**Grazioso Salvare Dashboard**

Alfred Morgan

10/19/2025

SNHU CS-340 – Client/Server Development

The Grazioso Salvare Dashboard project is a full-stack Python application for loading, visualizing, and managing animal data from the Austin Animal Center. The stated purpose of the app is to “allow Grazioso Salvare to quickly identify animals that match criteria for various types of rescue missions: water, mountain, and disaster rescues or individuals in need of tracking.” The project connects a MongoDB database to a browser-based web dashboard created using Dash, allowing the organization to filter, analyze, and map animal records by various fields directly from the user interface.

The Python application implements full CRUD functionality by exposing a class called AnimalShelter that interfaces to MongoDB using pymongo. The “read” CRUD method is the only method utilized in the application to dynamically pull back records based on user input. When the data loads, it is displayed in a searchable, sortable table. The radio button filters are used by the user to select a mission type and the table, pie chart, and map all update automatically to show only the animals that match the filter’s criteria. The filter options are Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and Reset (All).

When a record is selected in the table, the map centers on that animal’s coordinates and displays a marker with the name and breed of the selected animal. The pie chart is updated in real time to show the breed distribution of all animals currently in the filter selection. Loading spinners were added to the table and map while data is loading for a smoother, more professional user experience. Table virtualization was also added to improve scrolling performance with large datasets. The header of the dashboard shows the Grazioso Salvare logo, which is linked to the SNHU homepage, and a unique tag that reads “Alfred Morgan • 10/19/2025” to verify authorship and date of project completion.

The system passes all functional and integration tests and was verified. The dashboard loads and connects to MongoDB successfully. All 20,002 records are loaded from the animal collection, and each rescue filter properly returns the correct subset of animals. Clicking “Reset (All)” successfully reloads the full dataset after a short pause due to the size of the data. The pie chart clearly shows the top eight breeds and combines smaller groups into an “Other” category to keep the visualization uncluttered. The map updates correctly to display only the selected animal.

Figure 1: Dashboard initial view showing all records loaded and Grazioso Salvare logo displayed.

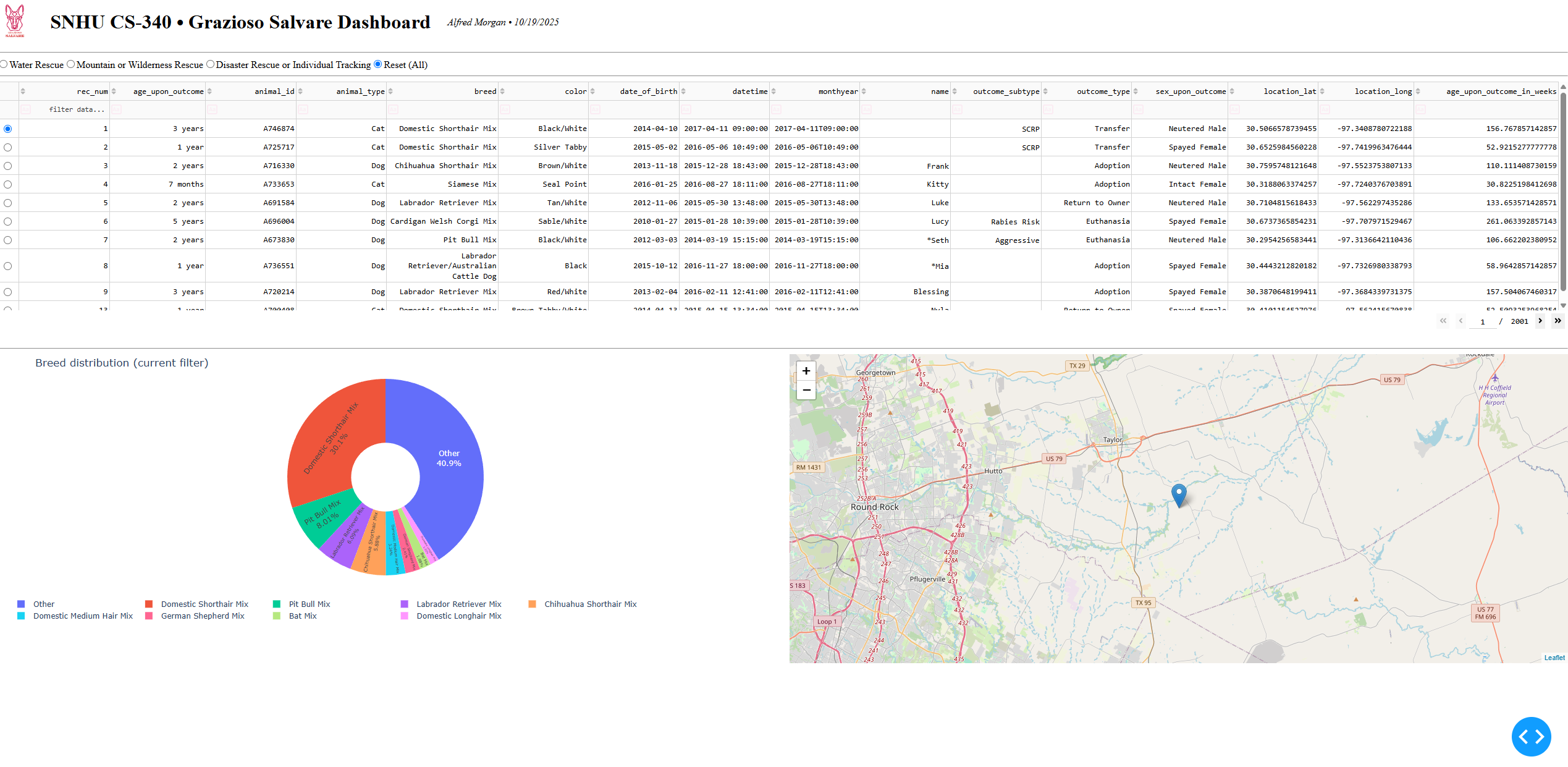


Figure 2: Dashboard filtered for “Water Rescue” dogs with pie chart and map updating dynamically.

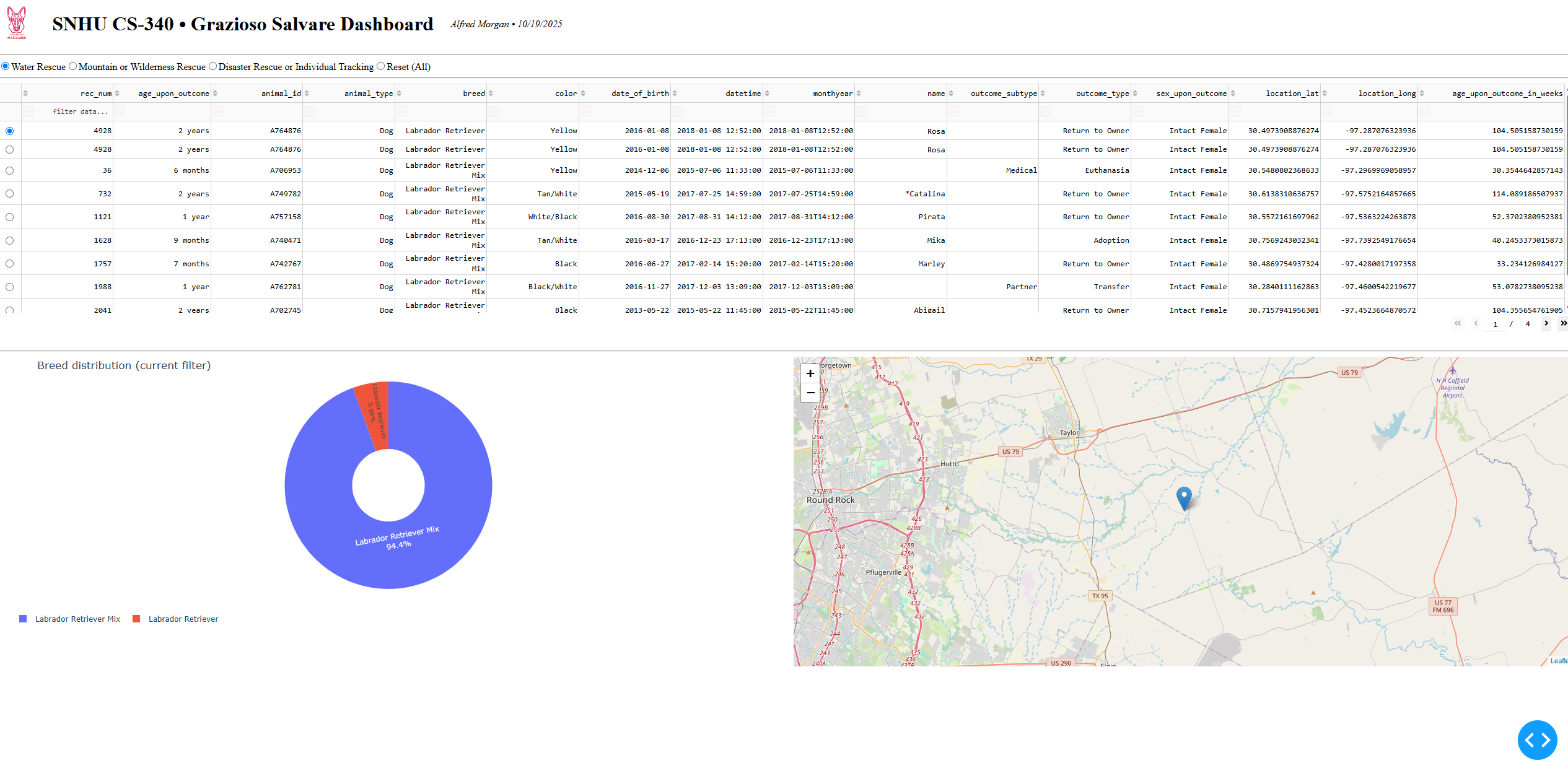


Figure 3: Dashboard displaying “Mountain or Wilderness Rescue” filter with updated breed distribution.

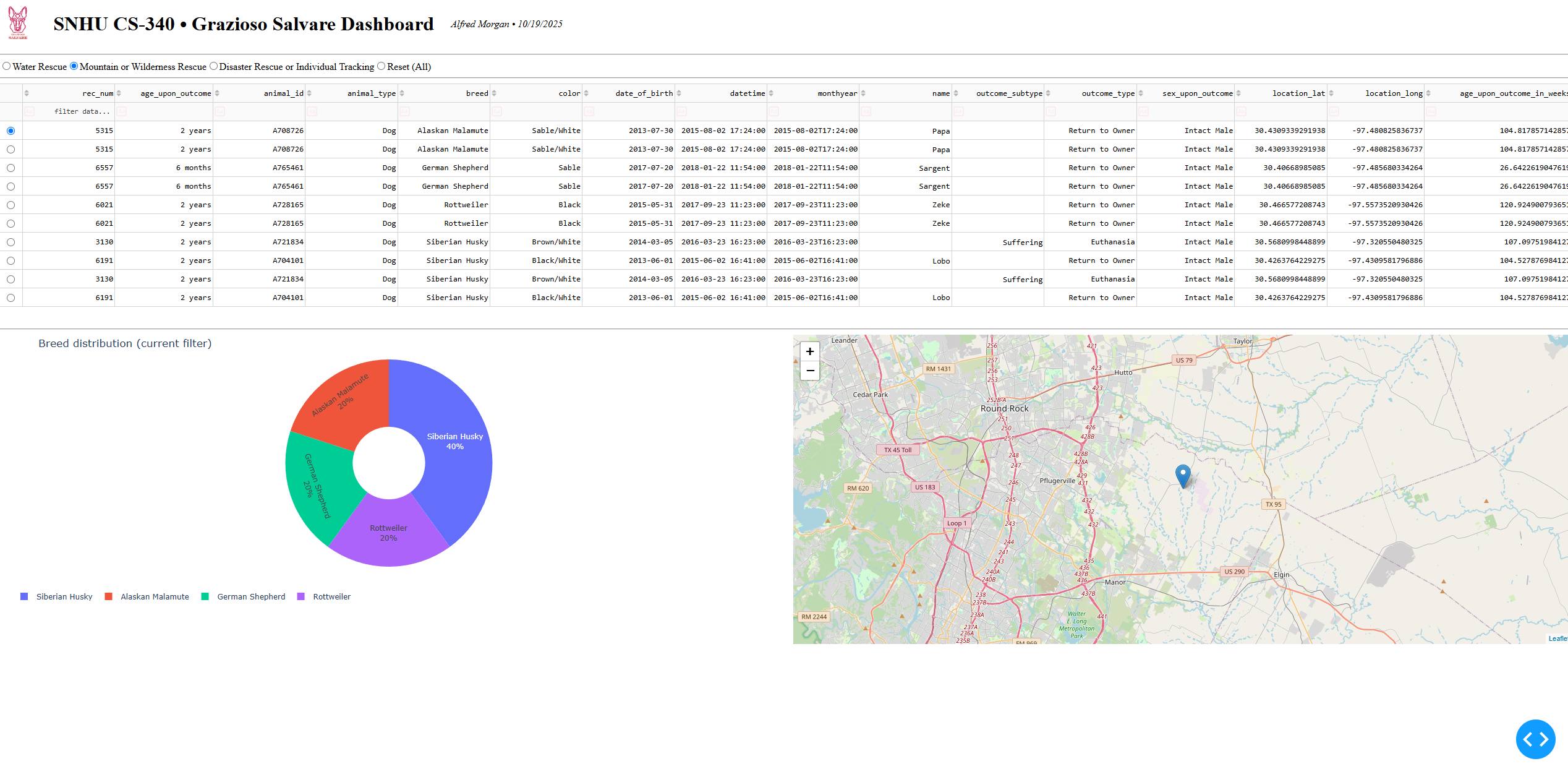


Figure 4: “Disaster Rescue or Individual Tracking” filter showing narrowed results and single map marker.

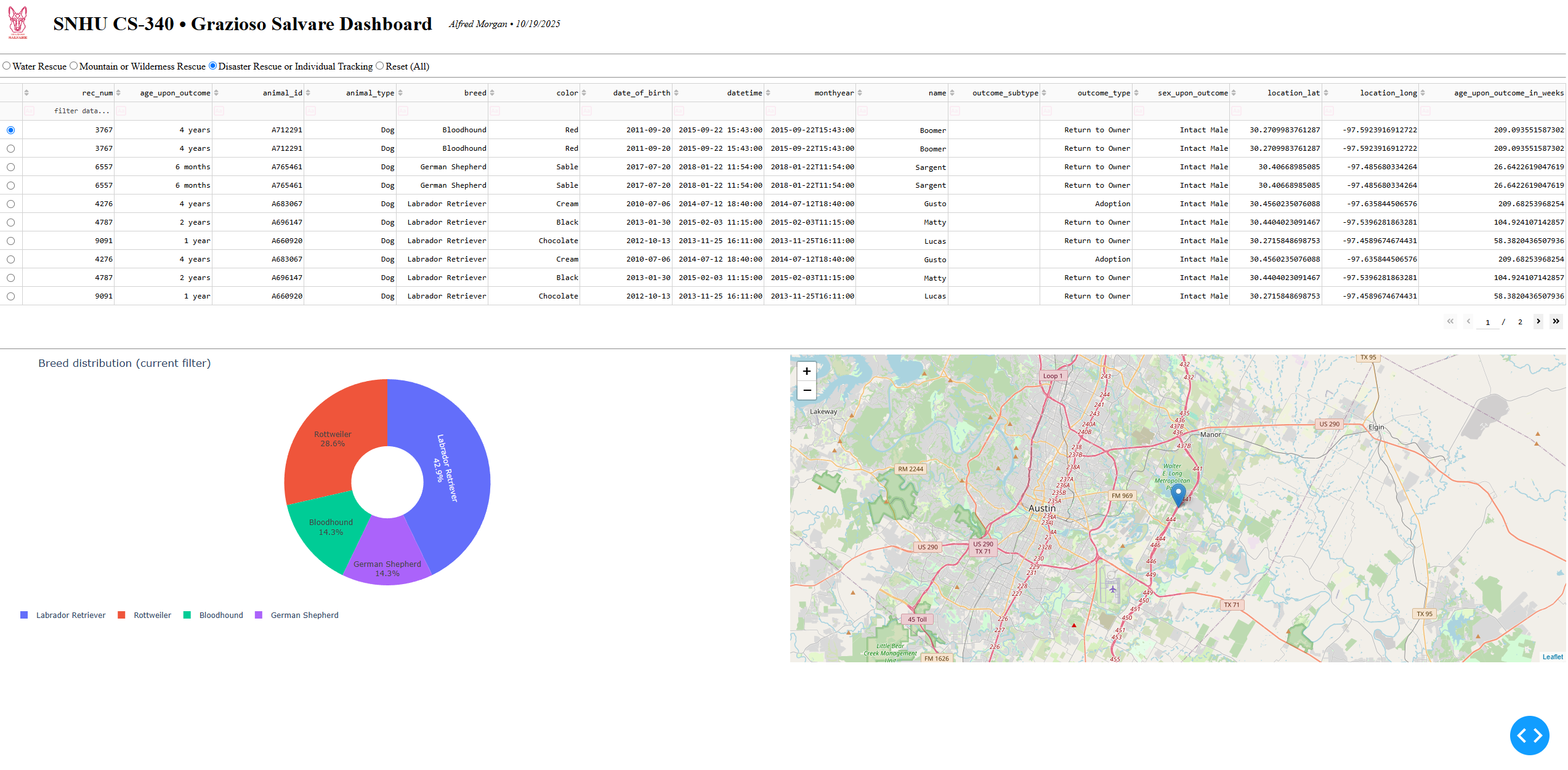
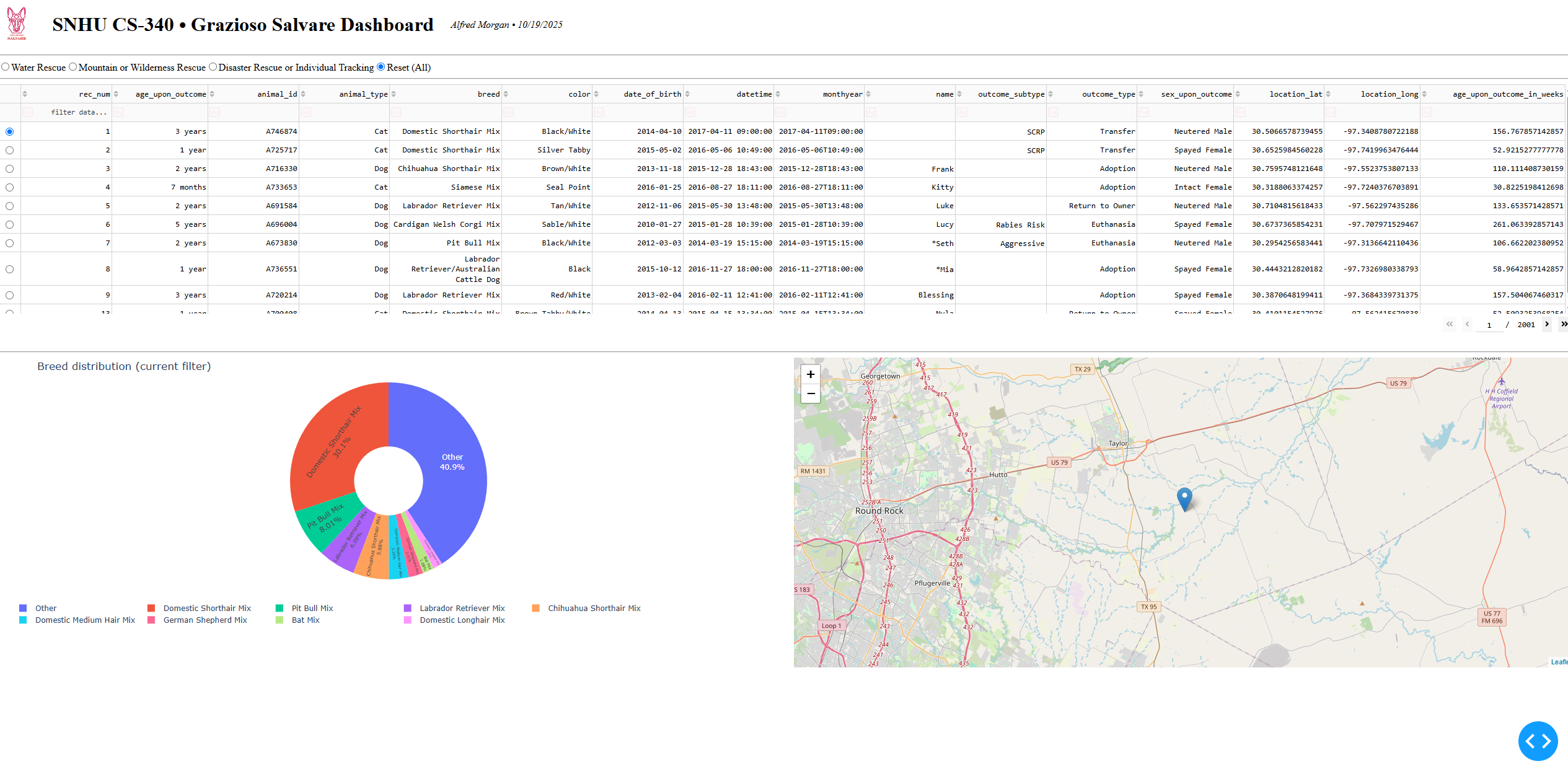


Figure 5: “Reset (All)” reloading full dataset with chart and map restored to global view.



The project used several tools to create the database and the view (UI) for the application. MongoDB Community Server version 5+ was selected for the data store because it is schema-less, highly scalable, and maps very well to a Python dictionary. Documents can also be queried by specifying conditions such as $in, $gte, $lte, and others, which allows filtering of variable animal records very easily. The pymongo Python library is used to manage the connection and safely perform CRUD operations on the database. The Dash library provides the controller and the view for the application, which means it is entirely possible to build a web application using only Python without any HTML or JavaScript knowledge. Dash provides layout components (html.Div, dash\_table.DataTable, dcc.RadioItems) as well as specific Leaflet and plotly components for map and chart building. The charts use plotly.express pie charting to build a dynamic graphic that updates on the fly. The map visualization is implemented using dash\_leaflet for professional-grade graphics and geographic display and interactivity. Development and testing were completed entirely inside Jupyter Notebook, which acts as a shell for the application using the JupyterDash Python package. The notebook then also displays both plots as the app runs.

MongoDB was chosen as the model component of the MVC because it is a document-based NoSQL database that stores JSON-like documents. The data in a MongoDB document closely resembles the Python dictionary type, which makes it easy to pass data between the database and the front-end without conversion steps. MongoDB is known for supporting atomic updates and flexible schemas, which allow applications to survive some inconsistencies or missing data. The read and query performance was excellent even on a larger dataset of 20,000+ animal records.

Dash was chosen as the view/controller component because it allows a developer to build interactive web dashboards using only Python. Dash acts as the view and as the controller, meaning it both serves the user interface (view) and manages how that input updates and changes the displayed data. Dash callbacks link specific interface elements to backend data operations such that any filter change or record selection automatically updates the charts and map. The entire project can be shown as a demonstration of the Model-View-Controller design pattern in which MongoDB is the model, Dash is both the view and the controller, and the Python application logic can be seen as the glue between the other components.

The environment to reproduce the project consists of a few things. First, MongoDB must be installed locally and the aac.csv dataset from the Austin Animal Center must be imported into the aac database with the animals collection. A database user aacuser should be created with password watchdogman425 and granted read-write privileges. The necessary Python packages can then be installed using pip and the list of requirements found in the project files. The packages include jupyter-dash, dash, dash-leaflet, plotly, pymongo, pandas, and matplotlib. After setting up the environment, the project can be run by opening the ProjectTwoDashboard.ipynb notebook and running the last code cell, which will initialize and launch the Dash application to a local server url of form http://127.0.0.1:8050/. Clicking on the server link will open the interactive dashboard in the default web browser.

There were a few challenges encountered during the project development and problem solving, all of which were debugged and resolved through multiple iterations of testing and inspection. The original image for the logo resulted in a FileNotFoundError because it did not match the working directory of the notebook, and this was easily fixed by verifying the correct filename “Grazioso Salvare Logo.png.” MongoDB had some authentication errors which were resolved by specifying the authSource parameter in the connection URI string. After adding the loading spinners, the Leaflet map component disappeared temporarily until the container was given a fixed minimum height of 300 pixels. The “Reset (All)” radio button took a long time to reload after switching from another filter, so the read() method was changed to use a direct query to the database (read({})) to improve query performance. To improve the readability of the pie chart, the number of items displayed was capped, and remaining values were placed into an “Other” category. Finally, the table and map responsiveness was improved by adding table virtualization and loading spinners to all major components.

The tools MongoDB and Dash were chosen for the project because they combined flexibility, speed, and visual interactivity for the application. MongoDB is fast and schema-less, making it easy to work with both complete and variable data structures. Dash is a Python framework that can be used to build a complete web application by combining backend controller logic and frontend interactivity in one language, significantly simplifying development. The two combined tools make it possible to produce a responsive, visually attractive, and easily maintained solution for the needs of Grazioso Salvare.

In summary, the completed Grazioso Salvare Dashboard meets all of the project’s requirements. A working CRUD backend is paired with a responsive, visual frontend. Both integration and demonstration of the connection between MongoDB and Dash are clear. The dashboard filters and displays animal data for various rescue missions, and supports data visualization and mapping. All data representations display as expected, and multiple screenshots are provided to confirm that each component is working as intended, successfully completing the development and testing phases.