JS-2 Project

Learning Management System (LMS)

Generated on: February 10, 2026 at 01:36 PM

Complete Source Code Documentation

Table of Contents

1. Folder Structure

```
js-2/
■■■ index.html              # Login page
■■■ assignments.html        # Assignments management page
■■■ courses.html            # Courses list page
■■■ dashboard.html          # Dashboard with stats
■■■ profile.html            # User profile page
■■■ signup.html             # User registration page
■■■ README.md               # Project documentation
■■■ JS_IMPLEMENTATION_GUIDE.md   # JavaScript implementation guide
■■■ PROJECT_SUMMARY.md      # Complete project summary
■■■ netlify.toml            # Netlify configuration
■
■■■ css/
■   ■■■ main.css           # Global styles, variables, login page
■   ■■■ layout.css         # Layout, sidebar, responsive design
■   ■■■ component.css      # Component styles (cards, buttons, forms)
■
■■■ js/
■   ■■■ auth.js            # Login validation module
■   ■■■ assignments.js     # Assignment management module
■   ■■■ courses.js         # Course list rendering module
■   ■■■ dashboard.js       # Dashboard rendering module
■   ■■■ profile.js         # Profile edit module
■   ■■■ signup.js          # User registration module
■   ■■■ theme.js           # Theme switcher & sidebar toggle
■   ■■■ utils.js           # Utility helper functions
■
■■■ images/                # Image assets folder (empty)
```

2. HTML Files

File: index.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;LMS Login&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;main class="login-container"&gt;

        &lt;form class="login-form" id="loginForm"&gt;

            &lt;h1&gt;LMS Login Page&lt;/h1&gt;

            &lt;div class="form-group"&gt;

                &lt;label for="username"&gt;Username:&lt;/label&gt;

                &lt;input type="text" id="username" name="username" placeholder="Enter your username

                &lt;span class="error-message" id="usernameError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;div class="form-group"&gt;

                &lt;label for="password"&gt;Password:&lt;/label&gt;

                &lt;input type="password" id="password" name="password" placeholder="Enter your pass

                &lt;span class="error-message" id="passwordError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;button id="loginButton" type="submit" class="btn btn-primary"&gt;Login&lt;/button&gt

            &lt;div style="text-align: center; margin-top: 1rem;"&gt;

                &lt;p style="color: var(--text-light);"&gt;Don't have an account? &lt;a href="signup
            &lt;/div&gt;

            &lt;p&gt;Username : guest&lt;/p&gt; &lt;p&gt;Password : 1234&lt;/p&gt;&lt;/p&gt;

        &lt;/form&gt;

    &lt;/main&gt;

    &lt;script src="js/auth.js"&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

File: dashboard.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;LMS Dashboard&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;div class="app-container"&gt;

        &lt;!-- Sidebar Navigation --&gt;

        &lt;aside class="sidebar" id="sidebar"&gt;

            &lt;div class="sidebar-header"&gt;

                &lt;h2&gt;LMS Portal&lt;/h2&gt;

            &lt;/div&gt;

            &lt;nav class="sidebar-nav"&gt;

                &lt;ul&gt;

                    &lt;li&gt;&lt;a href="dashboard.html" class="nav-link" data-page="dashboard"&gt;

                    &lt;li&gt;&lt;a href="courses.html" class="nav-link" data-page="courses"&gt;Cour

                    &lt;li&gt;&lt;a href="assignments.html" class="nav-link" data-page="assignments"

                    &lt;li&gt;&lt;a href="profile.html" class="nav-link" data-page="profile"&gt;Prof

                    &lt;li&gt;&lt;a href="index.html" class="nav-link"&gt;Logout&lt;/a&gt;&lt;/li&gt

                &lt;/ul&gt;

            &lt;/nav&gt;

        &lt;/aside&gt;

        &lt;!-- Sidebar overlay for mobile --&gt;

        &lt;div class="sidebar-overlay" id="sidebarOverlay"&gt;&lt;/div&gt;

        &lt;!-- Main Content Area --&gt;
        &lt;main class="main-content"&gt;
            &lt;!-- Top Bar --&gt;
            &lt;div class="top-bar"&gt;
                &lt;button class="menu-toggle" id="menuToggle" aria-label="Toggle Menu"&gt;■&lt;/but
                &lt;h1&gt;Welcome, &lt;span id="userName"&gt;Student&lt;/span&gt;!&lt;/h1&gt;
                &lt;button class="theme-toggle" id="themeToggle" aria-label="Toggle Theme"&gt;■&lt;/
            &lt;/div&gt;
            &lt;!-- Dashboard Content --&gt;
            &lt;div class="dashboard-container"&gt;
                &lt;!-- Stats Cards Grid - Will be populated by JavaScript --&gt;
                &lt;div class="stats-grid" id="statsGrid"&gt;
                    &lt;!-- Dashboard stat cards will be inserted here by dashboard.js --&gt;
                &lt;/div&gt;
                &lt;!-- Upcoming Assignments Section --&gt;
                &lt;div class="dashboard-section"&gt;
                    &lt;h2&gt;Upcoming Assignments&lt;/h2&gt;
                    &lt;div class="assignments-preview" id="assignmentsPreview"&gt;
                        &lt;!-- Assignment preview items will be inserted here by dashboard.js --&gt
                    &lt;/div&gt;
                &lt;/div&gt;
            &lt;/div&gt;
        &lt;/main&gt;
    &lt;/div&gt;
    &lt;script src="js/dashboard.js"&gt;&lt;/script&gt;
    &lt;script src="js/theme.js"&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
```

File: courses.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;My Courses - LMS&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;div class="app-container"&gt;

        &lt;!-- Sidebar Navigation --&gt;

        &lt;aside class="sidebar" id="sidebar"&gt;

            &lt;div class="sidebar-header"&gt;

                &lt;h2&gt;LMS Portal&lt;/h2&gt;

            &lt;/div&gt;

            &lt;nav class="sidebar-nav"&gt;

                &lt;ul&gt;

                    &lt;li&gt;&lt;a href="dashboard.html" class="nav-link" data-page="dashboard"&gt;

                    &lt;li&gt;&lt;a href="courses.html" class="nav-link" data-page="courses"&gt;Cour

                    &lt;li&gt;&lt;a href="assignments.html" class="nav-link" data-page="assignments"

                    &lt;li&gt;&lt;a href="profile.html" class="nav-link" data-page="profile"&gt;Prof

                    &lt;li&gt;&lt;a href="index.html" class="nav-link"&gt;Logout&lt;/a&gt;&lt;/li&gt

                &lt;/ul&gt;

            &lt;/nav&gt;

        &lt;/aside&gt;

        &lt;!-- Sidebar overlay for mobile --&gt;

        &lt;div class="sidebar-overlay" id="sidebarOverlay"&gt;&lt;/div&gt;

        &lt;!-- Main Content Area --&gt;
        &lt;main class="main-content"&gt;

            &lt;!-- Top Bar --&gt;

            &lt;div class="top-bar"&gt;

                &lt;button class="menu-toggle" id="menuToggle" aria-label="Toggle Menu"&gt;■&lt;/but

                &lt;h1&gt;My Courses&lt;/h1&gt;

                &lt;button class="theme-toggle" id="themeToggle" aria-label="Toggle Theme"&gt;■&lt;/

            &lt;/div&gt;

            &lt;!-- Courses Content --&gt;

            &lt;div class="dashboard-container"&gt;

                &lt;div class="dashboard-section"&gt;

                    &lt;h2&gt;Enrolled Courses&lt;/h2&gt;

                    &lt;p&gt;Below are all the courses you are currently enrolled in. Click on a cou

                &lt;/div&gt;

                &lt;!-- Course Cards Grid - Will be populated by JavaScript --&gt;

                &lt;div class="course-grid" id="courseGrid"&gt;

                    &lt;!-- Course cards will be inserted here by courses.js --&gt;

                &lt;/div&gt;

            &lt;/div&gt;

        &lt;/main&gt;

    &lt;/div&gt;

    &lt;script src="js/courses.js"&gt;&lt;/script&gt;

    &lt;script src="js/theme.js"&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

File: assignments.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;Assignments - LMS&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;div class="app-container"&gt;

        &lt;!-- Sidebar Navigation --&gt;

        &lt;aside class="sidebar" id="sidebar"&gt;

            &lt;div class="sidebar-header"&gt;

                &lt;h2&gt;LMS Portal&lt;/h2&gt;

            &lt;/div&gt;

            &lt;nav class="sidebar-nav"&gt;

                &lt;ul&gt;

                    &lt;li&gt;&lt;a href="dashboard.html" class="nav-link" data-page="dashboard"&gt;

                    &lt;li&gt;&lt;a href="courses.html" class="nav-link" data-page="courses"&gt;Cour

                    &lt;li&gt;&lt;a href="assignments.html" class="nav-link" data-page="assignments"

                    &lt;li&gt;&lt;a href="profile.html" class="nav-link" data-page="profile"&gt;Prof

                    &lt;li&gt;&lt;a href="index.html" class="nav-link"&gt;Logout&lt;/a&gt;&lt;/li&gt

                &lt;/ul&gt;

            &lt;/nav&gt;

        &lt;/aside&gt;

        &lt;!-- Sidebar overlay for mobile --&gt;

        &lt;div class="sidebar-overlay" id="sidebarOverlay"&gt;&lt;/div&gt;

        &lt;!-- Main Content Area --&gt;
        &lt;main class="main-content"&gt;

            &lt;!-- Top Bar --&gt;

            &lt;div class="top-bar"&gt;

                &lt;button class="menu-toggle" id="menuToggle" aria-label="Toggle Menu"&gt;■&lt;/but

                &lt;h1&gt;My Assignments&lt;/h1&gt;

                &lt;button class="theme-toggle" id="themeToggle" aria-label="Toggle Theme"&gt;■&lt;/

            &lt;/div&gt;

            &lt;!-- Assignments Content --&gt;

            &lt;div class="dashboard-container"&gt;

                &lt;div class="dashboard-section"&gt;

                    &lt;h2&gt;All Assignments&lt;/h2&gt;

                    &lt;p&gt;View all your assignments, their status, and due dates. Click "Submit"

                &lt;/div&gt;

                &lt;!-- Assignments List - Will be populated by JavaScript --&gt;

                &lt;div class="assignments-list" id="assignmentsList"&gt;

                    &lt;!-- Assignment items will be inserted here by assignments.js --&gt;

                &lt;/div&gt;

            &lt;/div&gt;

        &lt;/main&gt;

    &lt;/div&gt;

    &lt;script src="js/assignments.js"&gt;&lt;/script&gt;

    &lt;script src="js/theme.js"&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

File: profile.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;My Profile - LMS&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;div class="app-container"&gt;

        &lt;!-- Sidebar Navigation --&gt;

        &lt;aside class="sidebar" id="sidebar"&gt;

            &lt;div class="sidebar-header"&gt;

                &lt;h2&gt;LMS Portal&lt;/h2&gt;

            &lt;/div&gt;

            &lt;nav class="sidebar-nav"&gt;

                &lt;ul&gt;

                    &lt;li&gt;&lt;a href="dashboard.html" class="nav-link" data-page="dashboard"&gt;

                    &lt;li&gt;&lt;a href="courses.html" class="nav-link" data-page="courses"&gt;Cour

                    &lt;li&gt;&lt;a href="assignments.html" class="nav-link" data-page="assignments"

                    &lt;li&gt;&lt;a href="profile.html" class="nav-link" data-page="profile"&gt;Prof

                    &lt;li&gt;&lt;a href="index.html" class="nav-link"&gt;Logout&lt;/a&gt;&lt;/li&gt

                &lt;/ul&gt;

            &lt;/nav&gt;

        &lt;/aside&gt;

        &lt;!-- Sidebar overlay for mobile --&gt;

        &lt;div class="sidebar-overlay" id="sidebarOverlay"&gt;&lt;/div&gt;

        &lt;!-- Main Content Area --&gt;
```

```
&lt;main class="main-content"&gt;

    &lt;!-- Top Bar --&gt;

    &lt;div class="top-bar"&gt;

        &lt;button class="menu-toggle" id="menuToggle" aria-label="Toggle Menu"&gt;■&lt;/but

        &lt;h1&gt;My Profile&lt;/h1&gt;

        &lt;button class="theme-toggle" id="themeToggle" aria-label="Toggle Theme"&gt;■&lt;/

    &lt;/div&gt;

    &lt;!-- Profile Content --&gt;

    &lt;div class="dashboard-container"&gt;

        &lt;div class="profile-container"&gt;

            &lt;div class="profile-card"&gt;

                &lt;!-- Profile Header --&gt;

                &lt;div class="profile-header"&gt;

                    &lt;div class="profile-avatar" id="profileAvatar"&gt;

                        S

                    &lt;/div&gt;

                    &lt;div class="profile-info"&gt;

                        &lt;h2 id="profileName"&gt;Student Name&lt;/h2&gt;

                        &lt;p id="profileEmail"&gt;student@university.edu&lt;/p&gt;

                    &lt;/div&gt;

                &lt;/div&gt;

                &lt;!-- Profile Form --&gt;

                &lt;form class="profile-form" id="profileForm"&gt;

                    &lt;div class="form-group"&gt;

                        &lt;label for="fullName"&gt;Full Name&lt;/label&gt;

                        &lt;input type="text" id="fullName" name="fullName" value="John Doe"

                    &lt;/div&gt;

                    &lt;div class="form-group"&gt;

                        &lt;label for="email"&gt;Email Address&lt;/label&gt;

                        &lt;input type="email" id="email" name="email" value="john.doe@unive

                    &lt;/div&gt;

                    &lt;div class="form-group"&gt;

                        &lt;label for="studentId"&gt;Student ID&lt;/label&gt;

                        &lt;input type="text" id="studentId" name="studentId" value="STU1234
                        &lt;/div&gt;
                        &lt;div class="form-group"&gt;
                            &lt;label for="department"&gt;Department&lt;/label&gt;
                            &lt;input type="text" id="department" name="department" value="Compu
                        &lt;/div&gt;
                        &lt;div class="form-group"&gt;
                            &lt;label for="phone"&gt;Phone Number&lt;/label&gt;
                            &lt;input type="tel" id="phone" name="phone" value="+1234567890" dis
                        &lt;/div&gt;
                        &lt;div class="form-group"&gt;
                            &lt;label for="year"&gt;Academic Year&lt;/label&gt;
                            &lt;input type="text" id="year" name="year" value="3rd Year" disable
                        &lt;/div&gt;
                        &lt;!-- Form Actions --&gt;
                        &lt;div class="form-actions"&gt;
                            &lt;button type="button" id="editButton" class="btn btn-primary"&gt;
                            &lt;button type="submit" id="saveButton" class="btn btn-success" sty
                            &lt;button type="button" id="cancelButton" class="btn btn-danger" st
                        &lt;/div&gt;
                    &lt;/form&gt;
                &lt;/div&gt;
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/main&gt;
&lt;/div&gt;
&lt;script src="js/profile.js"&gt;&lt;/script&gt;
&lt;script src="js/theme.js"&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
```

File: signup.html

```
&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;Sign Up - LMS&lt;/title&gt;

    &lt;link rel="stylesheet" href="css/main.css"&gt;

    &lt;link rel="stylesheet" href="css/layout.css"&gt;

    &lt;link rel="stylesheet" href="css/component.css"&gt;

&lt;/head&gt;

&lt;body&gt;

    &lt;!-- Alert container for notifications --&gt;

    &lt;div id="alertContainer"&gt;&lt;/div&gt;

    &lt;main class="login-container"&gt;

        &lt;form class="login-form" id="signupForm"&gt;

            &lt;h1&gt;Create Account&lt;/h1&gt;

            &lt;div class="form-group"&gt;

                &lt;label for="signupUsername"&gt;Username:&lt;/label&gt;

                &lt;input type="text" id="signupUsername" name="username" placeholder="Choose a user

                &lt;span class="error-message" id="signupUsernameError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;div class="form-group"&gt;

                &lt;label for="signupEmail"&gt;Email:&lt;/label&gt;

                &lt;input type="email" id="signupEmail" name="email" placeholder="Enter your email"&

                &lt;span class="error-message" id="signupEmailError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;div class="form-group"&gt;

                &lt;label for="signupPassword"&gt;Password:&lt;/label&gt;

                &lt;input type="password" id="signupPassword" name="password" placeholder="Create a

                &lt;span class="error-message" id="signupPasswordError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;div class="form-group"&gt;
```

```
                &lt;label for="confirmPassword"&gt;Confirm Password:&lt;/label&gt;

                &lt;input type="password" id="confirmPassword" name="confirmPassword" placeholder="C

                &lt;span class="error-message" id="confirmPasswordError"&gt;&lt;/span&gt;

            &lt;/div&gt;

            &lt;button id="signupButton" type="submit" class="btn btn-primary"&gt;Create Account&lt;

            &lt;div style="text-align: center; margin-top: 1rem;"&gt;

                &lt;p style="color: var(--text-light);"&gt;Already have an account? &lt;a href="inde

            &lt;/div&gt;

        &lt;/form&gt;

    &lt;/main&gt;

    &lt;script src="js/signup.js"&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

3. CSS Files

File: css/main.css

```css
/* @import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100..900;1,100..900&

 */

* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

:root {

  --primary-color: #4f46e5;

  --primary-hover: #4338ca;

  --secondary-color: #06b6d4;

  --success-color: #10b981;

  --danger-color: #ef4444;

  --warning-color: #f59e0b;

  --dark-bg: #1e293b;

  --light-bg: #f8fafc;

  --text-dark: #0f172a;

  --text-light: #64748b;

  --border-color: #e2e8f0;

  --shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);

  --shadow-lg: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);

}

body {

  font-family: 'Montserrat', sans-serif;

  background: linear-gradient(135deg, #134e5e 0%, #71b280 100%);

  color: var(--text-dark);

  line-height: 1.6;

  min-height: 100vh;

}
```

```
.login-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  padding: 20px;
}
.login-container h1 {
  text-align: center;
  color: var(--dark-bg);
  font-size: 2rem;
  font-weight: 700;
  margin-bottom: 2rem;
}
.login-form {
  background: white;
  padding: 3rem;
  border-radius: 16px;
  box-shadow: var(--shadow-lg);
  width: 100%;
  max-width: 450px;
  animation: fadeIn 0.5s ease-in;
}
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(-20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
.login-form .form-group {
```

```css
  margin-bottom: 1.5rem;

}

.login-form .form-group:last-of-type {

  margin-bottom: 2rem;

}

.login-form label {

  display: block;

  font-weight: 600;

  color: var(--text-dark);

  margin-bottom: 0.5rem;

  font-size: 0.95rem;

}

.login-form input {

  width: 100%;

  padding: 0.875rem 1rem;

  border: 2px solid var(--border-color);

  border-radius: 8px;

  font-size: 1rem;

  font-family: 'Montserrat', sans-serif;

  transition: all 0.3s ease;

  background-color: var(--light-bg);

}

.login-form input:focus {

  outline: none;

  border-color: var(--primary-color);

  background-color: white;

  box-shadow: 0 0 0 3px rgba(79, 70, 229, 0.1);

}

.login-form button {

  width: 100%;

  padding: 1rem;

  background: linear-gradient(135deg, var(--primary-color), var(--secondary-color));

  color: white;

  border: none;
  border-radius: 8px;

  font-size: 1rem;

  font-weight: 600;

  font-family: 'Montserrat', sans-serif;

  cursor: pointer;

  transition: all 0.3s ease;

  box-shadow: var(--shadow);

}

.login-form button:hover {

  transform: translateY(-2px);

  box-shadow: var(--shadow-lg);

}

.login-form button:active {

  transform: translateY(0);

}

.login-form button:disabled {

  background: var(--text-light);

  cursor: not-allowed;

  transform: none;

}

.error-message {

  color: var(--danger-color);

  font-size: 0.875rem;

  margin-top: 0.5rem;

  display: none;

}

.error-message.show {

  display: block;

}

/* ========== Utility Classes ========== */

.text-center {

  text-align: center;

}

.mt-1 { margin-top: 0.5rem; }
```

```css
.mt-2 { margin-top: 1rem; }

.mt-3 { margin-top: 1.5rem; }

.mb-1 { margin-bottom: 0.5rem; }

.mb-2 { margin-bottom: 1rem; }

.mb-3 { margin-bottom: 1.5rem; }

.hidden {

  display: none;

}

.visible {

  display: block;

}
```

File: css/layout.css

```
/* ========== Layout Styles ========== */

/* App Container */

.app-container {

  display: flex;

  min-height: 100vh;

  position: relative;

}

/* Sidebar Styles */

.sidebar {

  width: 250px;

  background-color: #2d3748;

  color: white;

  position: fixed;

  top: 0;

  left: 0;

  height: 100vh;

  overflow-y: auto;

  transition: transform 0.3s ease;

  z-index: 1000;

}

.sidebar-header {

  padding: 1.5rem;

  border-bottom: 1px solid rgba(255, 255, 255, 0.1);

}

.sidebar-header h2 {

  font-size: 1.5rem;

  font-weight: 700;

  margin: 0;

}

.sidebar-nav ul {

  list-style: none;

  padding: 1rem 0;
  margin: 0;

}

.sidebar-nav li {

  margin: 0;

}

.sidebar-nav a {

  display: block;

  padding: 1rem 1.5rem;

  color: #cbd5e0;

  text-decoration: none;

  font-weight: 500;

  transition: all 0.3s ease;

}

.sidebar-nav a:hover {

  background-color: rgba(255, 255, 255, 0.1);

  color: white;

  padding-left: 2rem;

}

.sidebar-nav a.active {

  background-color: rgba(79, 70, 229, 0.3);

  color: #a5b4fc;

  border-left: 4px solid #4f46e5;

  padding-left: calc(1.5rem - 4px);

}

/* Main Content Area */

.main-content {

  flex: 1;

  margin-left: 250px;

  min-height: 100vh;

  background-color: rgba(255, 255, 255, 0.05);

}

/* Top Bar */

.top-bar {

  background-color: rgba(255, 255, 255, 0.95);
  padding: 1rem 2rem;

  border-bottom: 1px solid var(--border-color);

  display: flex;

  justify-content: space-between;

  align-items: center;

  position: sticky;

  top: 0;

  z-index: 100;

  box-shadow: var(--shadow);

}

.top-bar h1 {

  font-size: 1.5rem;

  color: var(--text-dark);

  margin: 0;

}

.menu-toggle {

  display: none;

  background: none;

  border: none;

  font-size: 1.5rem;

  cursor: pointer;

  color: var(--text-dark);

  padding: 0.5rem;

}

.theme-toggle {

  background: var(--primary-color);

  color: white;

  border: none;

  padding: 0.5rem 1rem;

  border-radius: 8px;

  cursor: pointer;

  font-size: 1.2rem;

  transition: all 0.3s ease;

}
```

```
.theme-toggle:hover {

  transform: scale(1.1);

}

/* Dashboard Container */

.dashboard-container {

  padding: 2rem;

  max-width: 1400px;

}

/* Mobile Responsive */

@media (max-width: 768px) {

  .sidebar {

    transform: translateX(-100%);

  }

  .sidebar.active {

    transform: translateX(0);

  }

  .main-content {

    margin-left: 0;

  }

  .menu-toggle {

    display: block;

  }

  .top-bar h1 {

    font-size: 1.2rem;

  }

  .dashboard-container {

    padding: 1rem;

  }

}

/* Sidebar Overlay for Mobile */

.sidebar-overlay {

  display: none;

  position: fixed;

  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  z-index: 999;
}
.sidebar-overlay.active {
  display: block;
}
@media (min-width: 769px) {
  .sidebar-overlay {
    display: none !important;
  }
}
```

File: css/component.css

```
/* ========== Component Styles ========== */

/* Dashboard Stats Grid */

.stats-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

  gap: 1.5rem;

  margin-bottom: 2rem;

}

.stat-card {

  background: white;

  padding: 1.5rem;

  border-radius: 12px;

  box-shadow: var(--shadow);

  transition: all 0.3s ease;

}

.stat-card:hover {

  transform: translateY(-5px);

  box-shadow: var(--shadow-lg);

}

.stat-card h3 {

  font-size: 0.875rem;

  color: var(--text-light);

  font-weight: 600;

  text-transform: uppercase;

  margin-bottom: 0.5rem;

  letter-spacing: 0.5px;

}

.stat-card .stat-value {

  font-size: 2rem;

  font-weight: 700;

  color: var(--text-dark);

  margin-bottom: 0.25rem;
```

```css
}

.stat-card .stat-label {

  font-size: 0.875rem;

  color: var(--text-light);

}

/* Dashboard Section */

.dashboard-section {

  background: white;

  padding: 2rem;

  border-radius: 12px;

  box-shadow: var(--shadow);

  margin-bottom: 2rem;

}

.dashboard-section h2 {

  font-size: 1.5rem;

  color: var(--text-dark);

  margin-bottom: 1.5rem;

  border-bottom: 2px solid var(--primary-color);

  padding-bottom: 0.5rem;

}

/* Assignments Preview */

.assignments-preview {

  display: flex;

  flex-direction: column;

  gap: 1rem;

}

.assignment-item {

  display: flex;

  justify-content: space-between;

  align-items: center;

  padding: 1rem;

  background-color: var(--light-bg);

  border-radius: 8px;

  border-left: 4px solid var(--primary-color);
  transition: all 0.3s ease;
}
.assignment-item:hover {
  background-color: #e2e8f0;
  transform: translateX(5px);
}
.assignment-info h4 {
  font-size: 1rem;
  color: var(--text-dark);
  margin-bottom: 0.25rem;
}
.assignment-info p {
  font-size: 0.875rem;
  color: var(--text-light);
  margin: 0;
}
.assignment-status {
  padding: 0.5rem 1rem;
  border-radius: 20px;
  font-size: 0.875rem;
  font-weight: 600;
}
.assignment-status.pending {
  background-color: #fef3c7;
  color: #92400e;
}
.assignment-status.submitted {
  background-color: #d1fae5;
  color: #065f46;
}
.assignment-status.late {
  background-color: #fee2e2;
  color: #991b1b;
}
```

```
/* Course Cards */

.course-grid {

  display: grid;

  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));

  gap: 1.5rem;

}

.course-card {

  background: white;

  border-radius: 12px;

  box-shadow: var(--shadow);

  overflow: hidden;

  transition: all 0.3s ease;

}

.course-card:hover {

  transform: translateY(-5px);

  box-shadow: var(--shadow-lg);

}

.course-header {

  padding: 1.5rem;

  background: linear-gradient(135deg, var(--primary-color), var(--secondary-color));

  color: white;

}

.course-header h3 {

  font-size: 1.25rem;

  margin-bottom: 0.5rem;

}

.course-header p {

  font-size: 0.875rem;

  opacity: 0.9;

  margin: 0;

}

.course-body {

  padding: 1.5rem;

}
```

```css
.course-info {

  display: flex;

  justify-content: space-between;

  margin-bottom: 1rem;

}

.course-info span {

  font-size: 0.875rem;

  color: var(--text-light);

}

.course-status {

  display: inline-block;

  padding: 0.25rem 0.75rem;

  border-radius: 20px;

  font-size: 0.75rem;

  font-weight: 600;

  text-transform: uppercase;

}

.course-status.active {

  background-color: #d1fae5;

  color: #065f46;

}

.course-status.completed {

  background-color: #dbeafe;

  color: #1e40af;

}

.course-progress {

  margin-top: 1rem;

}

.progress-bar {

  width: 100%;

  height: 8px;

  background-color: #e2e8f0;

  border-radius: 10px;

  overflow: hidden;
```

```
}

.progress-fill {

  height: 100%;

  background: linear-gradient(90deg, var(--primary-color), var(--secondary-color));

  transition: width 0.3s ease;

}

/* Buttons */

.btn {

  padding: 0.75rem 1.5rem;

  border-radius: 8px;

  border: none;

  font-weight: 600;

  cursor: pointer;

  transition: all 0.3s ease;

  font-family: 'Montserrat', sans-serif;

}

.btn-primary {

  background: var(--primary-color);

  color: white;

}

.btn-primary:hover {

  background: var(--primary-hover);

  transform: translateY(-2px);

  box-shadow: var(--shadow);

}

.btn-success {

  background: var(--success-color);

  color: white;

}

.btn-success:hover {

  background: #059669;

}

.btn-danger {

  background: var(--danger-color);
  color: white;

}

.btn-danger:hover {

  background: #dc2626;

}

.btn:disabled {

  background: var(--text-light);

  cursor: not-allowed;

  transform: none;

}

/* Profile Styles */

.profile-container {

  max-width: 800px;

  margin: 0 auto;

}

.profile-card {

  background: white;

  border-radius: 12px;

  box-shadow: var(--shadow);

  padding: 2rem;

}

.profile-header {

  display: flex;

  align-items: center;

  gap: 2rem;

  margin-bottom: 2rem;

  padding-bottom: 2rem;

  border-bottom: 1px solid var(--border-color);

}

.profile-avatar {

  width: 120px;

  height: 120px;

  border-radius: 50%;

  background: linear-gradient(135deg, var(--primary-color), var(--secondary-color));
```

```css
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 3rem;
  color: white;
  font-weight: 700;
}
.profile-info h2 {
  font-size: 1.75rem;
  color: var(--text-dark);
  margin-bottom: 0.5rem;
}
.profile-info p {
  color: var(--text-light);
  font-size: 1rem;
}
.profile-form {
  display: grid;
  gap: 1.5rem;
}
.form-group {
  display: flex;
  flex-direction: column;
}
.form-group label {
  font-weight: 600;
  color: var(--text-dark);
  margin-bottom: 0.5rem;
}
.form-group input {
  padding: 0.875rem;
  border: 2px solid var(--border-color);
  border-radius: 8px;
  font-size: 1rem;
  font-family: 'Montserrat', sans-serif;

  transition: all 0.3s ease;

}

.form-group input:focus {

  outline: none;

  border-color: var(--primary-color);

  box-shadow: 0 0 0 3px rgba(79, 70, 229, 0.1);

}

.form-group input:disabled {

  background-color: var(--light-bg);

  cursor: not-allowed;

}

.form-actions {

  display: flex;

  gap: 1rem;

  margin-top: 1rem;

}

/* Alert Notification */

.alert {

  padding: 1rem 1.5rem;

  border-radius: 8px;

  margin-bottom: 1rem;

  display: flex;

  align-items: center;

  gap: 0.75rem;

  animation: slideDown 0.3s ease;

}

@keyframes slideDown {

  from {

    opacity: 0;

    transform: translateY(-20px);

  }

  to {

    opacity: 1;
    transform: translateY(0);
  }
}
.alert.success {
  background-color: #d1fae5;
  color: #065f46;
  border-left: 4px solid var(--success-color);
}
.alert.error {
  background-color: #fee2e2;
  color: #991b1b;
  border-left: 4px solid var(--danger-color);
}
.alert.info {
  background-color: #dbeafe;
  color: #1e40af;
  border-left: 4px solid var(--primary-color);
}
/* Dark Mode Support */
body.dark-mode {
  background: linear-gradient(135deg, #1a202c 0%, #2d3748 100%);
}
body.dark-mode .stat-card,
body.dark-mode .dashboard-section,
body.dark-mode .course-card,
body.dark-mode .profile-card {
  background-color: #2d3748;
  color: white;
}
body.dark-mode .top-bar {
  background-color: #1a202c;
  border-bottom-color: #4a5568;
}
body.dark-mode .stat-card h3,
```

```css
body.dark-mode .stat-card .stat-value,

body.dark-mode .dashboard-section h2,

body.dark-mode .course-body,

body.dark-mode .profile-info h2,

body.dark-mode .form-group label,

body.dark-mode .top-bar h1 {

  color: white;

}

body.dark-mode .assignment-item {

  background-color: #4a5568;

}

body.dark-mode .form-group input {

  background-color: #4a5568;

  border-color: #4a5568;

  color: white;

}

body.dark-mode .form-group input:disabled {

  background-color: #1a202c;

}

/* Responsive Design */

@media (max-width: 768px) {

  .stats-grid {

    grid-template-columns: 1fr;

  }

  .course-grid {

    grid-template-columns: 1fr;

  }

  .profile-header {

    flex-direction: column;

    text-align: center;

  }

  .assignment-item {

    flex-direction: column;

    align-items: flex-start;
    gap: 0.5rem;
  }
  .form-actions {
    flex-direction: column;
  }
  .form-actions .btn {
    width: 100%;
  }
}
/* ========== Assignments List Styles ========== */
.assignments-list {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.assignment-card {
  background: white;
  padding: 1.5rem;
  border-radius: 12px;
  box-shadow: var(--shadow);
  display: flex;
  justify-content: space-between;
  align-items: center;
  transition: all 0.3s ease;
  border-left: 4px solid var(--primary-color);
}
.assignment-card:hover {
  box-shadow: var(--shadow-lg);
  transform: translateX(5px);
}
.assignment-card.late {
  border-left-color: var(--danger-color);
}
.assignment-card.submitted {
  border-left-color: var(--success-color);

  opacity: 0.8;

}

.assignment-details {

  flex: 1;

}

.assignment-details h3 {

  font-size: 1.125rem;

  color: var(--text-dark);

  margin-bottom: 0.5rem;

}

.assignment-details .assignment-meta {

  display: flex;

  gap: 1.5rem;

  font-size: 0.875rem;

  color: var(--text-light);

  margin-bottom: 0.5rem;

}

.assignment-details .assignment-course {

  font-weight: 600;

  color: var(--primary-color);

}

.assignment-details .assignment-due {

  display: flex;

  align-items: center;

  gap: 0.25rem;

}

.assignment-actions {

  display: flex;

  gap: 1rem;

  align-items: center;

}

.submit-btn {

  padding: 0.625rem 1.25rem;
  border-radius: 8px;

  border: none;

  font-weight: 600;

  cursor: pointer;

  transition: all 0.3s ease;

  font-family: 'Montserrat', sans-serif;

  background: var(--success-color);

  color: white;

}

.submit-btn:hover:not(:disabled) {

  background: #059669;

  transform: translateY(-2px);

  box-shadow: var(--shadow);

}

.submit-btn:disabled {

  background: var(--text-light);

  cursor: not-allowed;

  transform: none;

}

/* Alert Container */

#alertContainer {

  position: fixed;

  top: 20px;

  right: 20px;

  z-index: 9999;

  max-width: 400px;

}

@media (max-width: 768px) {

  .assignment-card {

    flex-direction: column;

    align-items: flex-start;

    gap: 1rem;

  }

  .assignment-actions {
    width: 100%;

    justify-content: space-between;

  }

  #alertContainer {

    left: 20px;

    right: 20px;

    max-width: none;

  }

}
```

4. JavaScript Files

File: js/auth.js

```
/**
 * auth.js - Login Validation Module
 *
 * Task 1: Login Validation (Client-Side)
 *
 * Requirements:
 * - Validate username and password fields
 * - Display error messages if fields are empty
 * - Allow login only if both fields are filled
 * - Redirect to dashboard page on successful validation
 *
 * JavaScript Requirements:
 * - Event handling (submit, click)
 * - DOM selection and manipulation
 * - Prevent default form submission
 *
 * Elements to interact with:
 * - #loginForm - form element
 * - #username - username input
 * - #password - password input
 * - #usernameError - error message for username
 * - #passwordError - error message for password
 * - #loginButton - submit button
 */
// Get DOM elements
const loginForm = document.getElementById('loginForm');
const username = document.getElementById('username');
const password = document.getElementById('password');
const usernameError = document.getElementById('usernameError');
const passwordError = document.getElementById('passwordError');

const loginButton = document.getElementById('loginButton');

loginForm.addEventListener('submit', (e) =&gt; {

    e.preventDefault();

    const usernameValue = username.value.trim();

    const passwordValue = password.value.trim();

    let isValid = true;

    // Validate username

    if (usernameValue === '') {

        usernameError.textContent = 'Username is required';

        usernameError.classList.add('show');

        username.style.borderColor = 'var(--danger-color)';

        isValid = false;

    } else if (usernameValue.length &lt; 3) {

        usernameError.textContent = 'Username must be at least 3 characters';

        usernameError.classList.add('show');

        username.style.borderColor = 'var(--danger-color)';

        isValid = false;

    } else if (!/^[a-zA-Z0-9_]+$/.test(usernameValue)) {

        usernameError.textContent = 'Username can only contain letters, numbers, and underscore';

        usernameError.classList.add('show');

        username.style.borderColor = 'var(--danger-color)';

        isValid = false;

    } else {

        usernameError.classList.remove('show');

        username.style.borderColor = 'var(--border-color)';

    }

    // Validate password

    if (passwordValue === '') {

        passwordError.textContent = 'Password is required';

        passwordError.classList.add('show');

        password.style.borderColor = 'var(--danger-color)';

        isValid = false;

    } else {
        passwordError.classList.remove('show');

        password.style.borderColor = 'var(--border-color)';

    }

    // If validation passes, check credentials

    if (isValid) {

        // Check if user exists and password matches

        const users = JSON.parse(localStorage.getItem('users')) || [];

        const user = users.find(u =&gt; u.username.toLowerCase() === usernameValue.toLowerCase());

        if (!user) {

            usernameError.textContent = 'Username not found. Please sign up first.';

            usernameError.classList.add('show');

            username.style.borderColor = 'var(--danger-color)';

            return;

        }

        if (user.password !== passwordValue) {

            passwordError.textContent = 'Incorrect password';

            passwordError.classList.add('show');

            password.style.borderColor = 'var(--danger-color)';

            return;

        }

        // Successful login

        localStorage.setItem('username', user.username);

        localStorage.setItem('userEmail', user.email);

        localStorage.setItem('isLoggedIn', 'true');

        showSuccessMessage();

        setTimeout(() =&gt; {

            window.location.href = 'dashboard.html';

        }, 1000);

    }

});

// Clear error when user starts typing

username.addEventListener('input', () =&gt; {

    if (username.value.trim() !== '') {

        usernameError.classList.remove('show');
        username.style.borderColor = 'var(--border-color)';

    }

});

password.addEventListener('input', () =&gt; {

    if (password.value.trim() !== '') {

        passwordError.classList.remove('show');

        password.style.borderColor = 'var(--border-color)';

    }

});

// Function to show success alert

function showSuccessMessage() {

    const alertContainer = document.getElementById('alertContainer');

    if (!alertContainer) return;

    const alert = document.createElement('div');

    alert.className = 'alert success';

    alert.innerHTML = `&lt;span&gt;Login successful! Redirecting to dashboard...&lt;/span&gt;`;

    alertContainer.appendChild(alert);

    setTimeout(() =&gt; {

        alert.remove();

    }, 1500);

}
```

File: js/dashboard.js

```
/**
 * dashboard.js - Dashboard Module
 *
 * Task 3: Dynamic Dashboard Cards
 * Task 9: Navigation Active State
 * Task 10: Alert System
 */
// Sample data structure for dashboard stats
const dashboardStats = [
    {
        title: 'Total Courses',
        value: 6,
        label: 'Enrolled Courses'
    },
    {
        title: 'Assignments Due',
        value: 3,
        label: 'Pending Submissions'
    },
    {
        title: 'Attendance',
        value: '92%',
        label: 'Overall Attendance'
    },
    {
        title: 'GPA',
        value: '3.8',
        label: 'Current Semester'
    }
];
// Sample data for assignments preview
const upcomingAssignments = [
    {
        id: 1,
        title: 'Web Development Project',
        course: 'CS301',
        dueDate: '2026-02-15',
        status: 'pending'
    },
    {
        id: 2,
        title: 'Database Design',
        course: 'CS402',
        dueDate: '2026-02-18',
        status: 'pending'
    },
    {
        id: 3,
        title: 'Algorithm Analysis',
        course: 'CS505',
        dueDate: '2026-02-20',
        status: 'submitted'
    }
];
// Initialize dashboard when page loads
document.addEventListener('DOMContentLoaded', () =&gt; {
    displayUserName();
    renderDashboardStats();
    renderUpcomingAssignments();
});
// Display logged-in user's name
function displayUserName() {
    const userName = document.getElementById('userName');
    if (userName) {
        const storedUsername = localStorage.getItem('username') || 'Student';
        userName.textContent = storedUsername;
    }

}

// Task 3: Render dashboard statistics cards

function renderDashboardStats() {

    const statsGrid = document.getElementById('statsGrid');

    if (!statsGrid) return;

    // Clear existing content

    statsGrid.innerHTML = '';

    // Loop through stats and create cards

    dashboardStats.forEach(stat =&gt; {

        const statCard = document.createElement('div');

        statCard.className = 'stat-card';

        statCard.innerHTML = `

            &lt;h3&gt;${stat.title}&lt;/h3&gt;

            &lt;div class="stat-value"&gt;${stat.value}&lt;/div&gt;

            &lt;div class="stat-label"&gt;${stat.label}&lt;/div&gt;

        `;

        statsGrid.appendChild(statCard);

    });

}

// Render upcoming assignments preview

function renderUpcomingAssignments() {

    const assignmentsPreview = document.getElementById('assignmentsPreview');

    if (!assignmentsPreview) return;

    // Clear existing content

    assignmentsPreview.innerHTML = '';

    if (upcomingAssignments.length === 0) {

        assignmentsPreview.innerHTML = '&lt;p style="color: var(--text-light);"&gt;No upcoming assig

        return;

    }

    // Loop through assignments and create preview items

    upcomingAssignments.forEach(assignment =&gt; {

        const assignmentItem = document.createElement('div');

        assignmentItem.className = 'assignment-item';
        const dueDate = new Date(assignment.dueDate);

        const today = new Date();

        const daysUntil = Math.ceil((dueDate - today) / (1000 * 60 * 60 * 24));

        let dueDateText = '';

        if (daysUntil &lt; 0) {

            dueDateText = 'Overdue';

        } else if (daysUntil === 0) {

            dueDateText = 'Due today';

        } else if (daysUntil === 1) {

            dueDateText = 'Due tomorrow';

        } else {

            dueDateText = `Due in ${daysUntil} days`;

        }

        assignmentItem.innerHTML = `

            &lt;div class="assignment-info"&gt;

                &lt;h4&gt;${assignment.title}&lt;/h4&gt;

                &lt;p&gt;${assignment.course} • ${dueDateText}&lt;/p&gt;

            &lt;/div&gt;

            &lt;span class="assignment-status ${assignment.status}"&gt;${assignment.status.toUpperCa

        `;

        assignmentsPreview.appendChild(assignmentItem);

    });

}
```

File: js/courses.js

```
/**
 * courses.js - Courses Module
 *
 * Task 4: Course List Rendering
 *
 * Requirements:
 * - Store course details in a JavaScript array
 * - Dynamically generate course cards
 * - Display course status (Active / Completed)
 *
 * JavaScript Requirements:
 * - DOM creation (createElement)
 * - Template rendering using JavaScript
 * - Conditional rendering for status labels
 *
 * Elements to interact with:
 * - #courseGrid - container for course cards
 * - #menuToggle - sidebar toggle button
 *
 * Task 9: Navigation Active State
 * - Highlight active menu item based on current page
 */
// Sample data structure for courses
const courses = [
    {
        id: 1,
        name: 'Web Development',
        code: 'CS301',
        instructor: 'Dr. Smith',
        status: 'active',
        progress: 75,
        credits: 3
    },

    {

        id: 2,

        name: 'Database Management',

        code: 'CS402',

        instructor: 'Dr. Johnson',

        status: 'active',

        progress: 60,

        credits: 4

    },

    {

        id: 3,

        name: 'Data Structures',

        code: 'CS201',

        instructor: 'Dr. Williams',

        status: 'completed',

        progress: 100,

        credits: 3

    },

    {

        id: 4,

        name: 'Operating Systems',

        code: 'CS403',

        instructor: 'Dr. Brown',

        status: 'active',

        progress: 45,

        credits: 4

    },

    {

        id: 5,

        name: 'Computer Networks',

        code: 'CS404',

        instructor: 'Dr. Davis',

        status: 'active',
        progress: 30,

        credits: 3

    },

    {

        id: 6,

        name: 'Software Engineering',

        code: 'CS505',

        instructor: 'Dr. Wilson',

        status: 'active',

        progress: 50,

        credits: 3

    }

];

// Initialize courses page

document.addEventListener('DOMContentLoaded', () =&gt; {

    renderCourses();

});

// Task 4: Render course cards dynamically

function renderCourses() {

    const courseGrid = document.getElementById('courseGrid');

    if (!courseGrid) return;

    // Clear existing content

    courseGrid.innerHTML = '';

    if (courses.length === 0) {

        courseGrid.innerHTML = '&lt;p style="color: var(--text-light); grid-column: 1/-1;"&gt;No cou

        return;

    }

    // Loop through courses and create cards

    courses.forEach(course =&gt; {

        const courseCard = createCourseCard(course);

        courseGrid.appendChild(courseCard);

    });

}

// Create individual course card
```

```
function createCourseCard(course) {

    const card = document.createElement('div');

    card.className = 'course-card';

    // Determine status class and text

    const statusClass = course.status === 'active' ? 'active' : 'completed';

    const statusText = course.status === 'active' ? 'ACTIVE' : 'COMPLETED';

    card.innerHTML = `

        &lt;div class="course-header"&gt;

            &lt;h3&gt;${course.name}&lt;/h3&gt;

            &lt;p&gt;${course.code} • ${course.instructor}&lt;/p&gt;

        &lt;/div&gt;

        &lt;div class="course-body"&gt;

            &lt;div class="course-info"&gt;

                &lt;span&gt;Credits: ${course.credits}&lt;/span&gt;

                &lt;span class="course-status ${statusClass}"&gt;${statusText}&lt;/span&gt;

            &lt;/div&gt;

            &lt;div class="course-progress"&gt;

                &lt;div class="progress-bar"&gt;

                    &lt;div class="progress-fill" style="width: ${course.progress}%"&gt;&lt;/div&gt;

                &lt;/div&gt;

                &lt;p style="font-size: 0.875rem; color: var(--text-light); margin-top: 0.5rem;"&gt;

                    Progress: ${course.progress}%

                &lt;/p&gt;

            &lt;/div&gt;

        &lt;/div&gt;

    `;

    // Add hover effect for interactivity

    card.addEventListener('click', () =&gt; {

        if (window.showAlert) {

            window.showAlert(`Viewing details for ${course.name}`, 'info');

        }

    });

    return card;

}
```

File: js/assignments.js

```
/**
 * assignments.js - Assignments Module
 *
 * Task 5: Assignment Status Management
 * - Store assignment data in JavaScript
 * - Display assignments dynamically
 * - Change status text and color based on due date (static logic)
 * - Statuses: Pending, Submitted, Late
 *
 * Task 6: Assignment Submission Simulation
 * - On clicking "Submit Assignment":
 *   - Change status from Pending to Submitted
 *   - Disable submit button
 *   - Show confirmation message
 *
 * JavaScript Requirements:
 * - Event listeners
 * - Button state management
 * - Dynamic text updates
 * - Conditional statements
 * - Dynamic class assignment
 * - Date comparison (basic)
 *
 * Elements to interact with:
 * - #assignmentsList - container for assignment cards
 * - .submit-btn - submit buttons (dynamically created)
 * - #alertContainer - for showing notifications
 *
 * Task 9: Navigation Active State
 * Task 10: Alert & Notification System
 */
// Sample data structure for assignments
```

```
const assignments = [
    {
        id: 1,
        title: 'Web Development Project',
        course: 'CS301 - Web Development',
        dueDate: '2026-02-15',
        status: 'pending',
        description: 'Create a responsive website using HTML, CSS, and JavaScript'
    },
    {
        id: 2,
        title: 'Database Design Assignment',
        course: 'CS402 - Database Management',
        dueDate: '2026-02-18',
        status: 'pending',
        description: 'Design an ER diagram for a library management system'
    },
    {
        id: 3,
        title: 'Algorithm Analysis Report',
        course: 'CS505 - Software Engineering',
        dueDate: '2026-02-12',
        status: 'late',
        description: 'Analyze time complexity of sorting algorithms'
    },
    {
        id: 4,
        title: 'Network Protocol Implementation',
        course: 'CS404 - Computer Networks',
        dueDate: '2026-02-25',
        status: 'pending',
        description: 'Implement a simple TCP client-server application'
    },
    {
        id: 5,
        title: 'Operating System Concepts',
        course: 'CS403 - Operating Systems',
        dueDate: '2026-02-08',
        status: 'submitted',
        description: 'Write a report on process scheduling algorithms'
    }
];
// Initialize assignments page
document.addEventListener('DOMContentLoaded', () =&gt; {
    renderAssignments();
});
// Task 5 & 6: Render assignments dynamically
function renderAssignments() {
    const assignmentsList = document.getElementById('assignmentsList');
    if (!assignmentsList) return;
    // Clear existing content
    assignmentsList.innerHTML = '';
    if (assignments.length === 0) {
        assignmentsList.innerHTML = '&lt;p style="color: var(--text-light);"&gt;No assignments avail
        return;
    }
    // Loop through assignments and create cards
    assignments.forEach(assignment =&gt; {
        const assignmentCard = createAssignmentCard(assignment);
        assignmentsList.appendChild(assignmentCard);
    });
}
// Create individual assignment card
function createAssignmentCard(assignment) {
    const card = document.createElement('div');
    card.className = 'assignment-card';
    // Determine status based on due date and current status
    const dueDate = new Date(assignment.dueDate);
    const today = new Date();

    today.setHours(0, 0, 0, 0);

    dueDate.setHours(0, 0, 0, 0);

    let status = assignment.status;

    let statusClass = status;

    // If not submitted and past due date, mark as late

    if (status === 'pending' && dueDate &lt; today) {

        status = 'late';

        statusClass = 'late';

        card.classList.add('late');

    } else if (status === 'submitted') {

        card.classList.add('submitted');

    }

    // Format due date

    const options = { year: 'numeric', month: 'short', day: 'numeric' };

    const formattedDate = dueDate.toLocaleDateString('en-US', options);

    // Calculate days until due

    const daysUntil = Math.ceil((dueDate - today) / (1000 * 60 * 60 * 24));

    let dueDateText = '';

    if (status === 'submitted') {

        dueDateText = 'Submitted';

    } else if (daysUntil &lt; 0) {

        dueDateText = `Overdue by ${Math.abs(daysUntil)} days`;

    } else if (daysUntil === 0) {

        dueDateText = 'Due today!';

    } else if (daysUntil === 1) {

        dueDateText = 'Due tomorrow';

    } else {

        dueDateText = `Due in ${daysUntil} days`;

    }

    card.innerHTML = `

        &lt;div class="assignment-details"&gt;

            &lt;h3&gt;${assignment.title}&lt;/h3&gt;

            &lt;div class="assignment-meta"&gt;
```

```
                &lt;span class="assignment-course"&gt;${assignment.course}&lt;/span&gt;

                &lt;span class="assignment-due"&gt;■ ${formattedDate} • ${dueDateText}&lt;/span&gt;

            &lt;/div&gt;

            &lt;p style="color: var(--text-light); font-size: 0.875rem; margin-top: 0.5rem;"&gt;${as

        &lt;/div&gt;

        &lt;div class="assignment-actions"&gt;

            &lt;span class="assignment-status ${statusClass}"&gt;${status.toUpperCase()}&lt;/span&gt

            &lt;button class="submit-btn" data-id="${assignment.id}" ${status !== 'pending' ? 'disab

                ${status === 'submitted' ? 'Submitted' : 'Submit'}

            &lt;/button&gt;

        &lt;/div&gt;

    `;

    // Task 6: Add submit button event listener

    const submitBtn = card.querySelector('.submit-btn');

    if (submitBtn && status === 'pending') {

        submitBtn.addEventListener('click', () =&gt; handleSubmit(assignment.id, card, submitBtn));

    }

    return card;

}

// Task 6: Handle assignment submission

function handleSubmit(assignmentId, card, button) {

    // Find the assignment

    const assignment = assignments.find(a =&gt; a.id === assignmentId);

    if (!assignment) return;

    // Update assignment status

    assignment.status = 'submitted';

    // Update UI

    const statusBadge = card.querySelector('.assignment-status');

    statusBadge.textContent = 'SUBMITTED';

    statusBadge.className = 'assignment-status submitted';

    // Update button

    button.textContent = 'Submitted';

    button.disabled = true;

    // Update card styling
    card.classList.remove('late');

    card.classList.add('submitted');

    // Show success message

    if (window.showAlert) {

        window.showAlert(`${assignment.title} submitted successfully!`, 'success');

    }

}
```

File: js/profile.js

```
/**
 * profile.js - Profile Management Module
 *
 * Task 7: Profile Page Edit Mode
 * Task 9: Navigation Active State
 * Task 10: Alert System
 */
// Get DOM elements
const profileForm = document.getElementById('profileForm');
const editButton = document.getElementById('editButton');
const saveButton = document.getElementById('saveButton');
const cancelButton = document.getElementById('cancelButton');
const profileName = document.getElementById('profileName');
const profileEmail = document.getElementById('profileEmail');
const profileAvatar = document.getElementById('profileAvatar');
// Form input fields
const fullNameInput = document.getElementById('fullName');
const emailInput = document.getElementById('email');
const studentIdInput = document.getElementById('studentId');
const departmentInput = document.getElementById('department');
const phoneInput = document.getElementById('phone');
const yearInput = document.getElementById('year');
// Store original values for cancel functionality
let originalValues = {};
// Initialize profile page
document.addEventListener('DOMContentLoaded', () =&gt; {
    loadUserProfile();
    initProfileButtons();
});
// Load user profile data
function loadUserProfile() {
    // Get stored username
    const username = localStorage.getItem('username') || 'Student';

    const email = localStorage.getItem('userEmail') || 'student@university.edu';

    // Get user data from users array

    const users = JSON.parse(localStorage.getItem('users')) || [];

    const currentUser = users.find(user =&gt; user.username === username);

    if (currentUser) {

        // Populate form fields with user data

        if (fullNameInput) fullNameInput.value = currentUser.fullName || currentUser.username;

        if (emailInput) emailInput.value = currentUser.email || email;

        if (studentIdInput) studentIdInput.value = currentUser.studentId || 'Not set';

        if (departmentInput) departmentInput.value = currentUser.department || 'Not set';

        if (phoneInput) phoneInput.value = currentUser.phone || '';

        if (yearInput) yearInput.value = currentUser.year || 'Not set';

    } else {

        // Fallback values

        if (fullNameInput) fullNameInput.value = username;

        if (emailInput) emailInput.value = email;

        if (studentIdInput) studentIdInput.value = 'Not set';

        if (departmentInput) departmentInput.value = 'Not set';

        if (phoneInput) phoneInput.value = '';

        if (yearInput) yearInput.value = 'Not set';

    }

    // Update profile header

    if (profileName) profileName.textContent = fullNameInput.value;

    if (profileEmail) profileEmail.textContent = emailInput.value;

    if (profileAvatar) profileAvatar.textContent = fullNameInput.value.charAt(0).toUpperCase();

    // Store original values

    storeOriginalValues();

}

// Store original form values

function storeOriginalValues() {

    originalValues = {

        fullName: fullNameInput.value,

        email: emailInput.value,
        studentId: studentIdInput.value,

        department: departmentInput.value,

        phone: phoneInput.value,

        year: yearInput.value

    };

}

// Initialize button event listeners

function initProfileButtons() {

    // Task 7: Edit button - Enable editing mode

    if (editButton) {

        editButton.addEventListener('click', () =&gt; {

            enableEditMode();

        });

    }

    // Save button - Save changes

    if (saveButton) {

        saveButton.addEventListener('click', (e) =&gt; {

            e.preventDefault();

            saveProfile();

        });

    }

    // Cancel button - Discard changes

    if (cancelButton) {

        cancelButton.addEventListener('click', () =&gt; {

            cancelEdit();

        });

    }

}

// Enable edit mode

function enableEditMode() {

    // Enable all input fields

    const inputs = [fullNameInput, emailInput, studentIdInput, departmentInput, phoneInput, yearInpu

    inputs.forEach(input =&gt; {

        if (input) input.disabled = false;
    });

    // Show/hide buttons

    if (editButton) editButton.style.display = 'none';

    if (saveButton) saveButton.style.display = 'inline-block';

    if (cancelButton) cancelButton.style.display = 'inline-block';

    // Focus on first field

    if (fullNameInput) fullNameInput.focus();

}

// Save profile changes

function saveProfile() {

    // Validate inputs

    if (!validateProfileInputs()) {

        return;

    }

    // Get current username

    const username = localStorage.getItem('username');

    // Get users array

    const users = JSON.parse(localStorage.getItem('users')) || [];

    // Find and update current user

    const userIndex = users.findIndex(user =&gt; user.username === username);

    if (userIndex !== -1) {

        // Update user object with new data

        users[userIndex].fullName = fullNameInput.value;

        users[userIndex].email = emailInput.value;

        users[userIndex].studentId = studentIdInput.value;

        users[userIndex].department = departmentInput.value;

        users[userIndex].phone = phoneInput.value;

        users[userIndex].year = yearInput.value;

        // Save updated users array

        localStorage.setItem('users', JSON.stringify(users));

        // Update session email if changed

        localStorage.setItem('userEmail', emailInput.value);

    }

    // Update profile header
    if (profileName) profileName.textContent = fullNameInput.value;

    if (profileEmail) profileEmail.textContent = emailInput.value;

    if (profileAvatar) profileAvatar.textContent = fullNameInput.value.charAt(0).toUpperCase();

    // Store new values

    storeOriginalValues();

    // Disable edit mode

    disableEditMode();

    // Show success message

    if (window.showAlert) {

        window.showAlert('Profile updated successfully!', 'success');

    }

}

// Cancel edit and restore original values

function cancelEdit() {

    // Restore original values

    fullNameInput.value = originalValues.fullName;

    emailInput.value = originalValues.email;

    studentIdInput.value = originalValues.studentId;

    departmentInput.value = originalValues.department;

    phoneInput.value = originalValues.phone;

    yearInput.value = originalValues.year;

    // Disable edit mode

    disableEditMode();

    // Show info message

    if (window.showAlert) {

        window.showAlert('Changes discarded', 'info', 2000);

    }

}

// Disable edit mode

function disableEditMode() {

    // Disable all input fields

    const inputs = [fullNameInput, emailInput, studentIdInput, departmentInput, phoneInput, yearInpu

    inputs.forEach(input =&gt; {

        if (input) input.disabled = true;
    });

    // Show/hide buttons

    if (editButton) editButton.style.display = 'inline-block';

    if (saveButton) saveButton.style.display = 'none';

    if (cancelButton) cancelButton.style.display = 'none';

}

// Validate profile inputs

function validateProfileInputs() {

    // Check full name

    if (!fullNameInput.value.trim()) {

        if (window.showAlert) {

            window.showAlert('Full name is required', 'error');

        }

        fullNameInput.focus();

        return false;

    }

    // Check email format

    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!emailPattern.test(emailInput.value.trim())) {

        if (window.showAlert) {

            window.showAlert('Please enter a valid email address', 'error');

        }

        emailInput.focus();

        return false;

    }

    // Check phone format (basic)

    if (phoneInput.value && !/^[\d\s\-+()]+$/.test(phoneInput.value)) {

        if (window.showAlert) {

            window.showAlert('Please enter a valid phone number', 'error');

        }

        phoneInput.focus();

        return false;

    }

    return true;
    }
```

File: js/signup.js

```
/**

 * signup.js - User Registration Module

 *

 * Requirements:

 * - Validate all form fields

 * - Check if username/email already exists

 * - Ensure password and confirm password match

 * - Store user credentials in localStorage

 * - Redirect to login page after successful registration

 */

// Get DOM elements

const signupForm = document.getElementById('signupForm');

const signupUsername = document.getElementById('signupUsername');

const signupEmail = document.getElementById('signupEmail');

const signupPassword = document.getElementById('signupPassword');

const confirmPassword = document.getElementById('confirmPassword');

const signupUsernameError = document.getElementById('signupUsernameError');

const signupEmailError = document.getElementById('signupEmailError');

const signupPasswordError = document.getElementById('signupPasswordError');

const confirmPasswordError = document.getElementById('confirmPasswordError');

// Handle form submission

signupForm.addEventListener('submit', (e) =&gt; {

    e.preventDefault();

    const usernameValue = signupUsername.value.trim();

    const emailValue = signupEmail.value.trim();

    const passwordValue = signupPassword.value.trim();

    const confirmPasswordValue = confirmPassword.value.trim();

    let isValid = true;

    // Validate username

    if (usernameValue === '') {

        showError(signupUsernameError, signupUsername, 'Username is required');

        isValid = false;
```

```
} else if (usernameValue.length &lt; 3) {

    showError(signupUsernameError, signupUsername, 'Username must be at least 3 characters');

    isValid = false;

} else if (!/^[a-zA-Z0-9_]+$/.test(usernameValue)) {

    showError(signupUsernameError, signupUsername, 'Username can only contain letters, numbers,

    isValid = false;

} else if (isUsernameTaken(usernameValue)) {

    showError(signupUsernameError, signupUsername, 'Username already exists');

    isValid = false;

} else {

    hideError(signupUsernameError, signupUsername);

}

// Validate email

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (emailValue === '') {

    showError(signupEmailError, signupEmail, 'Email is required');

    isValid = false;

} else if (!emailPattern.test(emailValue)) {

    showError(signupEmailError, signupEmail, 'Please enter a valid email address');

    isValid = false;

} else if (isEmailTaken(emailValue)) {

    showError(signupEmailError, signupEmail, 'Email already registered');

    isValid = false;

} else {

    hideError(signupEmailError, signupEmail);

}

// Validate password

if (passwordValue === '') {

    showError(signupPasswordError, signupPassword, 'Password is required');

    isValid = false;

} else {

    hideError(signupPasswordError, signupPassword);

}

// Validate confirm password
```

```
    if (confirmPasswordValue === '') {

        showError(confirmPasswordError, confirmPassword, 'Please confirm your password');

        isValid = false;

    } else if (confirmPasswordValue !== passwordValue) {

        showError(confirmPasswordError, confirmPassword, 'Passwords do not match');

        isValid = false;

    } else {

        hideError(confirmPasswordError, confirmPassword);

    }

    // If all validations pass, create account

    if (isValid) {

        createAccount(usernameValue, emailValue, passwordValue);

    }

});

// Clear errors on input

signupUsername.addEventListener('input', () =&gt; {

    if (signupUsername.value.trim() !== '') {

        hideError(signupUsernameError, signupUsername);

    }

});

signupEmail.addEventListener('input', () =&gt; {

    if (signupEmail.value.trim() !== '') {

        hideError(signupEmailError, signupEmail);

    }

});

signupPassword.addEventListener('input', () =&gt; {

    if (signupPassword.value.trim() !== '') {

        hideError(signupPasswordError, signupPassword);

    }

});

confirmPassword.addEventListener('input', () =&gt; {

    if (confirmPassword.value.trim() !== '') {

        hideError(confirmPasswordError, confirmPassword);

    }
```

```
});

// Helper function to show error

function showError(errorElement, inputElement, message) {

    errorElement.textContent = message;

    errorElement.classList.add('show');

    inputElement.style.borderColor = 'var(--danger-color)';

}

// Helper function to hide error

function hideError(errorElement, inputElement) {

    errorElement.classList.remove('show');

    inputElement.style.borderColor = 'var(--border-color)';

}

// Check if username already exists

function isUsernameTaken(username) {

    const users = JSON.parse(localStorage.getItem('users')) || [];

    return users.some(user =&gt; user.username.toLowerCase() === username.toLowerCase());

}

// Check if email already exists

function isEmailTaken(email) {

    const users = JSON.parse(localStorage.getItem('users')) || [];

    return users.some(user =&gt; user.email.toLowerCase() === email.toLowerCase());

}

// Create new account

function createAccount(username, email, password) {

    // Get existing users or initialize empty array

    const users = JSON.parse(localStorage.getItem('users')) || [];

    // Create new user object

    const newUser = {

        username: username,

        email: email,

        password: password, // Note: In real apps, never store plain text passwords!

        createdAt: new Date().toISOString()

    };

    // Add new user to array
    users.push(newUser);

    // Save to localStorage

    localStorage.setItem('users', JSON.stringify(users));

    // Show success message

    showSuccessAlert('Account created successfully! Redirecting to login...');

    // Redirect to login page after delay

    setTimeout(() =&gt; {

        window.location.href = 'index.html';

    }, 2000);

}

// Show success alert

function showSuccessAlert(message) {

    const alertContainer = document.getElementById('alertContainer');

    if (!alertContainer) return;

    const alert = document.createElement('div');

    alert.className = 'alert success';

    alert.innerHTML = `&lt;span&gt;✓ ${message}&lt;/span&gt;`;

    alertContainer.appendChild(alert);

    setTimeout(() =&gt; {

        alert.remove();

    }, 2500);

}
```

File: js/theme.js

```
/**
 * theme.js - Theme Switcher & Sidebar Toggle Module
 *
 * Task 8: Theme Switcher (Light / Dark Mode)
 * Task 2: Sidebar Toggle (Responsive UI)
 */
// Initialize theme on page load
document.addEventListener('DOMContentLoaded', () =&gt; {
    initTheme();
    initSidebar();
    setActiveNavigation();
});
// Task 8: Theme Switcher
function initTheme() {
    const themeToggle = document.getElementById('themeToggle');
    if (!themeToggle) return;
    // Load saved theme from localStorage
    const savedTheme = localStorage.getItem('theme') || 'light';
    if (savedTheme === 'dark') {
        document.body.classList.add('dark-mode');
        themeToggle.textContent = '■■';
    } else {
        themeToggle.textContent = '■';
    }
    // Toggle theme on button click
    themeToggle.addEventListener('click', () =&gt; {
        document.body.classList.toggle('dark-mode');
        const isDark = document.body.classList.contains('dark-mode');
        // Update button icon
        themeToggle.textContent = isDark ? '■■' : '■';
        // Save theme preference
        localStorage.setItem('theme', isDark ? 'dark' : 'light');
        // Show notification

        showAlert(isDark ? 'Dark mode enabled' : 'Light mode enabled', 'info', 1500);

    });

}

// Task 2: Sidebar Toggle

function initSidebar() {

    const menuToggle = document.getElementById('menuToggle');

    const sidebar = document.getElementById('sidebar');

    const overlay = document.getElementById('sidebarOverlay');

    if (!menuToggle || !sidebar) return;

    // Open/close sidebar on menu toggle

    menuToggle.addEventListener('click', () =&gt; {

        sidebar.classList.toggle('active');

        if (overlay) {

            overlay.classList.toggle('active');

        }

    });

    // Close sidebar when clicking overlay

    if (overlay) {

        overlay.addEventListener('click', () =&gt; {

            sidebar.classList.remove('active');

            overlay.classList.remove('active');

        });

    }

    // Close sidebar when clicking a nav link (mobile)

    const navLinks = document.querySelectorAll('.nav-link');

    navLinks.forEach(link =&gt; {

        link.addEventListener('click', () =&gt; {

            if (window.innerWidth &lt;= 768) {

                sidebar.classList.remove('active');

                if (overlay) {

                    overlay.classList.remove('active');

                }

            }
        });

    });

}

// Task 9: Set active navigation based on current page

function setActiveNavigation() {

    const navLinks = document.querySelectorAll('.nav-link');

    const currentPage = window.location.pathname.split('/').pop() || 'index.html';

    navLinks.forEach(link =&gt; {

        link.classList.remove('active');

        const href = link.getAttribute('href');

        if (href === currentPage) {

            link.classList.add('active');

        }

    });

}

// Task 10: Alert notification system

function showAlert(message, type = 'info', duration = 3000) {

    const alertContainer = document.getElementById('alertContainer');

    if (!alertContainer) return;

    const alert = document.createElement('div');

    alert.className = `alert ${type}`;

    const icon = type === 'success' ? '✓' : type === 'error' ? '✗' : '■';

    alert.innerHTML = `&lt;span&gt;${icon} ${message}&lt;/span&gt;`;

    alertContainer.appendChild(alert);

    // Auto-hide after duration

    setTimeout(() =&gt; {

        alert.style.opacity = '0';

        alert.style.transform = 'translateX(100%)';

        setTimeout(() =&gt; {

            alert.remove();

        }, 300);

    }, duration);

}

// Export for use in other modules
```

```
if (typeof window !== 'undefined') {

    window.showAlert = showAlert;

}
```

File: js/utils.js

```
/**
 * utils.js - Utility Helper Functions
 *
 * Common functions that can be used across multiple modules
 * Include this file in your HTML if you want to use these utilities
 */
/**
 * Show alert notification
 * @param {string} message - The message to display
 * @param {string} type - Alert type: 'success', 'error', 'info'
 * @param {number} duration - Duration in milliseconds (default: 3000)
 */
function showAlert(message, type = 'info', duration = 3000) {
    const alertContainer = document.getElementById('alertContainer');
    if (!alertContainer) return;
    const alert = document.createElement('div');
    alert.className = `alert ${type}`;
    alert.innerHTML = `
        &lt;span&gt;${message}&lt;/span&gt;
    `;
    alertContainer.appendChild(alert);
    // Auto-hide after duration
    setTimeout(() =&gt; {
        alert.style.opacity = '0';
        setTimeout(() =&gt; {
            alert.remove();
        }, 300);
    }, duration);
}
/**
 * Format date to readable string
 * @param {string} dateString - Date in YYYY-MM-DD format
 * @returns {string} Formatted date

 */

function formatDate(dateString) {

    const date = new Date(dateString);

    const options = { year: 'numeric', month: 'short', day: 'numeric' };

    return date.toLocaleDateString('en-US', options);

}

/**

 * Check if date is past due

 * @param {string} dueDate - Due date in YYYY-MM-DD format

 * @returns {boolean}

 */

function isPastDue(dueDate) {

    const today = new Date();

    const due = new Date(dueDate);

    today.setHours(0, 0, 0, 0);

    due.setHours(0, 0, 0, 0);

    return due &lt; today;

}

/**

 * Get days until due date

 * @param {string} dueDate - Due date in YYYY-MM-DD format

 * @returns {number} Number of days

 */

function getDaysUntilDue(dueDate) {

    const today = new Date();

    const due = new Date(dueDate);

    today.setHours(0, 0, 0, 0);

    due.setHours(0, 0, 0, 0);

    const diff = due - today;

    return Math.ceil(diff / (1000 * 60 * 60 * 24));

}

/**

 * Set active navigation link based on current page
 * @param {string} currentPage - Current page name (e.g., 'dashboard')
 */
function setActiveNav(currentPage) {
    const navLinks = document.querySelectorAll('.nav-link');
    navLinks.forEach(link =&gt; {
        link.classList.remove('active');
        if (link.dataset.page === currentPage) {
            link.classList.add('active');
        }
    });
}
/**
 * Get stored username from localStorage
 * @returns {string} Username or 'Student'
 */
function getStoredUsername() {
    return localStorage.getItem('username') || 'Student';
}
/**
 * Store username in localStorage
 * @param {string} username
 */
function storeUsername(username) {
    localStorage.setItem('username', username);
}
/**
 * Toggle sidebar for mobile view
 */
function toggleSidebar() {
    const sidebar = document.getElementById('sidebar');
    const overlay = document.getElementById('sidebarOverlay');
    if (sidebar && overlay) {
        sidebar.classList.toggle('active');
        overlay.classList.toggle('active');
    }
}
/**
 * Initialize sidebar toggle functionality
 */
function initSidebarToggle() {
    const menuToggle = document.getElementById('menuToggle');
    const overlay = document.getElementById('sidebarOverlay');
    if (menuToggle) {
        menuToggle.addEventListener('click', toggleSidebar);
    }
    if (overlay) {
        overlay.addEventListener('click', toggleSidebar);
    }
}
/**
 * Initialize theme toggle functionality
 */
function initThemeToggle() {
    const themeToggle = document.getElementById('themeToggle');
    if (!themeToggle) return;
    // Load saved theme
    const savedTheme = localStorage.getItem('theme');
    if (savedTheme === 'dark') {
        document.body.classList.add('dark-mode');
        themeToggle.textContent = '■■';
    }
    // Toggle theme on click
    themeToggle.addEventListener('click', () =&gt; {
        document.body.classList.toggle('dark-mode');
        const isDark = document.body.classList.contains('dark-mode');
        themeToggle.textContent = isDark ? '■■' : '■';
        localStorage.setItem('theme', isDark ? 'dark' : 'light');
    });
```

```
}

// Export functions for use in modules (optional - for organization)

// If using modules, uncomment these:

// export { showAlert, formatDate, isPastDue, getDaysUntilDue, setActiveNav, getStoredUsername, stor
```

5. Configuration Files

File: netlify.toml

```
[build]

  publish = "."

[[redirects]]

  from = "/*"

  to = "/index.html"

  status = 200
```