Architecture

Compare and contrast the types of front-end development you used in your full stack project, including Express HTML, JavaScript, and the single-page application (SPA).

During my full-stack project, I explored different front-end development approaches such as Express HTML rendering, JavaScript, and Single-Page Applications (SPAs). Our Travlr Gateways full-stack application started as a static HTML website and then changed to an Express HTML website, and eventually, we added an AngularJS SPA application. The last aspect of the website we created to make it workable is the addition of security. It included creation or user authentication, allowing users to log in to the website using their email and password. We used JavaScript to give the website functionality at every stage, and sometimes, we use it in conjunction with Express HTML and alone with AngularJS. Express HTML rendering entailed producing HTML templates on the server and providing the client with finished pages upon request. Although updating dynamic information frequently needed full-page refreshes, which negatively impacted user experience, it was beautiful for SEO and initial load speed. AngularJS offered solid capabilities for component design, which worked well in helping me divide my issues and turned out to be reusable. TypeScript files that link a TypeScript class to the HTML extensions are part of the AngularJS components. The components of AngularJS functioned as extension components in HTML pages, giving me the solid capability to place these components in the HTML where they are required to be.

Why did the backend use a NoSQL MongoDB database?

The back-end uses NoSQL MongoDB because MongoDB's NoSQL back-end provides flexibility, scalability, and performance advantages in handling diverse and evolving data structures typical in web applications. MongoDB database uses a flexible document-based data model, such as JSON-like documents, rather than a rigid, schema-based structure in relational databases. This flexibility allows for easier and more natural representation of complex data and evolving schemas in web applications. Also, MongoDB is designed for horizontal scalability, allowing for distributed databases across multiple servers or nodes. This scalability benefits applications expecting significant data volume growth or user traffic.

Additionally, NoSQL databases like MongoDB often provide faster read and write operations for certain use cases. Since we needed to create a MEAN stack application, I used NoSQL, which is an ideal database back-end. Another reason behind using MongoDB over SQL is that it uses JavaScript, which integrates seamlessly with the other components of the MEAN stack. It allowed us to write front-end, back-end, and database JavaScript code. Moreover, MongoDB is quick to install, simple to manage on a server, and straightforward to integrate with the other components of the MEAN stack.

Functionality

How is JSON different from Javascript and how does JSON tie together the frontend and backend development pieces?

JSON is a streamlined data exchange format to store and transfer information. Unlike JavaScript, a programming language, JSON is a language-independent format suitable for various programming languages, not just JavaScript. Its structure resembles JavaScript's object literal notation, employing key-value pairs to represent data in a readable and easily parsable format by humans and machines. Conversely, JavaScript is a programming language used to create dynamic web content, manipulate the DOM, and facilitate interactive elements on web pages. While JSON shares similarities with JavaScript objects, it's distinct in its purpose as a data format rather than a programming language. JSON is pivotal in connecting front-end and back-end systems in web development like MEAN stack development. When acting as a common language for data exchange, the server often responds by providing the requested information in JSON format when a client requests data from a server. JavaScript on the client side can effortlessly parse this JSON data, converting it into JavaScript objects for seamless integration and utilization within the front-end application.

Provide instances in the full stack process when you refactored code to improve functionality and efficiencies and name the benefits that come from reusable user interface (UI) components.

One of the instances in the Full Stack Development I just created where code refactoring occurred was when I created a card list and refactored it to card components. Suppose we are working on a web application with a form component on the frontend that sends user data to the backend. Initially, the code might not be as modular or efficient. We can refactor the code by creating a reusable function that handles the submission process based on the data received instead of having duplicate code for handling different form submissions. In the Card folder, the image and trip-specific data were in the Card component, which the Card List listed. Refactoring backend code like the card list can involve improving database queries, optimizing algorithms, or restructuring code for better scalability. This can lead to faster response times when one log in to the website or sends a request while logged in, which will be more efficient resource utilization. Creating reusable UI components in the front end can lead to cleaner code and easier maintenance. For instance, creating a reusable form component that can be easily configured for different types of data inputs.

Some of the benefits of reusable components are that they ensure that the UI elements are consistent across the application, maintaining a unified look and feel. Reusable components save time for developers when creating software because sometimes, developers reuse an existing component rather than building from scratch, speeding up development. Also, when a change is needed, updating a single reusable component affects all instances where it's used, reducing the chances of errors and making maintenance easier.

Testing

Methods for request and retrieval necessitate various types of API testing of endpoints, in addition to the difficulties of testing with added layers of security. Explain your understanding of methods, endpoints, and security in a full stack application.

In Full Stack Application like the one I just completed, when we mention method, it refers to the actions or operations that can be performed on the server. These methods are often associated with HTTP verbs like GET, POST, PUT, and DELETE. Each method serves a specific purpose. For instance, GET retrieves data, POST submits new data, PUT updates existing data, and DELETE removes data.

Now, let's discuss what endpoints mean in a full-stack application. Endpoints are the URLs that these methods interact with. They represent specific resources or functionalities on the server. For instance, /users could be an endpoint that deals with user-related operations, while /products might handle product-related tasks. When testing APIs, it's essential to evaluate the functionality of these methods by interacting with the endpoints they correspond to, which involves sending requests using tools like Postman to the API endpoints with various inputs to ensure the correct behavior and response.

Let's talk about security, the last thing we created in our MEAN stack development. The application will be vulnerable and not function as required without security because security adds another layer of complexity to API testing. We can also secure APIs using various methods such as authentication like OAuth, JWT, encryption (SSL/TLS), input validation, rate limiting, and more. Testing the security aspects involves checking if these measures are properly implemented and if they protect the API from common threats like unauthorized access, injection attacks, data breaches, and denial-of-service attempts.

Reflection

How has this course helped you in reaching your professional goals? What skills have you learned, developed, or mastered in this course to help you become a more marketable candidate in your career field?

I have learned so much in this course and have gained much experience and knowledge on how websites and other applications are developed. I used to visit different websites without knowing what contributes to their development, but this course has opened my eyes and given me an understanding, and I have begun to love it. My professional goal is to become a more versatile and marketable candidate in the tech industry.

My ability to work on both frontend and backend development could open up opportunities in various companies or even allow me to work independently on freelance projects. For instance, mastering languages like HTML, CSS, JavaScript, and Python can enhance my abilities to understand and create different web applications.

Also, my proficiency in database technologies such as MySQL, MongoDB, Visual Studio Code, and Jupyter Notebook could allow me to organize and manage data efficiently. My understanding of server-side programming in this project with technologies like Node.js, Express, or Django could enable me to create robust server systems and manage databases. I have enjoyed using Git and PowerShell in this course and becoming adept at using Git or other version control systems will help me collaborate on projects and manage code effectively.

In summary, I enjoyed the class, and I would love to learn more and become an expert in the future. Also, I recommend this course to anyone pursuing a computer science degree.