Case Study : HTML , CSS, Bootstrap and JS

Responsive Customer Engagement Portal

Time Allocation

- **Total Duration:** 90 minutes
- **HTML:** 20 minutes
- **CSS:** 20 minutes
- **Bootstrap:** 20 minutes
- **JavaScript:** 30 minutes

Business Context

A company wants to build a **responsive customer engagement portal** that works across devices and browsers. The portal should present information clearly, capture user input, support accessibility, and provide interactive behavior using modern frontend practices.

You are part of the frontend team responsible for implementing this portal using **HTML, CSS, JavaScript, and Bootstrap**.

User Story ID	User Story (What needs to be done)	Key Focus Areas
US-01	As a user, I want a well-structured HTML page so that content is readable and logically organized.	Basic structure, head/body, semantic tags

US-02	As a user, I want content grouped using inline and block elements so that layout behaves as expected.	Inline vs block elements
US-03	As a user, I want a form to collect basic user information.	Forms, input types, labels
US-04	As a user, I want a semantic webpage with accessibility support so that it is usable for all users.	Semantic markup, A11Y
US-05	As a user, I want audio and video embedded properly with controls.	HTML5 audio/video
US-06	As a user, I want styling applied using external CSS for better maintainability.	CSS basics, external styling
US-07	As a user, I want elements styled using selectors and box models so the layout looks consistent.	Selectors, box model
US-08	As a user, I want a responsive layout using flexbox and modern units.	Flexbox, rem/em/vh/vw
US-09	As a user, I want visually enhanced elements using CSS3 features.	Border-radius, modern CSS

US-10	As a user, I want a page layout using Bootstrap cards and tables.	Bootstrap components
US-11	As a user, I want a navigation bar with pagination.	Navbar, pagination
US-12	As a user, I want a validated form using Bootstrap styles.	Forms, validation
US-13	As a user, I want basic interactivity using JavaScript logic.	Variables, functions, loops
US-14	As a user, I want dynamic content updates using DOM manipulation.	DOM access, events
US-15	As a user, I want data fetched asynchronously and handled safely.	Fetch API, promises, error handling

Final Self-Evaluation Rubrics Table

Evaluation Area	**Level 1 – Needs Improvement**	**Level 2 – Meets Expectation**	**Level 3 – Exceeds Expectation**
Requirement Understanding	Misinterpreted or missed key requirements	Understood core requirement but missed minor details	Fully understood and addressed all requirements
Structure & Semantics	Poor structure; incorrect or missing semantic usage	Basic structure present with partial semantic usage	Clean, well-structured, semantic and meaningful markup
Code Correctness	Code has errors or does not execute as expected	Code executes with minor logical or styling issues	Error-free code with correct output and behavior

Styling & Layout	Layout breaks or inconsistent across sections	Layout mostly consistent with minor alignment issues	Consistent, responsive, and visually balanced layout
Responsiveness	Not responsive or breaks on screen resize	Partially responsive with some limitations	Fully responsive using modern units and layouts
Accessibility (A11Y)	Accessibility not considered	Basic accessibility applied (labels, alt text)	Strong A11Y support including semantics and readability
JavaScript Logic	Logic incomplete or incorrect	Logic works for standard scenarios	Logic handles edge cases and is optimized
DOM Interaction	DOM manipulation missing or incorrect	Basic DOM manipulation implemented	Efficient and dynamic DOM updates with events
Asynchronous Handling	Async operations fail or block execution	Async operations work with limited error handling	Robust async handling with proper error management
Bootstrap Usage	Incorrect or excessive use of Bootstrap classes	Correct usage of standard Bootstrap components	Optimal use of Bootstrap with customization
Code Reusability	Repeated or hard-coded logic/styles	Some reuse of code/styles	Modular, reusable, and maintainable code
Performance & Optimization	Page loads slowly or inefficient DOM usage	Acceptable performance with minor inefficiencies	Optimized performance and minimal reflows
Validation & Error Handling	No validation or error feedback	Basic validation implemented	Clear, user-friendly validation and error messages
Best Practices	Ignores recommended practices	Follows most best practices	Consistently follows industry best practices
Time Management	Task not completed within time	Completed with slight time overruns	Completed comfortably within allotted time

No.	Area	Instruction
1	Submission Format	Submit a single ZIP folder containing all project files
2	Folder Naming	<FullName>_<Batch>_<CaseStudyName>
3	Folder Structure	Must include index.html, css, js, assets, README, Self-Evaluation file
4	HTML File	Use HTML5 structure, semantic elements, forms, media as per user stories
5	CSS File	Use external CSS, selectors, box model, flexbox, responsive units
6	JavaScript File	Include DOM manipulation, validations, event handling, no console errors
7	Inline Code	Avoid inline CSS and JS unless explicitly required
8	README File	Include project overview, user stories covered, run instructions
9	Self-Evaluation	Submit completed self-evaluation rubrics (Excel or PDF)
10	Browser Support	Must work on latest Chrome and Edge
11	Responsiveness	Page should work on desktop, tablet, and mobile
12	Accessibility	Use labels, alt attributes, readable contrast, keyboard support
13	Coding Standards	Use meaningful names, proper indentation, no unused code
14	Prohibited Actions	No plagiarism, no external frameworks beyond syllabus
15	Time Limit	90 minutes total
16	Late Submission	Up to 15 mins late – penalty; beyond 15 mins – not accepted
17	Evaluation Basis	User story completion, code quality, responsiveness, JS logic
18	Declaration	README must include originality declaration