Ty Wheeler

5-2 Milestone Four Narrative

CS 499

6/4/25

*Artifact description*

The artifact I selected for this enhancement was my Grazioso Salvare Dashboard from CS 340. This project consisted of a data analysis dashboard built with Dash, powered by a MongoDB backend which was designed to help a rescue organization filter animal records based on rescue type and criteria. The initial version used PyMongo to read directly from a NoSQL database and displayed that data in a dashboard. I met the course outcomes I planned to demonstrate with this enhancement as this project showcases my ability to evaluate and redesign a database system for reliability, security and scalability.

*Justification for inclusion in ePortfolio*

I selected this artifact because it demonstrates my abilities in full stack software development across data modeling and backend APIs. The original project already highlighted my understanding of dashboard frameworks and database querying, but the enhancements made this a more secure and finished system. I enhanced the original project by migrating from MongoDB to PostgreSQL, developing a secure Flask REST API with role based access control as well as refactoring the Dash frontend to retrieve data through authenticated API requests. These changes improved the system’s scalability, security, and modularity. Migrating the backend from MongoDB to PostgreSQL required restructuring the data into a normalized schema with dogs, rescue types, and locations tables. This improvement not only made for more consistency through SQL constraints but also enabled for more advanced relational queries using SQL joins. I also developed a secure Flask REST API with routes protected using flask login. The dogs endpoint is accessible only to authenticated users and admin users are granted permission to delete records through the DELETE /dogs/<id> route. This was implemented and then tested in Postman to verify successful implementation These features demonstrate my backend development skills especially related to securing and designing role based APIs. Lastly I refactored the Dash dashboard to decouple the frontend from direct database access. I replaced all shelter.read() database calls with HTTP requests to the Flask API. This change makes the system more modular and maintainable allowing the backend to change independently of the frontend.

*Reflection on the enhancement process*

Throughout this entire process I gained valuable experience in secure API design, client-server interaction, and relational database modeling. A challenge I faced was refactoring the original MongoDB structure into a PostgreSQL schema. This required careful planning of table relationships. I had to ensure that foreign keys for rescue types and locations were properly generated and linked. Another challenge came during the transition from direct database access to the Flask API layer. I initially had a 405 Method Not Allowed error when testing routes in Postman, which I resolved by correcting the HTTP method and verifying the route decorators. Implementing role based authentication introduced complexity as well, especially with making sure that Flask Login sessions were preserved between requests. I fixed this by using requests.Session() in the Dash application to simulate a persistent login and include authentication cookies when retrieving data. Decoupling the Dash dashboard also led to a JSON decoding error due to unauthorized access attempts. This occurred because the dashboard attempted to call the /dogs API route without a valid session which resulted in an HTML error response instead of JSON. I solved this by ensuring the dashboard logs in using API credentials before making a request. Lastly I encountered issues with missing or malformed fields in the dataset, such as null coordinates or blank names, which initially broke the map and graph rendering. To address this I added defensive programming techniques to clean and filter the data before visualization, ensuring the dashboard could handle various edge cases without crashing. Overall, this artifact now serves as a comprehensive demonstration of my ability to design, troubleshoot, and secure a full stack software system with an integrated database which it visualizes in real time with intuitive controls.