Austin Sarkis

CS 340

4/11/2025

**README**

**Project Functionality and Proof of Work**

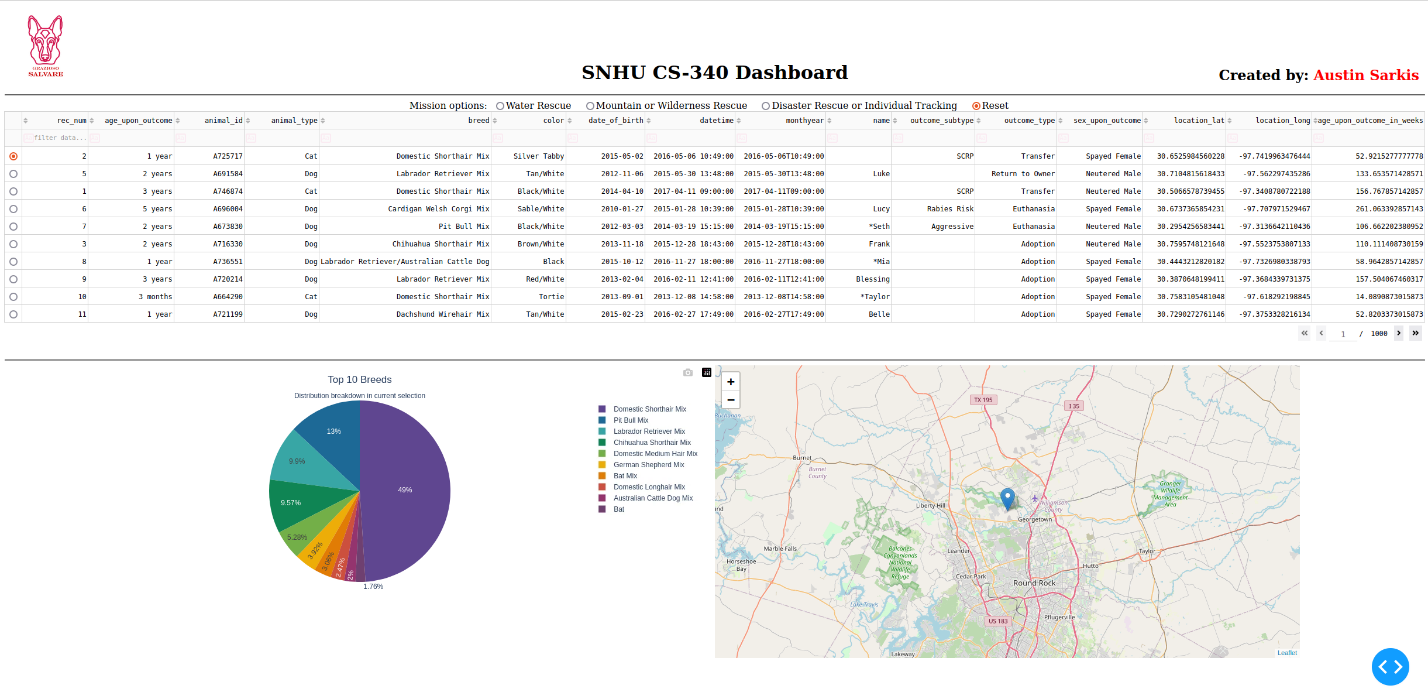
This project involved developing a web-based dashboard application for Grazioso Salvare by Global Rain. The primary purpose of the dashboard is to enable users to easily filter and visualize dog profiles from the Austin Animal Center Outcomes dataset, helping to identify dogs suitable for search-and-rescue training.

The dashboard is an interactive, client-facing web application that connects to a MongoDB database and updates dynamically based on user input. Key components of the dashboard include:

* Interactive Radio Button Filters: These allow users to select different rescue scenarios: Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue, or Individual Tracking. There is also a "Reset" option.
* Data Table: This table responds to the selected filters, displaying relevant information about the dogs.
* Dynamic Geolocation Map: This map visualizes the locations of the selected animals.
* Secondary Pie Chart: This chart shows the distribution of dog breeds among the selected records.

Additionally, the application incorporates Grazioso Salvare's branding by featuring the company logo, which is clickable and links to www.snhu.edu. It also includes a unique identifier that credits Austin Sarkis as the creator.

Screenshots of the dashboard are included to demonstrate proof of the required functionality. They show the starting state, as well as the dashboard after applying each rescue-type filter and resetting the data.

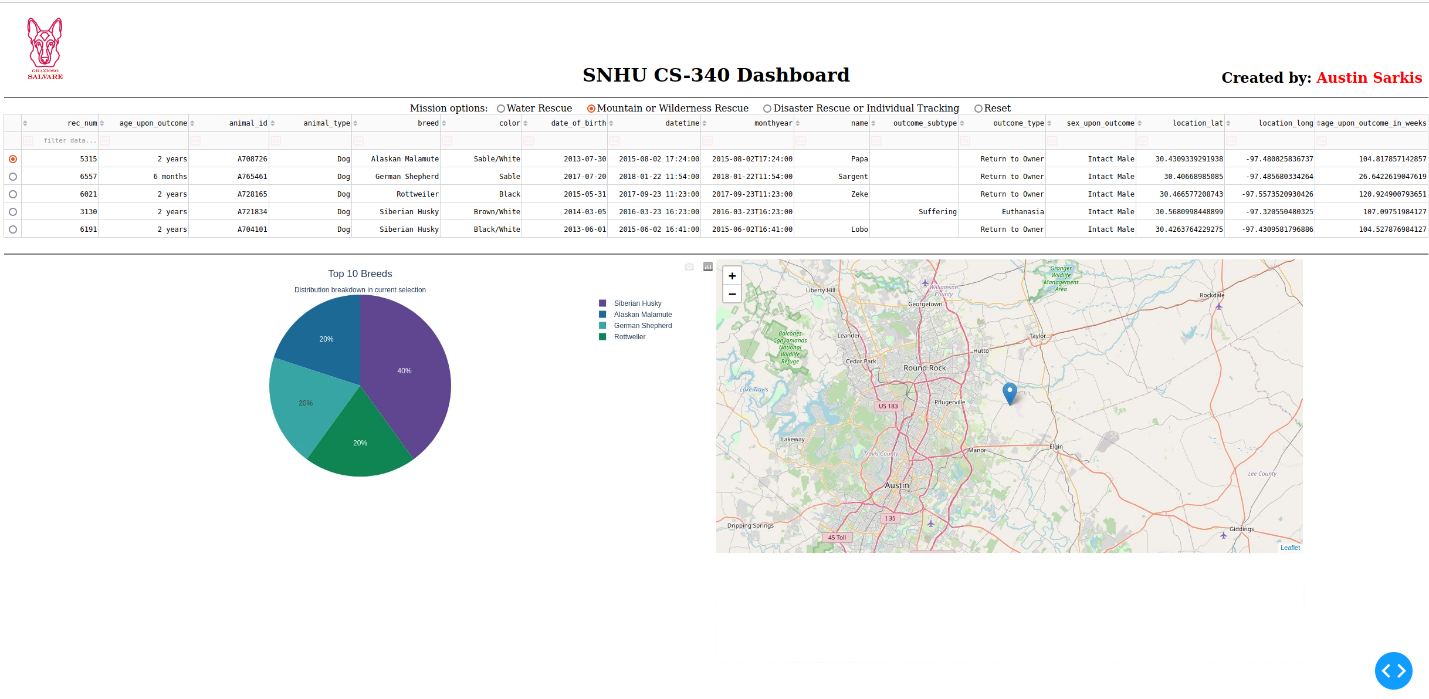


This screenshot shows the starting state of the dashboard.

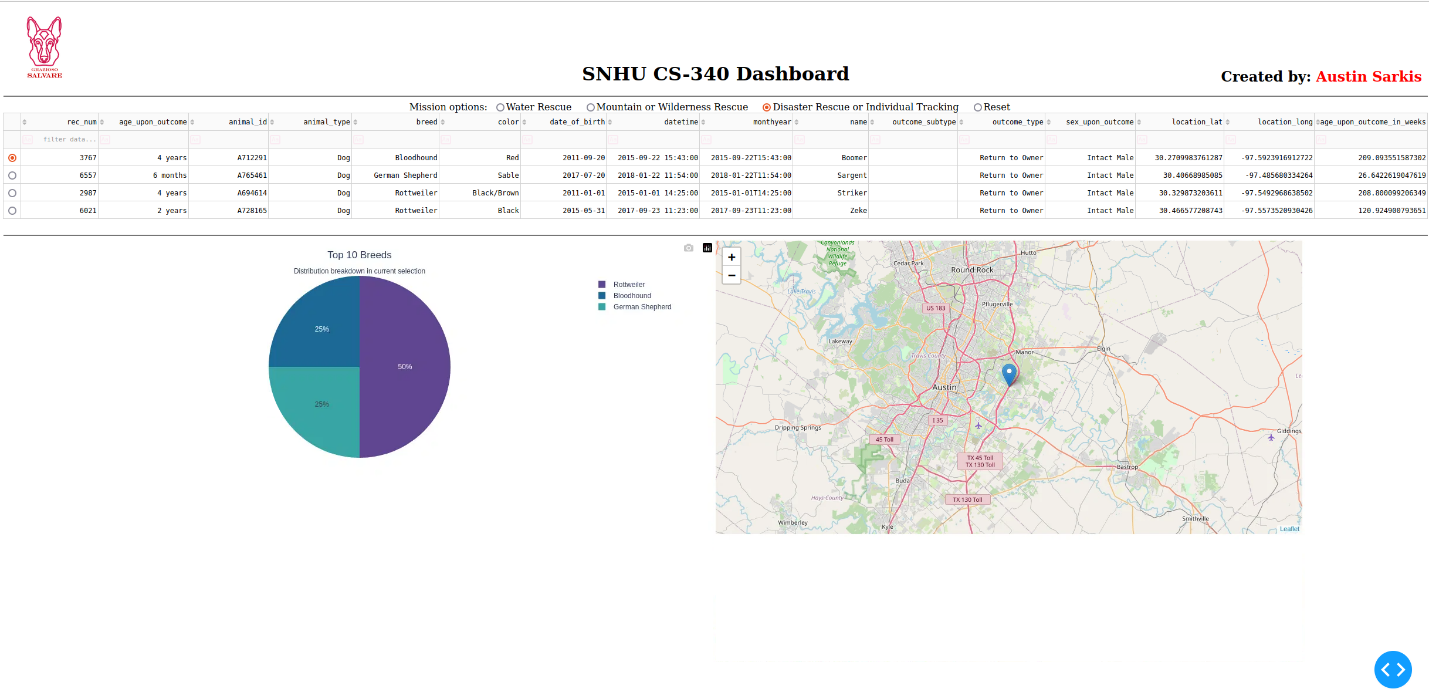
A screenshot of a map

AI-generated content may be incorrect.

This screenshot shows the Water Rescue filter being applied.



This screenshot shows the Mountain or Wilderness Rescue filter being applied.

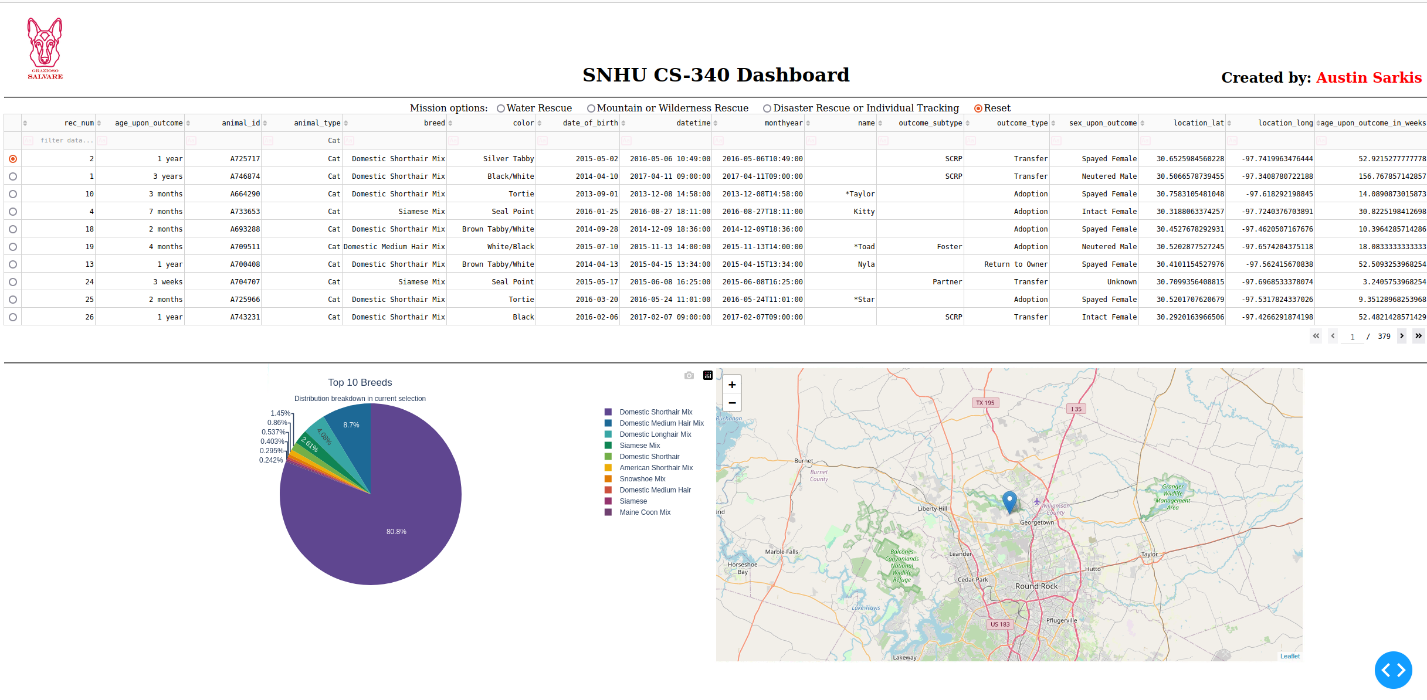


This screenshot shows the Disaster Rescue or Individual Tracking filter being applied

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot shows the dashboard returning back to its original state once Reset is selected.



This screenshot shows that other filters can be applied to the table directly instead of the radio button options. The Map and Pie Chart elements will update accordingly as well.

**Tools Used and Rationale**

The dashboard was developed using a variety of tools. Python was chosen as the primary programming language due to its flexibility, simplicity, and robust ecosystem of libraries for web development and data visualization. MongoDB was selected as the database because it integrates seamlessly with Python through the PyMongo library and provides strong support for document-based data storage. This makes it highly suitable for managing structured yet non-relational shelter outcome data.

To create the web application interface, we used the Dash framework. Dash was selected because it offers a powerful and user-friendly platform for developing interactive, data-driven dashboards in Python without requiring extensive knowledge of front-end development technologies such as JavaScript or HTML/CSS.

Additionally, we utilized libraries like Plotly Express for chart generation, Dash Leaflet for rendering geolocation maps, and Pandas for efficient data manipulation and querying.

**Why MongoDB Was Used for the Model Component**

MongoDB is an excellent choice for the model component of the MVC (Model-View-Controller) design pattern due to its flexible document structure and seamless integration with Python. It stores data in JSON-like documents, which map easily to Python dictionaries, facilitating straightforward queries and updates. The database's capability to quickly filter and retrieve documents based on fields such as animal\_type, breed, sex\_upon\_outcome, and age\_upon\_outcome\_in\_weeks makes it ideal for implementing dynamic filtering features in the dashboard. Moreover, its scalability and schema-less design provide additional benefits by supporting potential future expansions of the project without the need for significant changes to the database schema.

**How Dash Provides the View and Controller Structure**

The Dash framework provided both view and controller functionalities for the project. In Dash, layouts (views) are created using Python components such as HTML.Div, dcc.RadioItems, and dash\_table.DataTable. These components were utilized to build the dashboard layout, which includes elements like the logo, title, filtering options, data table, and charts. Dash callbacks function as controllers, managing user interactions with the radio buttons and updating the data table, map, and pie chart accordingly. The built-in interactivity and automatic updating features in Dash greatly simplified the development process. Additionally, Dash's flexibility allowed the project to satisfy all usability requirements, ensuring a responsive and intuitive user experience.

**Resources and Software Applications Used**

Throughout the project, several resources and software applications were utilized. MongoDB was employed to host the database, facilitating easy access via the CRUD Python module. The PyMongo library established the connection between Python and MongoDB. We used Dash along with its Core Components, HTML Components, DataTable, and Leaflet to build the interactive dashboard. Plotly Express was utilized to create the pie chart, while Pandas was used for data loading and filtering. Additionally, we referenced supporting documentation from the official websites of Dash and MongoDB as needed to guide our development process.

* <https://dash.plotly.com/dash-core-components>
* <https://dash.plotly.com/datatable>
* <https://www.mongodb.com/docs/manual/>

**Steps Taken to Complete the Project**

The project was completed by following a systematic development process. First, we created and populated a MongoDB database using data from the Austin Animal Center Outcomes dataset. Next, we developed a Python CRUD module to provide access to the database.

Once the backend was finished, we focused on developing the front-end dashboard. This involved creating a data table, setting up interactive filtering options, constructing a pie chart and a geolocation map, and styling the dashboard to align with Grazioso Salvare's branding requirements.

We connected the interactive radio button filters to the database queries using Dash callbacks and thoroughly tested the dynamic updates of the table and charts. Finally, we tested the dashboard by applying each filter option—Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue, Individual Tracking, and Reset—to ensure full functionality.

**Challenges Encountered and How They Were Overcome**

During the project, we encountered several challenges. One major issue was ensuring that the MongoDB queries accurately implemented multiple filtering conditions, such as matching specific breeds, sex, and training ages, while selecting only dogs. We addressed this by carefully constructing the MongoDB queries using the $in operator and range conditions like $gte (greater than or equal to) and $lte (less than or equal to).

Another challenge involved handling the Dash layout to ensure that the header, which includes the logo, title, and creator name, looked professional and maintained proper alignment across different screen sizes. We resolved this by utilizing a flexbox layout with appropriate alignment settings in Dash's HTML components.

Additionally, fine-tuning the pie chart to display only the top 10 breeds with a consistent color scheme required us to explore various built-in color palettes from Plotly. After testing several options, we chose the Prism color palette, which provided a colorful yet uniform appearance across the pie chart slices.