Zaahid De-Almeida

24301450

WST262 – SS2

25 April 2024

Tsk 2: Web Application Architecture

1. Overview of the Web Application Architecture

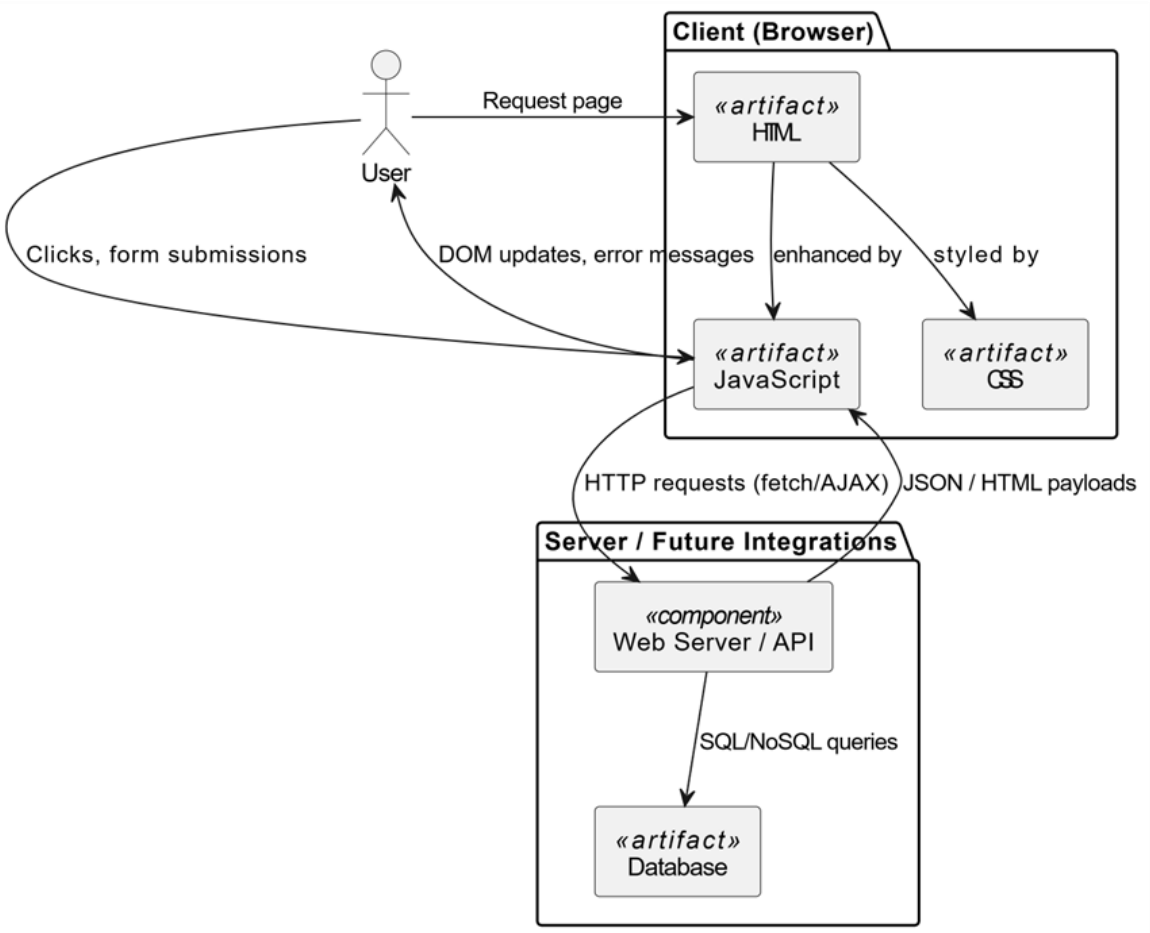
On the client side, the browser loads static assets such as HTML, CSS, and JavaScript. HTML defines the page structure with semantic tags like <header>, <nav>, <main>, <section>, and <footer>. CSS applies a professional theme and responsive layout using media queries, Flexbox or Grid, pseudo-classes, and hover effects. JavaScript handles dynamic behaviour by manipulating the DOM, processing form validation, enabling smooth scrolling, and toggling course details. These front-end technologies work together to create an accessible interface that can adapt to various screen sizes.

The server side comprises a Web Server/API component and a database, both shown as placeholders in the web architecture diagram. In a future build, the Web Server might use Node.js with Express to expose RESTful API endpoints. The database could be a relational SQL store (such as PostgreSQL) or a NoSQL document store (such as MongoDB). These server components would run on a remote host or cloud platform and manage persistent data like course listings, contact inquiries, and user sessions.

When a user interacts, such as submitting the contact form or requesting course details, JavaScript issues an HTTP request using fetch or AJAX. The Web Server receives the request, applies business logic, runs validations, and queries the database. It then returns a JSON payload to the client. JavaScript processes this response asynchronously, updating the DOM without a full page reload. Error handling and client-side validation ensure a smooth user experience and guard against invalid data.

This architecture supports modular components and semantic layering, which simplifies future enhancements such as adding authentication or analytics. Clear HTTP-based communication channels make it straightforward to integrate real-time updates or third-party APIs. Overall, the system is maintainable, scalable, and ready for a seamless backend integration.

2. Architecture Diagram



3. Scalability and Enhancements

To prepare the web application architecture for future growth and better features, the following five enhancements can be explored:

A relational database such as MySQL can be introduced to store course listings, user profiles, and inquiry submissions. This enables dynamic data management instead of hard-coding content and supports complex queries and reporting.

Incorporating a server framework (for example, Node.js with Express) allows business logic and form validation to run on the server side. Centralising these concerns enhances security, simplifies maintenance, and exposes clean RESTful APIs for the front end.

Real-time updates can be implemented using WebSockets (e.g., with Socket.io). This allows live course announcements or in-browser chat support without full page reloads, thereby improving engagement and responsiveness.

Connecting with external services such as payment gateways, analytics platforms, or an LMS API leverages proven solutions for billing, user behaviour tracking, and enrolment management. This adds functionality without rebuilding common features from scratch.

Packaging the server in Docker containers ensures predictable deployments and easier scaling. Serving static assets (HTML, CSS, JS, media) via a CDN reduces latency and automatically handles traffic spikes.