**CS-465 Module Eight Journal**

**Architecture**

**In this project, I built a full stack web application that served both customer-facing and administrative sides. On the frontend, I worked with Express HTML templates and JavaScript for the customer-facing portion and an Angular single-page application (SPA) for the admin side. Express HTML and JavaScript offered a straightforward, page-by-page rendering approach, which is easy to implement and provides simplicity for general customers browsing travel packages. In contrast, the Angular SPA provided a more dynamic, responsive experience, reducing page reloads and allowing administrators to manage data seamlessly through reusable UI components. The SPA architecture required more setup, but its efficiency and user experience advantages outweighed the initial complexity.**

**On the backend, I used a NoSQL MongoDB database. This was chosen because MongoDB stores data as documents in a flexible JSON-like structure, which aligned perfectly with the dynamic nature of the travel booking application. Unlike relational databases that require predefined schemas, MongoDB allowed for more agile development and simpler integration with JavaScript objects. This flexibility was crucial since customer and trip package data could vary in structure.**

**Functionality**

**One important piece in connecting the frontend and backend was JSON (JavaScript Object Notation). JSON is not the same as JavaScript; it is a lightweight, language-independent format designed specifically for storing and transmitting structured data. It tied together both sides of the application by acting as the “bridge” for data exchange. For example, when a customer searched for a travel package, the backend returned data in JSON format, which the frontend then parsed and displayed.**

**Throughout the development process, I refactored code multiple times to improve functionality and efficiency. One example was restructuring how trip data was retrieved and displayed in the SPA. By creating reusable UI components, I avoided repetitive code and made it easier to update functionality across multiple views. This also improved maintainability, since fixing a bug or updating a feature in one component applied consistently throughout the admin interface.**

**Testing**

**API testing was an essential part of validating that the system worked correctly. I tested endpoints for GET, POST, PUT, and DELETE requests, ensuring that each method returned the expected response. Adding authentication and security layers made testing more complex, since I needed to confirm that endpoints were properly restricted to authorized users. For instance, I tested that unauthenticated requests to admin routes were denied, while authenticated users could access and update package data. These tests reinforced my understanding of how endpoints, methods, and authentication interact in a secure full stack environment.**

**Additionally, using Postman allowed me to simulate various API calls, verify data persistence in MongoDB, and troubleshoot issues more quickly. This helped ensure the application was secure, stable, and user-friendly.**

**Reflection**

**This course has been a major milestone in helping me reach my professional goals. I developed skills in full stack architecture, API integration, MongoDB, Angular, and authentication, all of which are highly relevant to the career path I want to pursue in software engineering. Beyond just coding, I also strengthened my ability to design scalable architectures, create reusable components, and implement secure login systems. These are marketable skills that demonstrate my ability to handle real-world development challenges.**

**Building the Travlr Getaways application from the ground up gave me confidence in developing full stack projects and showed me the importance of thinking about security, functionality, and scalability from the very beginning. By including this project in my portfolio, I can show future employers that I not only understand how to build applications, but also how to secure and maintain them effectively.**