

## ASSIGNMENT – 14.2

NAME : S RUCHITHA

HALLTICKET NO : 2403A52316

BATCH NUMBER : 01

COURSE CODE : 24CS002PC215

PROGRAM NAME : B.TECH

YEAR/SEM : 2ND AND 3<sup>RD</sup>

### 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" that includes a <header>, <nav>, and <footer>. Ensure the code

has clean indenta on, proper tags, and includes comments for clarity.

## 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" />
6   <title>Student Info Portal</title>
7   <!-- Basic styles for clarity; remove or replace with external CSS as needed -->
8   <style>
9     body {
10       font-family: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
11       margin: 0;
12       line-height: 1.5;
13       color: #222;
14       background: #f7f9fc;
15     }
16     header {
17       background: #f0f5ff;
18       color: #fff;
19       padding: 1.25rem 1rem;
20     }
21     .container {
22       max-width: 1000px;
23       margin: 0 auto;
24       padding: 0 1rem;
25     }
26     nav {
27       margin-top: 0.5rem;
28     }
29     nav ul {
30       list-style: none;
31       padding: 0;
32       margin: 0;
33       display: flex;
34       gap: 0.75rem;
35       flex-wrap: wrap;
36     }
37     nav a {
38       color: #00a2f2;
39       text-decoration: none;
40       padding: 0.5rem 0.75rem;
41       border-radius: 4px;
42     }
43     nav a:hover, nav a:focus {
```

```
2 <html lang="en">
79 <body>
100   <main class="container" role="main">
108     <section aria-labelledby="quick-links-heading" style="margin-top:1rem;">
110       <div class="grid">
123         <!-- Card: Resources -->
124         <article class="card" id="resources" aria-labelledby="resources-heading">
125           <h4 id="resources-heading">Resources</h4>
126           <p class="muted">Academic calendars, forms, and helpful links for students and staff.</p>
127         </article>
128         <!-- Card: Contact / Support -->
129         <article class="card" id="contact" aria-labelledby="contact-heading">
130           <h4 id="contact-heading">Contact</h4>
131           <p class="muted">Get in touch with administration or technical support.</p>
132         </article>
133       </div>
134     </section>
135   </main>
136
137   <!-- Footer with basic site info -->
138   <footer>
139     <div class="container">
140       <!-- Footer content and small print -->
141       <div style="display:flex; justify-content:space-between; align-items:center; gap:1rem; flex-wrap:wrap;">
142         <div>
143           <strong>Student Info Portal</strong>
144           <div class="muted">Managed by the Registrar's Office</div>
145         </div>
146         <div class="muted" style="text-align:right;">
147           <div>Contact: registrar@example.edu</div>
148           <div>&copy; <span id="year"></span> Example University</div>
149         </div>
150       </div>
151     </div>
152   </footer>
153
154   <!-- Small script to set current year in footer -->
155   <script>
156     document.getElementById("year").textContent = new Date().getFullYear();
157   </script>
158 </body>
159 </html>
```

## OUTPUT:

- HTML code with <header>, <nav>, <footer>.
- Clean indentation, proper tags, and comments.

## 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 modern CSS styling to Task #1 homepage with a responsive navigation bar (flex or grid), a centered main content section, and a footer with a light gray background. Make sure it’s mobilefriendly and visually clean.”

## CODE:

```
1 /* Task #1 - Modern responsive styling for homepage */
2 /* Save as styles.css and pair with semantic HTML: header.site-header, nav.nav-links (ul>li>a),
3    main.site-main (with .container), and footer.site-footer. */
4
5 :root{
6   --bg: #ffffff;
7   --surface: #ffffff;
8   --muted-bg: #f3f4f6; /* light gray footer */
9   --text: #000000;
10  --muted: #6b728b;
11  --accent: #2563eb;
12  --accent-500: #1ed761;
13  --radius: 10px;
14  --gap: 1rem;
15  --max-width: 1100px;
16  --header-height: 72px;
17
18
19  /*[box-sizing:border-box]
20  html,body{height:100%}
21  body{
22    margin:0;
23    font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
24    background:var(--bg);
25    color:var(--text);
26    -webkit-font-smoothing:antialiased;
27    -moz-osx-font-smoothing:grayscale;
28    line-height:1.4;
29    -webkit-tap-highlight-color: transparent;
30  }
31
32  /* Container utility */
33  .container{
34    width:100%;
35    max-width:var(--max-width);
36    margin:auto;
37    padding:0 1.25rem;
38  }
39
40  /* Header / Nav */
41  .site-header{
42    position:relative;
43    display:flex;
```

```
200 @media (max-width:720px){
201
202   .nav-toggle{display:block}
203
204   /* Stack links vertically and make them full width for touch targets */
205   .nav-links{
206     flex-direction:column;
207     width:100%;
208     gap:0.25rem;
209     margin-top:0.5rem;
210   }
211
212   .nav-links a{
213     width:100%;
214     padding:0.75rem 0.9rem;
215     background:transparent;
216     justify-content:center;
217     font-weight:600;
218   }
219
220   .nav-cta{
221     margin-inline:auto;
222     width:100%;
223     text-align:center;
224   }
225
226   .hero{
227     padding:1.25rem;
228   }
229
230   .footer-inner{
231     flex-direction:column;
232     align-items:center;
233     gap:0.5rem;
234     text-align:center;
235   }
236
237 }
238
239 /* Respect reduced motion preference */
240 @media (prefers-reduced-motion: reduce){
241   *(transition:none!important)
242 }
```

OUTPUT:

- HTML + CSS combined.
- AI explains how CSS classes apply.
- AI refactors with with open() and try-except.

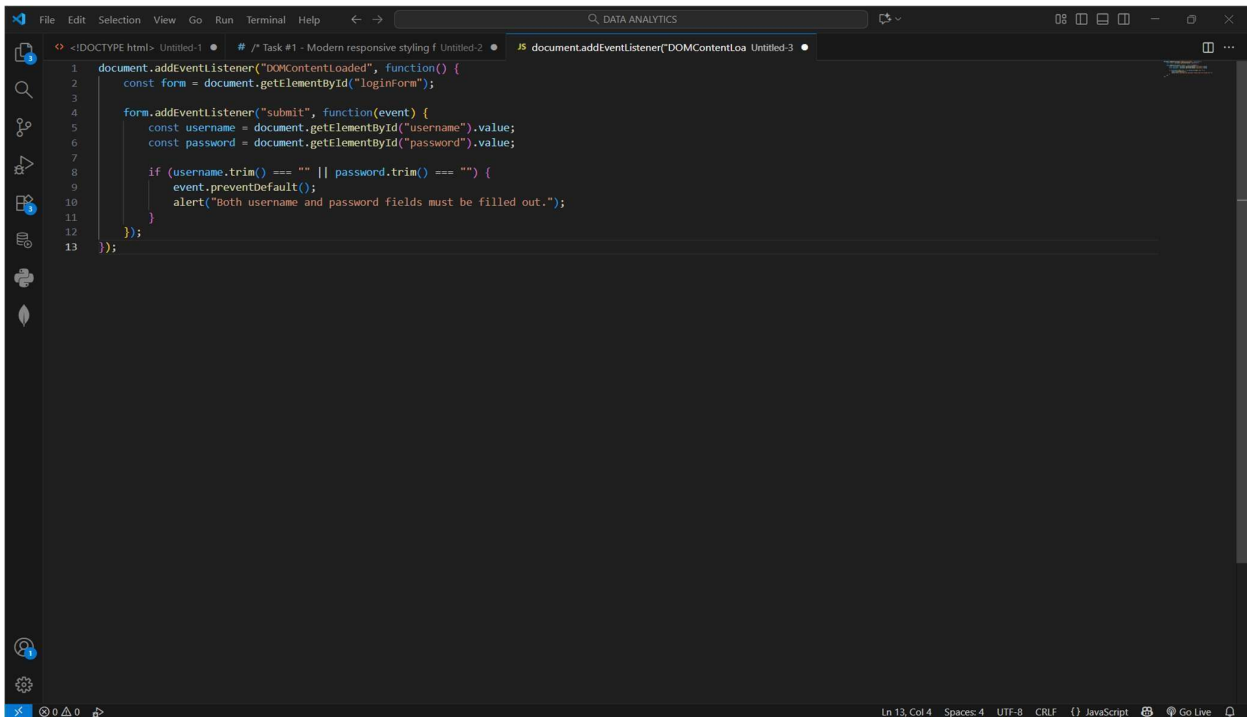
### TASK 3:

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, ensuring the username and password fields are not empty before submission.”

### CODE:

A screenshot of a code editor window with a dark theme. The editor has a menu bar at the top with options: File, Edit, Selection, View, Go, Run, Terminal, Help. Below the menu bar is a search bar and a toolbar with icons for file operations. The main editing area contains three tabs: '<IDOCTYPE html> Untitled-1', '# / Task #1 - Modern responsive styling f Untitled-2', and 'JS document.addEventListener("DOMContentLoaded" Untitled-3'. The active tab shows JavaScript code for form validation. The code is as follows:

```
1 document.addEventListener("DOMContentLoaded", function() {  
2   const form = document.getElementById("loginform");  
3  
4   form.addEventListener("submit", function(event) {  
5     const username = document.getElementById("username").value;  
6     const password = document.getElementById("password").value;  
7  
8     if (username.trim() === "" || password.trim() === "") {  
9       event.preventDefault();  
10      alert("Both username and password fields must be filled out.");  
11    }  
12  });  
13 });
```

The status bar at the bottom indicates 'Ln 13, Col 4', 'Spaces: 4', 'UTF-8', 'CRLF', '{} JavaScript', and 'Go Live'.

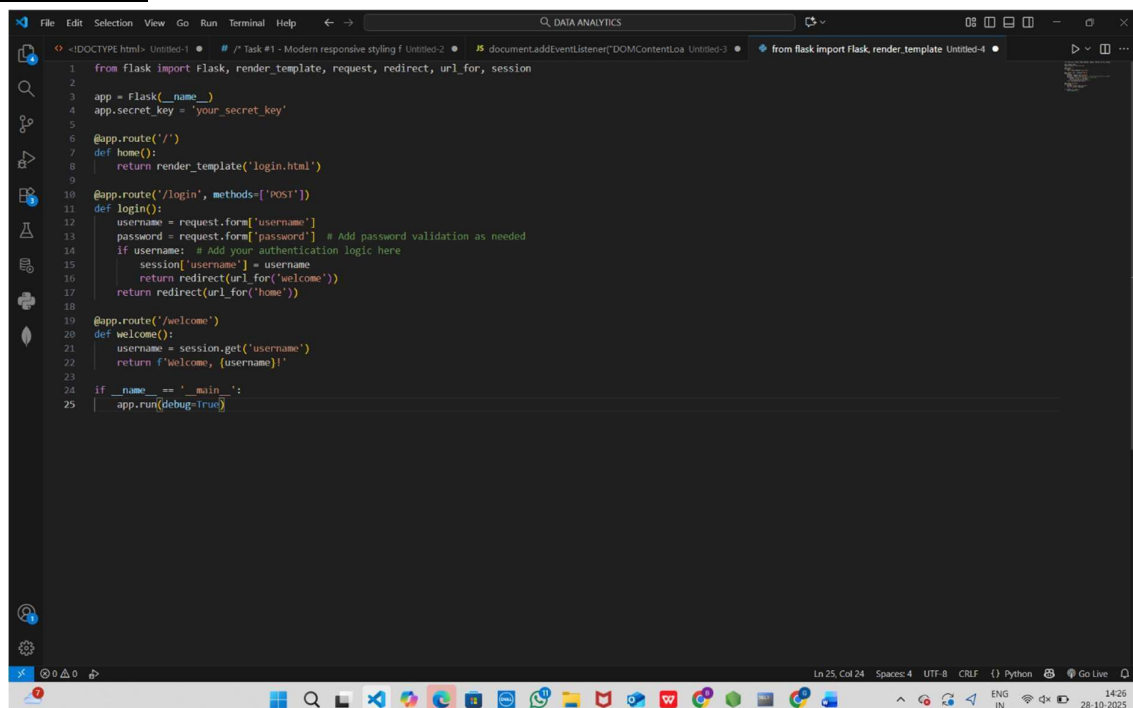
## OUTPUT:

- \*Working on submit JS validation.
- \*Clear error messages if inputs are empty.

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 app that serves the login HTML form from Task #3 and displays the username on a successful login.”

CODE:

A screenshot of a code editor window with a dark theme. The editor shows a Python Flask application. The code includes imports for Flask, render\_template, request, redirect, url\_for, and session. It defines a Flask app, sets a secret key, and creates three routes: a home route, a login route with POST method, and a welcome route. The login route handles user input, validates it, and redirects to the welcome route if successful. The welcome route retrieves the username from the session and returns a welcome message. The code is wrapped in an if \_\_name\_\_ == '\_\_main\_\_': block to run the app in debug mode.

```
1 from flask import Flask, render_template, request, redirect, url_for, session
2
3 app = Flask(__name__)
4 app.secret_key = 'your_secret_key'
5
6 @app.route('/')
7 def home():
8     return render_template('login.html')
9
10 @app.route('/login', methods=['POST'])
11 def login():
12     username = request.form['username']
13     password = request.form['password'] # Add password validation as needed
14     if username: # Add your authentication logic here
15         session['username'] = username
16         return redirect(url_for('welcome'))
17     return redirect(url_for('home'))
18
19 @app.route('/welcome')
20 def welcome():
21     username = session.get('username')
22     return f'Welcome, {username}!'
23
24 if __name__ == '__main__':
25     app.run(debug=True)
```

OUTPUT:

\*Working on submit JS valida on.

\*Clear error messages if inputs are empty