**Milestone One Narrative — Code Review: Animal Shelter Project**

**Artifact Overview**

The artifact I selected for my ePortfolio is the **Animal Shelter (Grazioso Salvare)** application, a CRUD dashboard backed by MongoDB that supports filtering, visualizing, and managing animal records to identify dogs suitable for rescue training. This project was originally developed as part of my coursework and represents a full-stack solution with user interface components, an API service layer, and a NoSQL database.

In its original form, the application allowed basic filtering by breed and age, returned results from MongoDB, and provided a simple visualization of eligible candidates. While functional, the initial design had limitations in structure, security, performance, and database optimization.

**Existing Functionality (Pre-Enhancement)**

The pre-enhancement code consisted of:

* **Monolithic structure** — UI logic, API routes, and database calls were intermixed, which made testing and modifications more difficult.
* **Basic CRUD operations** — Create, read, update, and delete functions were available, but lacked centralized validation or role-based permissions.
* **Limited search capabilities** — Queries were simple and often scanned the full dataset.
* **Minimal database optimization** — No compound indexes or schema validation were implemented.
* **Manual deployment** — No automated testing or continuous integration pipelines existed.

While the core features met the initial project requirements, the architecture left room for significant improvement in maintainability, scalability, and performance.

**Code Analysis**

During my review of the existing code, I identified several areas for improvement:

1. **Architecture and Maintainability**
   * Tight coupling between layers made it hard to introduce changes without risking regressions.
   * Lack of separation of concerns reduced code clarity.
2. **Security and Data Integrity**
   * No centralized input validation; user-provided data was passed directly to queries.
   * No role-based access control to limit CRUD operations based on user privileges.
3. **Performance**
   * Queries were not using indexes; frequent operations performed full collection scans.
   * Repeated queries recalculated the same results instead of caching them.
4. **Testing and Deployment**
   * No automated tests to catch regressions.
   * No CI/CD pipeline to ensure code quality before deployment.

**Planned Enhancements**

Based on the code analysis, my enhancement plan addresses all three program categories: **Software Design & Engineering**, **Algorithms & Data Structures**, and **Databases**.

1. **Software Design & Engineering**
   * Refactor the application into a layered architecture (UI, API, and Data Access layers).
   * Implement **role-based access control (RBAC)** and centralized input validation.
   * Add **unit and integration tests** for both API and data operations.
   * Set up a **CI/CD pipeline** with automated tests and linting, and containerize the application with Docker.
2. **Algorithms & Data Structures**
   * Develop an **adoptability scoring algorithm** using weighted attributes (age, temperament, health status, training history).
   * Implement a **kennel assignment heuristic** using a priority queue to allocate dogs to limited-capacity kennels based on score and compatibility.
   * Introduce an **LRU cache** for frequently requested query results to reduce redundant database calls.
3. **Databases**
   * Add **JSON Schema validation** to enforce document structure and prevent malformed data.
   * Create **compound indexes** for common queries, a **text index** for notes, and a **2dsphere index** for geospatial searches.
   * Develop aggregation pipelines for analytics and insights (e.g., candidate counts by breed and location).
   * Use explain() plans to verify query optimization and document performance improvements.

**Anticipated Outcomes**

By implementing these enhancements:

* **Maintainability** will improve through a cleaner architecture and well-structured code.
* **Security** will be strengthened with validation and access control.
* **Performance** will increase due to indexed queries, caching, and optimized algorithms.
* **Reliability** will be ensured through automated testing and CI/CD integration.
* **Analytical capability** will expand through aggregation pipelines and database enhancements.

**Reflection**

This milestone marks the starting point for transforming a functional but basic CRUD application into a robust, scalable, and production-ready system. The code review not only clarified what was already working but also highlighted specific changes that will showcase my growth in the three core competency areas of software design, algorithms, and databases. Completing this plan will directly align my work with the program outcomes and provide tangible, measurable improvements in the artifact.