LESSON 86

Designing the Frontend and Adding Interactivity

240 minutes (6 periods) | Applied Learning Tasks (ALTs) | Outcome 1.15 | Outcome 1.16 | Outcome 1.20 | Outcome 1.22 | Outcome 1.23 | Outcome 2.19 | Outcome 3.3

WHAT THIS LESSON DOES

In this lesson, you'll design and build an engaging frontend for your interactive information system using HTML, CSS, and JavaScript. Focus on user-friendly UI principles, creative design, and collaborate with your team to integrate with the backend effectively.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.15	consider the quality of the user experience when interacting with computers and list the principles of universal design, including the role of a user interface and the factors that contribute to its usability
1.16	compare two different user interfaces and identify different design decisions that shape the user experience
1.20	collaborate and assign roles and responsibilities within a team to tackle a computing task
1.22	read, write, test, and modify computer programs
1.23	reflect and communicate on the design and development process
2.19	test solutions and decisions to determine their short-term and long-term outcomes
3.3	use appropriate programming languages to develop an interactive website that can display information from a database that meets a set of users' needs

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand core UI principles to create intuitive and accessible interfaces.
- Develop skills in building frontend structures using HTML and CSS for visual appeal.
- Implement interactivity with JavaScript to enhance user engagement.
- Integrate frontend designs with backend systems for dynamic functionality.
- Evaluate usability through testing to improve design based on user feedback.

BEFORE THE LESSON

- Confirm every student has a working code editor and can create and save index.html, a CSS file and js/script.js in a folder they can reach again next session.
- Check that the backend routes teammates are building are actually running, or agree fallback sample data, so the fetch step in Integrating Features has something to talk to.
- Test the school filter does not block the browser console or any local server the teams need to run and view their pages.
- Confirm teams are organised and each student knows whether they own frontend, database or backend, since the lesson assumes that split.

Designing the Frontend and Adding Interactivity

HOW THE LESSON RUNS

- Introduction
- UI Principles and Interactivity
- Building the Interface with HTML and CSS
- Adding Interactivity with JavaScript
- Integrating Features like Queries
- Usability Testing
- Wrap Up

TEACHING MOVES

- 2. UI Principles and Interactivity. Anchor simplicity, consistency and accessibility to their own project. Ask each student to name one page in their system and say what the single most important element on it is. Clutter is the usual failing here.
- 3. Building the Interface with HTML and CSS. Insist they sketch or reuse a wireframe before typing. Have them build semantic structure first, header, nav, main, footer, then style. Warn against inline styles that make later changes painful.
- 4. Adding Interactivity with JavaScript. Check the script tag is before the closing body tag and the path matches the file location, the commonest reason nothing happens. Have them open the console and confirm one event listener fires before adding more.
- 5. Integrating Features like Queries. Walk through what fetch is doing: the browser asks the server, the server answers. Show the Network tab so they see the request. A page that looks right can still be sending nothing.
- 6. Usability Testing. Pair teams to test each other. Insist testers try tasks silently while the builder watches and does not help. The confused pauses are the data. Note two fixes to make.

WHAT TO WATCH FOR

- Students think a page that looks finished is working, when the JavaScript is failing silently or fetch is returning nothing. Make opening the browser console non-negotiable. Show them errors and network requests are visible there, and that no error is not the same as correct output.
- Confusing what runs in the browser with what runs on the server, so they expect fetch to work with no backend running. Draw the two-machine picture. The frontend cannot query data that no server is serving. Confirm the backend route responds before debugging the frontend code.
- Believing usability testing means asking a friend if it looks nice. Reframe it as watching someone complete a real task without help. Give testers a specific goal and forbid the builder from guiding them.

DIFFERENTIATION

- Emerging: If the environment is fighting a student, give them a minimal working index.html with the script tag already correct so they build from a page that runs. Have them describe out loud what one button should do before writing any JavaScript for it.
- Developing: Ask them to explain what fetch returns and where the result appears on the page before they wire the next feature. Prompt them to check their styling stays consistent across two pages, not just the one they are looking at.
- Proficient: Push toward Higher Level treatment: add form validation, handle a failed fetch gracefully, or test keyboard-only navigation for accessibility. Have them document their usability findings and the changes made as evidence they can cite for the Applied Learning Task.