Michael Isenhour

February 9, 2025

CS 499 – Computer Science Capstone

**Enhancement 3 Narrative**

**Briefly Describe the Artifact**

The artifact I selected for this enhancement is the database architecture of my Travlr Getaways project. Originally, this project was built using the MEAN stack (MongoDB, Express.js, Angular, and Node.js), where trip data was stored in MongoDB, a NoSQL database. While MongoDB allowed for flexible data storage, it lacked relational structure, indexing optimizations, and strong data integrity controls, which made it very inefficient as the dataset grew. Additionally, the absence of foreign key constraints and relational mapping meant that data relationships had to be manually managed in the application’s code, increasing overall complexity and reducing database efficiency.

The original MEAN application also lacked user authentication and role management, as there was no way for users to register and sign up from the site directly. In this enhancement, I implemented user registration and authentication, ensuring that when users sign up, they are automatically assigned the 'user' role, preventing them from giving themselves admin privileges. Any admin changes must be manually assigned in the database, which prevents unauthorized privilege escalation.

Also, when it came to the original MEAN application, there was no search functionality, pagination, or trip deletion features. Users had to manually scroll through all trips, making it inefficient for larger datasets. There was also no way to delete trips from the database, requiring manual intervention outside of the application. To fix these issues, my Flask implementation provides search filtering, pagination, and secure trip management, allowing admins to add, edit, and delete trips directly through the interface.

For security, I also ensured that admin-only functionalities are protected. In the MEAN stack, there were no access controls preventing unauthorized users from accessing admin features. In my Flask and PostgreSQL implementation, I restricted access so that only admins can view and interact with trip management tools. Non-admin users do not see edit, create, or delete buttons, and protected pages are restricted based on user roles. This significantly improves database security and prevents unauthorized modifications.

For my capstone project, I transitioned from MongoDB to PostgreSQL, a relational database management system that supports structured queries, foreign key constraints, indexing, and stored procedures. This transition allowed me to improve data integrity, optimize query execution, and enhance security while maintaining a scalable architecture.

**Justification for Inclusion in ePortfolio**

I included this artifact in my ePortfolio because it demonstrates my ability to design and implement optimized database structures that enhance overall efficiency, security, and scalability. The original MongoDB database had no built-in relational implementation, making it difficult to maintain data consistency across different collections. By migrating to PostgreSQL, I applied relational database principles to ensure that data relationships are properly structured, and queries execute efficiently.

In addition, I optimized query performance by introducing foreign keys, indexing, and stored procedures, which significantly improve data retrieval times. These enhancements showcase my ability to design and optimize databases using industry-standard best practices. By moving to a relational model, I created a more structured, scalable, and efficient system, which directly improves query execution, data integrity, and security.

**Enhancement Plan and Implementation**

To improve database efficiency and scalability, I migrated my data from MongoDB to PostgreSQL and implemented several optimizations to improve query execution and maintainability. The first major change was that I designed a relational schema that includes proper table relationships using foreign keys, ensuring that trip data is stored efficiently. Unlike MongoDB, where relationships had to be manually maintained in the application, PostgreSQL enforces data consistency at the database level, reducing the risk of inconsistent data.

Another key enhancement was introducing indexing on frequently queried fields such as trip names, start dates, and price ranges, allowing PostgreSQL to execute searches using Index Scans (O(log n)) instead of Sequential Scans (O(n)). This significantly decreases query execution time, making the application more responsive even as the dataset grows.

Lastly, I optimized security by implementing role-based access control (RBAC) at the database level, ensuring that only authorized users can modify data. In MongoDB, access control was handled at the application layer, but with PostgreSQL, I can enforce permissions within the database, making the system more secure. In addition, I also restricted admin controls so that only authorized users can see or interact with trip management features, further protecting database integrity.

**Meeting Course Outcomes**

Through this enhancement, I successfully met the course outcomes I set out to achieve in Module One.

**Collaborative Environments**

By designing a structured relational database, I improved data organization, making it easier for future developers to understand and collaborate on the project. The implementation of foreign keys and indexing ensures maintainability and scalability, which helps to support teamwork in future database optimizations.

**Professional Communication**

I ensured that the database structure and enhancements were easily understood and communicated through clear and structured SQL queries. By writing efficient and readable queries with proper formatting and comments, I made it easier to understand and maintain the database. I also implemented role-based access control (RBAC) to ensure that database permissions and restrictions were well-defined, improving both security and clarity in managing user roles. By maintaining a well-structured and transparent approach to database security and query optimization, I demonstrated my ability to communicate technical solutions effectively and ensure maintainability for future developers.

**Computing Solutions**

By transitioning from MongoDB to PostgreSQL, I developed a structured and efficient database that enhances data integrity and performance. The introduction of indexing and role-based access control ensures that the database can handle large datasets efficiently while maintaining security.

**Innovative Techniques**

Implementing relational database techniques such as indexing, foreign key constraints, and role-based access control demonstrates my ability to use well-founded and innovative strategies to optimize database operations. These enhancements allow for faster query execution and secure data management.

**Security Mindset**

Security was a top priority for this enhancement. By implementing RBAC at the database level, I restricted user access to sensitive database operations, ensuring that only admins can modify trip data. This significantly reduces the risk of data breaches and unauthorized modifications.

**Reflection on the Enhancement Process**

While this enhancement was somewhat challenging, it was easier than the other two enhancements by far. Before starting, I had limited experience working with relational database indexing, so transitioning from MongoDB to PostgreSQL required research and hands-on learning.

One of the biggest challenges I faced was ensuring that queries were executed efficiently while maintaining data consistency. In MongoDB, data retrieval relied on denormalization, whereas in PostgreSQL, I had to ensure that normalized tables were properly indexed when the database grew to maintain fast query speeds. I overcame this by using EXPLAIN ANALYZE to ensure PostgreSQL would optimize queries as the dataset grew. Another challenge was figuring out how to optimize query performance while maintaining data integrity to ensure that access controls were properly enforced at the database level.

Overall, these enhancements significantly improved database efficiency, query performance, and security, making my application much more scalable and optimized for long-term growth. This experience gave me a deeper understanding of relational database design, indexing strategies, and the importance of query optimization.