# Shebin Sajan

**CS 499 Milestone Four: Enhancement Three – Databases**

**Artifact Description**

The artifact I chose is the Travlr Getaways full-stack web application which I commenced during my CS 465 course. It is a MEAN stack-based project that uses MongoDB, Express.js, Angular and Node.js. Users can navigate through travel packages, make bookings, and the admins can manage them through an intuitive interface.

**Why This Artifact?**

I chose this artifact for the databases section because it features an application and a structured interaction with persistent data storage using MongoDB with Mongoose which was object data modeling. The project showcases my skills on defining and manipulating complex data schemas, model relations, and performing validation, querying, and optimization on a given set of data. My efforts in structuring the schema with embedded trip details along with applying validation, timestamps, and indexing underscore design and optimization capabilities in NoSQL databases.

**Enhancements Made**

In this enhancement, I refactored the original Trip schema by embedding it within a parent schema TravlrSchema that includes a destination field and a nested array of trip details. This restructure better reflects the hierarchical relationship of the data. I implemented field-level validation using Mongoose to enforce required fields such as trip name, length, resort, perPerson, and description. Additionally, I added timestamps to track document creation and updates. To improve filtering logic, I introduced a static method filterByDestination. I also wrapped all database calls in try-catch blocks to enhance error handling and system reliability. To support testing and demonstration, I created a seed.js script to populate the database with sample data.

• Schema Refactor: Combined and embedded tripSchema inside a parent TravlrSchema with a destination field and an array of trip details to better represent data hierarchy.

• Field Validation: Added required fields (trip name, length, resort, perPerson, description, etc.) with Mongoose validation rules.

* **Static Method**: Added filterByDestination helper to enhance filter logic and reusability.
* **Error Handling**: Wrapped DB operations with try-catch blocks for reliability and security.
* **Seed Script**: Created seed.js to populate dummy data and streamline testing.

Main Changes Done

**Enhancements Summary**

The database enhancements made include restructuring the schema for modularity and clarity, implementing input validation and error-handling for robust data integrity, and adding indexing for improved query performance. The addition of a seed script and updated assets further improves the development and testing experience.

### app\_api Directory File Locations (for Database Enhancement)

1. models/travlr.js
   * **Path**: app\_api/models/travlr.js
   * **Purpose**: Defines the **TravlrSchema** and embedded **TripSchema**
   * **Enhancements Made**:
     + Schema restructuring (embedded schema)
     + Field validations
     + Timestamps
     + Indexing
2. controllers/trips.js
   * **Path**: app\_api/controllers/trips.js
   * **Purpose**: Handles **business logic** and database operations for trips
   * **Enhancements Made**:
     + Wrapped DB operations in try-catch
     + Added use of static method filterByDestination
3. models/seed.js
   * **Path**: app\_api/models/seed.js (or sometimes at project root if you created it there)
   * **Purpose**: Populates **dummy data** into the MongoDB database
   * **Enhancements Made**:
     + Injects sample trip documents into travlr collection

**Course Outcomes**

This enhancement helped me meet the following CS 499 course outcomes

1. Outcome #4: I demonstrated the ability to apply sound and innovative practices by redesigning the schema and implementing best practices such as validation, indexing, and modular separation, which delivered a scalable and efficient database solution.

2. Outcome #5: I adopted a security-first mindset by validating all critical inputs and wrapping database logic in error-handling blocks, protecting against injection risks and potential runtime failures.

3. Outcome #2: I delivered professionally written documentation explaining each enhancement and how it integrates with the larger architecture to improve the performance and maintainability of the application.

**Reflection**

I learned how MongoDB schemas can be designed to depict nested data and organized at the same time through this enhancement. One of the most difficult parts of this enhancement was eliminating the redundant schema logic and enforcing uniform integration between the controllers and the new model. This enhancement has made me more confident in applying indexing, schema design, and error management in real-life scenarios. It has also strengthened my development as a full-stack developer with a well-built database design.