Nik Myers
CS-499-19736
November 17, 2025
Phillips

3-2 Milestone: Software Design/Engineering Enhancement Narrative

The artifact I selected for this enhancement (as well as the other two) is the Python-based Grazioso Salvare Animal Rescue Dashboard that I originally developed in CS-340. That version of the application used Dash and MongoDB to display filtered search results of animal records. While it successfully demonstrated data visualization and CRUD functionality, its design was largely monolithic. All data access, filtering, and rendering occurred inside a single bespoke Python module, which limited maintainability and scalability. For this first capstone enhancement, I focused on refactoring and modernizing the system architecture to demonstrate stronger software design and engineering practices.

The primary enhancement involved porting the application to a modern full-stack TypeScript environment using the Bun runtime, SvelteKit, and SQLite (via Drizzle ORM). This transition allowed the system to follow clear separation of concerns and take advantage of static typing, improved performance, and a well-defined component hierarchy. The new architecture is organized into distinct layers (routes, services, and repositories) which collectively follow the Model-View-Controller (MVC) paradigm. Routes handle API requests, services manage business logic and input validation, and repositories encapsulate all database access through Drizzle queries. This structure makes the codebase modular, easier to test, and much more maintainable.

From a software engineering perspective, I implemented several techniques that demonstrate professional design principles. First, I introduced strong typing and runtime validation using Zod schemas, ensuring all data passed between layers adheres to defined contracts. Second, I added a “repository” layer that isolates SQL logic and manages query

composition, including pagination, filtering, and geospatial range lookups, though the latter isn't yet implemented directly in the client. The database schema itself, stored in Drizzle's type-safe format, provides compile-time guarantees that queries match column definitions, reducing the risk of runtime SQL errors. Additionally, the service layer enforces standardized pagination and filtering logic, returning predictable, validated JSON responses to the SvelteKit frontend.

To improve performance and consistency, I added explicit ordering by the recNum field (the primary key of the AAC dataset) ensuring that results load deterministically rather than appearing in random order. I also established composite indexes on frequently filtered columns such as animal_type, breed, and outcome_type, along with latitude and longitude columns to support basic geographic queries. These optimizations allow the API to respond quickly even when searching large datasets.

On the frontend, I re-implemented the original dashboard's functionality using SvelteKit components that consume the REST-style API routes. The Svelte layer renders paginated search results dynamically while maintaining a lightweight, reactive user interface. Because the frontend and backend are now decoupled, future iterations could easily incorporate additional clients, such as a mobile app, without altering the data layer. As of this milestone, there is still much work to be done before the user interface is finished, and the app contains only barebones components to verify functionality with basic styling.

For quality assurance, I used Vitest to introduce automated unit tests covering the service and repository layers. These tests validate filtering, pagination, and location-based search logic using an in-memory SQLite database, enabling fast, deterministic test runs. This testing strategy reflects sound software engineering practice by ensuring that critical logic can be verified

without relying on manual UI testing. The caveat as of this assignment, however, is that not all of the functionality is currently “testable,” but the ultimate goal is complete coverage.

Overall, this enhancement transformed the project from a single-file prototype into a professionally structured full-stack web application. It now demonstrates key software design and engineering competencies, including modular architecture, typed data modeling, robust input validation, maintainable database access patterns, and automated testing. The resulting system is more reliable, extensible, and production-ready, effectively illustrating my growth as a software engineer and my ability to apply modern development frameworks and methodologies to a real-world application. Future enhancements will see the filtering controls, data visualization, and dynamic map implemented alongside improvements on the user interface.