**Databases Narrative**

**Name:** Anthony DePoalo

**Course:** CS-499 – Computer Science Capstone

**Instructor:** Joseph Martinez

**Date:** October 19, 2025

The artifact for this milestone is the database layer of my Travlr Getaways MEAN stack project, which I first built in CS-465 Full Stack Development. The system uses MongoDB with Mongoose to store, check, and organize user and trip data. It also manages login, error handling, and clean API responses. For this milestone, I worked on making the database connection stronger, improving how inputs are checked, adding limits where needed, and keeping data accurate under real working conditions.

I chose this artifact because it shows my ability to build and maintain reliable database systems that focus on safety, accuracy, and speed. These are important skills in professional software work, especially in large or critical systems like industrial control networks. Improving the database layer gave me hands-on experience with schema design, connection setup, and data validation that directly affects how stable and trustworthy the system is.

**Enhancements Implemented**

1. Improved Connection Handling

a. Added a retry system with exponential backoff so the app reconnects automatically if the database connection fails.

b. Added shutdown logic to close database connections safely when the system stops.

c. Used environment variables to control the database link, pool size, and timeouts for easier setup in different environments.

d. Files Updated: db.js

1. Schema Validation and Indexing

a. Updated users.js and travlr.js to include checks like unique lowercase emails and trip code patterns.

b. Added syncIndexes() at startup to keep database indexes up to date.

c. Used Mongoose settings like runValidators and strictQuery to stop bad data from being saved.

d. Files Updated: users.js, travlr.js

1. Error Handling and Response Standardization

a. Updated database controllers to turn MongoDB errors into clear JSON messages.

b. Made all error messages consistent for duplicate data, missing info, and login issues.

c. Moved error handling to a single place in routes/index.js for easier maintenance.

1. Security and Privacy Improvements

a. Simplified login error messages in passport.js to stop attackers from guessing valid usernames.

b. Logged login attempts safely for admins without showing private information.

**Skills Demonstrated**

**• Database Design:** Built clear and reliable data models with validation and indexing.

**• Error Handling:** Made sure all database errors give consistent JSON responses.

**• Environment Setup:** Used environment variables for safe and flexible database settings.

**• Performance Tuning:** Added indexing and reconnect logic to keep queries quick and stable.

**• Security Work:** Protected sensitive details, reduced data exposure, and used limited-access controls.

This milestone helped me see how validation, indexing, and handling errors affect how reliable data can be. Adding stricter checks at first caused the app to crash on startup until I fixed the order the models loaded and synced the indexes the right way. That showed me how important it is to understand how the database starts up and how each model depends on another.

Another problem was keeping the same error messages between the database and controller logic. I fixed that by creating one shared error-response format for the whole backend so the client always gets clear and consistent messages no matter where the issue comes from.

Overall, this milestone showed me how much database stability supports everything else in the app, from logging in to how smooth the user experience feels.This enhancement aligns with the following Computer Science program outcomes:

* CO3 – Design and Evaluate Computing Solutions: Improved query performance and system reliability through indexing and connection management.
* CO4 – Innovative Tools and Techniques: Applied environment configuration, retry logic, and graceful shutdown features aligned with modern deployment practices.
* CO5 – Security Mindset: Strengthened schema validation, error sanitization, and secure credential handling to mitigate common vulnerabilities.

Improving the algorithm parts of the Travlr Getaways project showed that I can design, build, and test algorithms that focus on both speed and security. I used encryption methods, front-end improvements, and data checks that follow common industry standards. These changes made the program faster, protected private data, and strengthened my ability to balance ease of use with the skills needed to build reliable and secure software for industrial control systems.