Nik Myers
CS-499-19736
November 30, 2025
Phillips

5-2 Milestone: Databases Enhancement Narrative

The artifact I selected for this enhancement is my Rescue-Mate web application, originally developed as a Python-based interactive dashboard for the Austin Animal Center dataset. Over the past several weeks, I have re-implemented and modernized it into a SvelteKit-based full-stack web application that uses TypeScript, Drizzle ORM, and SQLite for data persistence and interaction. This current milestone represents the continued evolution of that system, focusing on major database-driven enhancements that improved data filtering, query efficiency, and overall interactivity. This week also included adding thorough documentation and removing most of the “dead code” leftover from earlier versions, as well as substantial UI improvements.

This artifact demonstrates my ability to design and implement robust database solutions that directly drive application functionality and user experience. It belongs in my ePortfolio because it exemplifies my applied knowledge of data modeling, query optimization, and front-end integration with asynchronous database operations.

In this milestone, I expanded upon my existing database integration by adding filtering, pagination, and visualization features that rely on real-time SQL queries through Drizzle ORM. The “Rescue Type” filter allows users to refine database queries by animal type, dynamically updating the URL parameters and fetching only the relevant results from the dataset. I also implemented pagination that properly syncs with the table state, ensuring seamless navigation through the database results without unnecessary re-queries.

Another key advancement was the creation of the breed distribution chart (the DataViz.svelte component), which uses a remote function to query aggregate breed counts using SQL's COUNT() function. This design choice reduced processing time by shifting the aggregation logic from the client to the database, demonstrating an understanding of efficient data handling principles. Additionally, I introduced a MapLibre GL-powered map component that integrates with a reactive context class (MapContext) I created to enable data-driven map updates whenever a user selects an animal record, linking front-end interactivity directly to stored geolocation data.

Each of these enhancements reinforces the application's reliance on database operations to deliver dynamic, data-rich visualizations and responsive interactivity—hallmarks of professional, scalable software design. These improvements align strongly with the CS program outcomes related to software engineering, database design, and problem solving using computing principles, with my work this week specifically demonstrating my ability to implement innovative techniques to manage and query relational data effectively, program solutions for data manipulation, filtering, and aggregation that maintain accuracy and efficiency, and integrate secure, reactive communication between the database and user interface layers of a full-stack application.

Reflecting on my process this week, this milestone pushed me to refine both my understanding of Drizzle ORM's querying capabilities and how to structure reactive front-end components that depend on asynchronous database operations. A key learning moment was realizing that querying the database directly for aggregation was far more efficient than reshaping the client-side data once it had been consumed by the table component. This realization deepened my understanding of data processing distribution between front and back ends. I also gained valuable experience integrating multiple state-driven Svelte components with a single data source, a point of contention among developers using Svelte 5's new runes paradigm. The introduction of the MapContext class allowed for real-time map updates, a more elegant and scalable solution than manual event dispatching or trying to observe the table component's data from a different component. The challenges of managing state synchronization and ensuring data consistency across components helped strengthen my architectural decision-making skills.

From a design perspective, this milestone represented the most cohesive integration of the system's database, logic, and interface layers to date. The result is a nearly finished, data-driven application that accurately represents my growth as a software engineer capable of designing end-to-end systems.