The artifact I selected is a Dash-based data dashboard project that connects to a MongoDB database to visualize animal rescue data for the fictional company Grazioso Salvare. I originally created it during CS-340, where we built a full-stack dashboard using Python, Dash, and MongoDB. The project loads animal records from the database and allows users to filter by rescue type, explore breed distribution with a pie chart, and view animal locations on an interactive map.

I chose this artifact for my ePortfolio because it gave me the opportunity to demonstrate real-world use of data structures and algorithmic thinking in a web-based application. In particular, the enhancements I implemented for Category 2 focused on optimizing the way data is filtered, cached, and processed. I replaced repeated MongoDB queries with a more efficient in-memory filtering system using pandas masks, which is better suited for small to medium-sized datasets and showcases algorithmic efficiency. I also introduced caching to avoid redundant computations when users select the same filters multiple times, and used projection to reduce the amount of data transferred from the database. These changes improved the overall responsiveness of the dashboard and made the code more scalable and maintainable.

Yes, I met the course outcomes I planned for in Module One, specifically the ability to analyze and apply algorithms and data structures to solve computing problems. I don’t have any updates to my outcome-coverage plan because this enhancement aligned closely with my original goals for the Algorithms and Data Structures category.

While enhancing this artifact, I learned a lot about performance trade-offs when handling data in a web application. Switching from server-side filtering to in-memory vectorized filtering with pandas not only made the app feel faster, but it also gave me more control over how the data was processed. One challenge I ran into was handling encoding issues and inconsistencies in the data types returned from MongoDB, especially with fields like ObjectId and strings with extra whitespace. I resolved these by adding a cleanup function that standardizes and coerces the data before it's used in the dashboard.