1rem;
38     float: right;
39   }
40
41   nav a {
42     color: var(--secondary-color);
43     text-decoration: none;
44     font-weight: 500;
45   }
46
47   footer {
48     text-align: center;
49   }
50
51   .container {
52     display: grid;
53     grid-template-columns: 220px 1fr;
54     gap: 2rem;
55     padding: 2rem;
56     min-height: 75vh;
57   }
58
59   aside ul {
60     list-style: none;
61     background: #e7e7ef;
62     border-radius: 6px;
63     padding: 1rem;
64   }
```

styles.css > ...

```
65
66   aside a {
67     color: var(--primary-color);
68     text-decoration: none;
69     display: block;
70     margin-bottom: 1rem;
71   }
72
73   main {
74     background: var(--secondary-color);
75     border-radius: 6px;
76     padding: 2rem;
77     box-shadow: 0 2px 6px #ddd;
78   }
79
80   .dashboard {
81     margin-bottom: 2rem;
82   }
83
84   .shortcuts button {
85     margin-right: 1rem;
86     background: var(--primary-color);
87     color: var(--secondary-color);
88     border: none;
89     padding: 0.5rem 1rem;
90     margin-top: 1rem;
91     border-radius: 4px;
92     cursor: pointer;
93   }
94
```

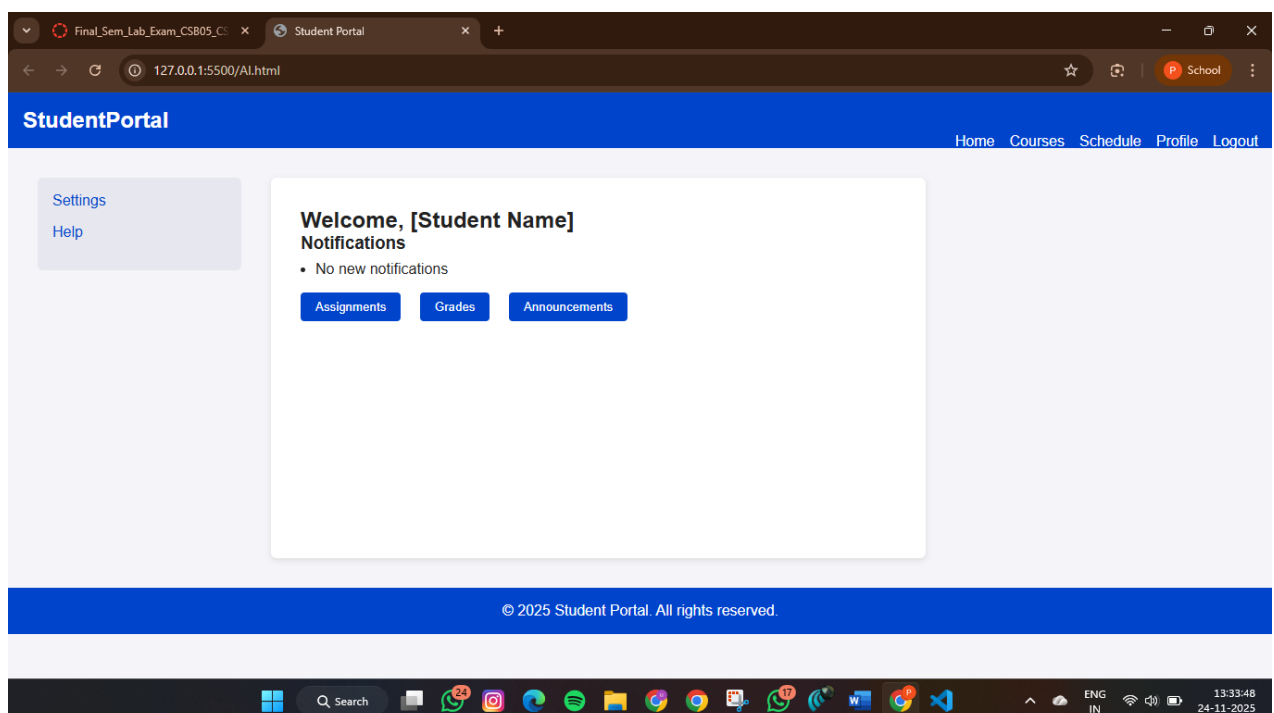
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```

94
95 .notifications ul {
96     margin-left: 1rem;
97     margin-top: 0.5rem;
98 }
99
100 @media (max-width: 900px) {
101     .container {
102         grid-template-columns: 1fr;
103         gap: 1rem;
104         padding: 1rem;
105     }
106     aside {
107         order: 2;
108     }
109     main {
110         order: 1;
111     }
112 }
113
114 @media (max-width: 600px) {
115     header, footer {
116         text-align: center;
117         padding: 0.5rem;
118     }
119     nav ul {
120         flex-direction: column;
121         gap: 0.5rem;
122         float: none;
123     }
124     .container {
125         padding: 0.5rem;
126     }
127     aside {
128         display: none;
129     }
130 }
131

```

## Output:



## Explanation

- **Grid:** Desktop layout uses a two-column grid. Collapses on smaller screens.
- **Responsiveness:** Media queries adapt layout for smaller screens—sidebar hides on mobile.
- **Fixed Header/Footer:** Ensured header and footer maintain visibility at the ends.
- **Design:** Modern palette, rounded corners, shadow, button accenting.

Q2: Add interactive JS

- Task 1: Use AI to write form validation logic.

&

- Task 2: Add event listeners for UI elements.

## Prompt :

If you want future integration with an LLM API (like OpenAI or Perplexity's API), replace the validate Form function's logic with an API call based on the user's email/password inputs, then process the response similarly.

## Code For both the task :

Alcoding.html > html > head > meta

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Interactive Form Validation</title>
7 </head>
8 <body>
9   <form id="signupForm">
10     <label>
11       Email:
12       <input type="email" id="email" required />
13     </label>
14     <label>
15       Password:
16       <input type="password" id="password" required minlength="6" />
17     </label>
18     <button type="submit">Sign Up</button>
19     <div id="formMessage"></div>
20   </form>
21
```

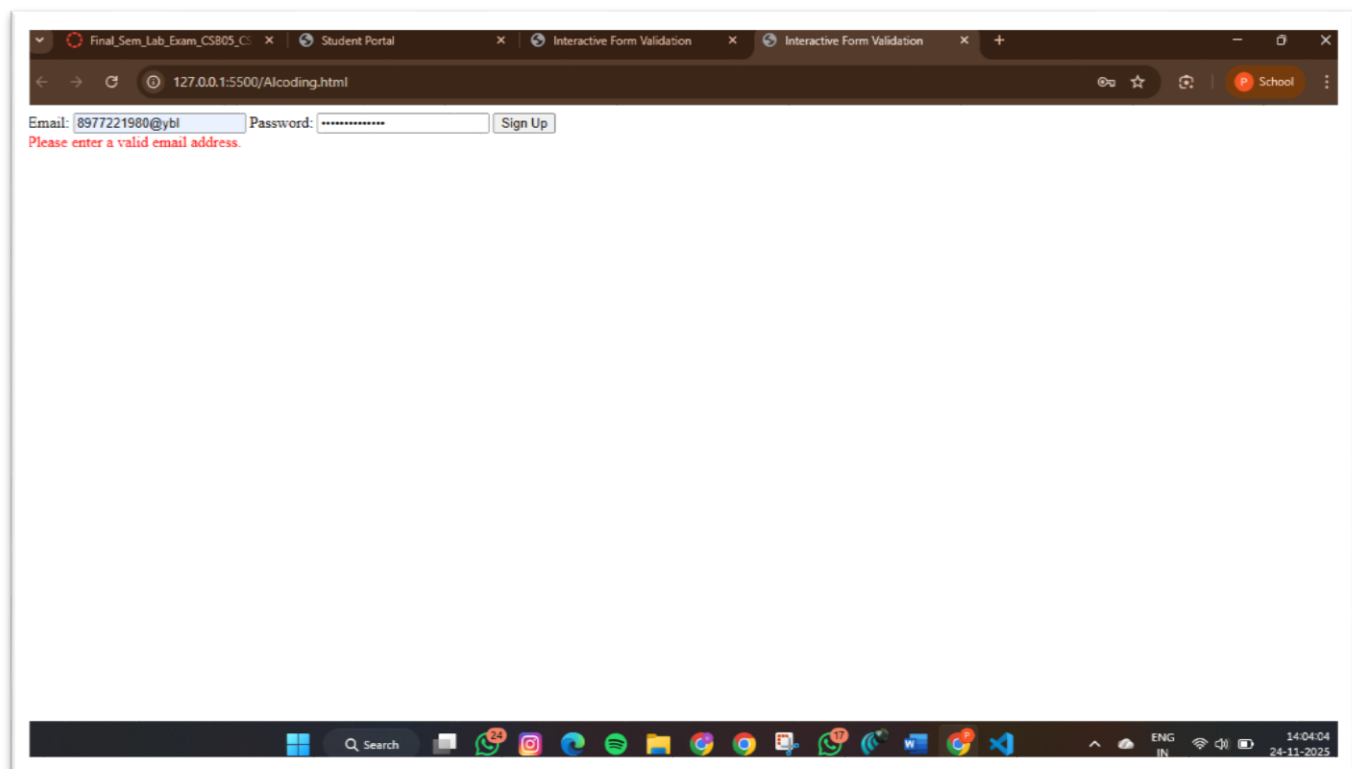
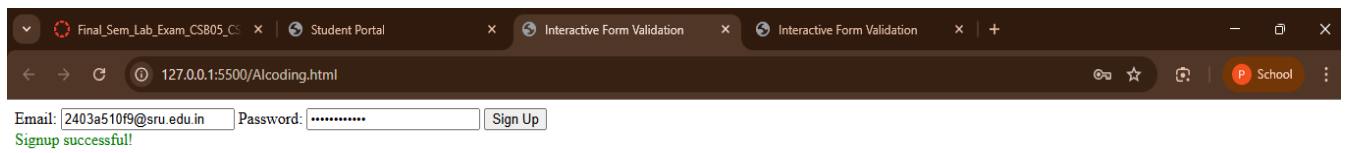
```
22 <script>
23   // Validation logic
24   function validateForm(email, password) {
25     const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
26     if (!emailPattern.test(email)) {
27       return { valid: false, message: "Please enter a valid email address." };
28     }
29     if (password.length < 6) {
30       return { valid: false, message: "Password must be at least 6 characters." };
31     }
32     if (email.endsWith("@example.com")) {
33       return { valid: false, message: "Please use your real email address." };
34     }
35     return { valid: true };
36   }

```

```
37   // Event listener
38   document.getElementById("signupForm").addEventListener("submit", function (event) {
39     event.preventDefault();
40     const email = document.getElementById("email").value;
41     const password = document.getElementById("password").value;
42     const result = validateForm(email, password);
43     const formMessage = document.getElementById("formMessage");
44     if (result.valid) {
45       formMessage.style.color = "green";
46       formMessage.textContent = "Signup successful!";
47     } else {
48       formMessage.style.color = "red";
49       formMessage.textContent = result.message;
50     }
51   });
52 </script>
53 </body>
54 </html>
--
```

Output:

Signup successful! :-





Explanation:

- **Form validation** uses regular expressions for email and checks password length.
- **Event listener** captures form submission, runs validation logic, and updates the message below the form interactively.
- For a real AI-backed app, connect this flow to an AI/LLM API for more advanced validation, suggestions, or even sentiment feedback