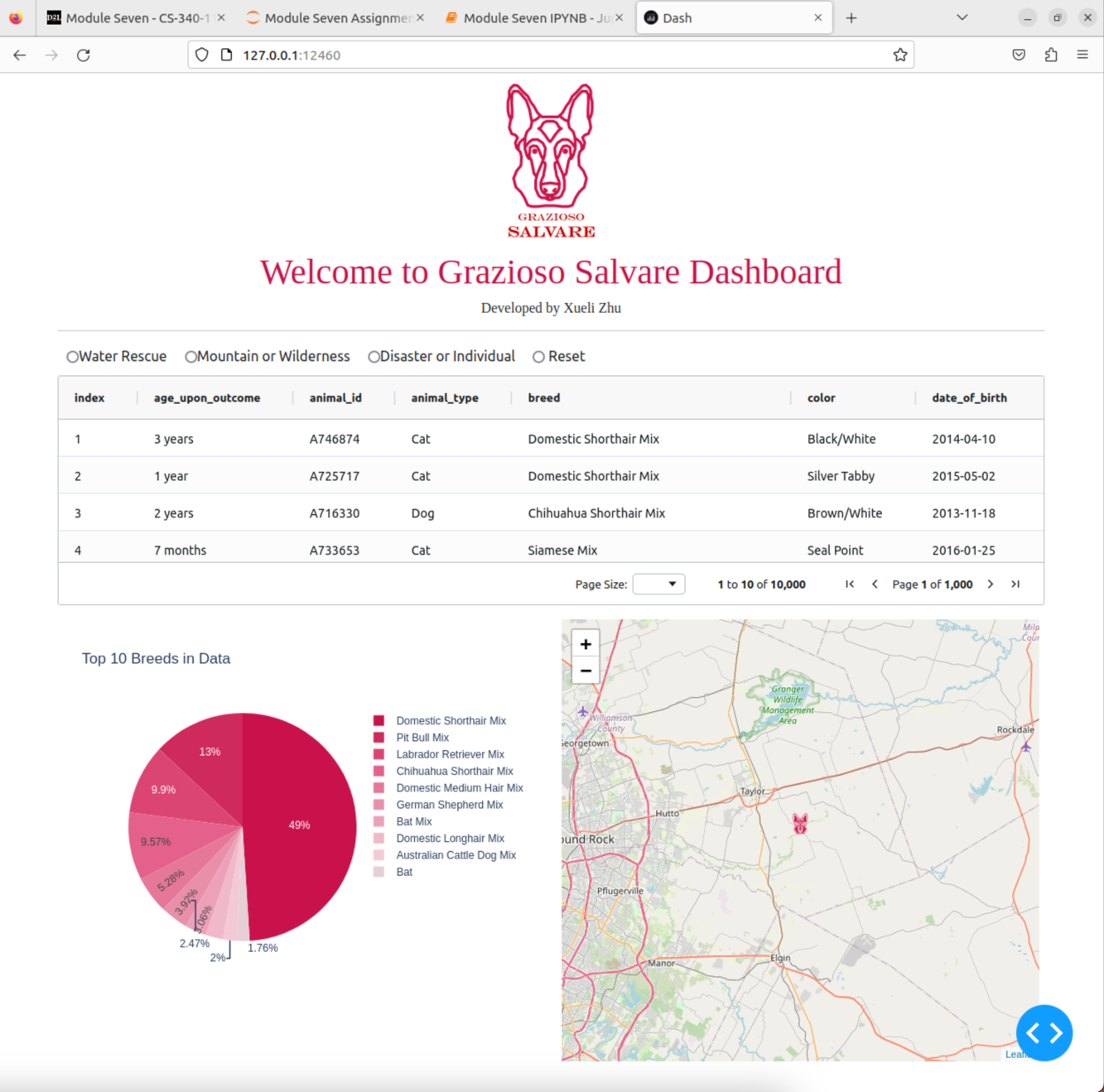
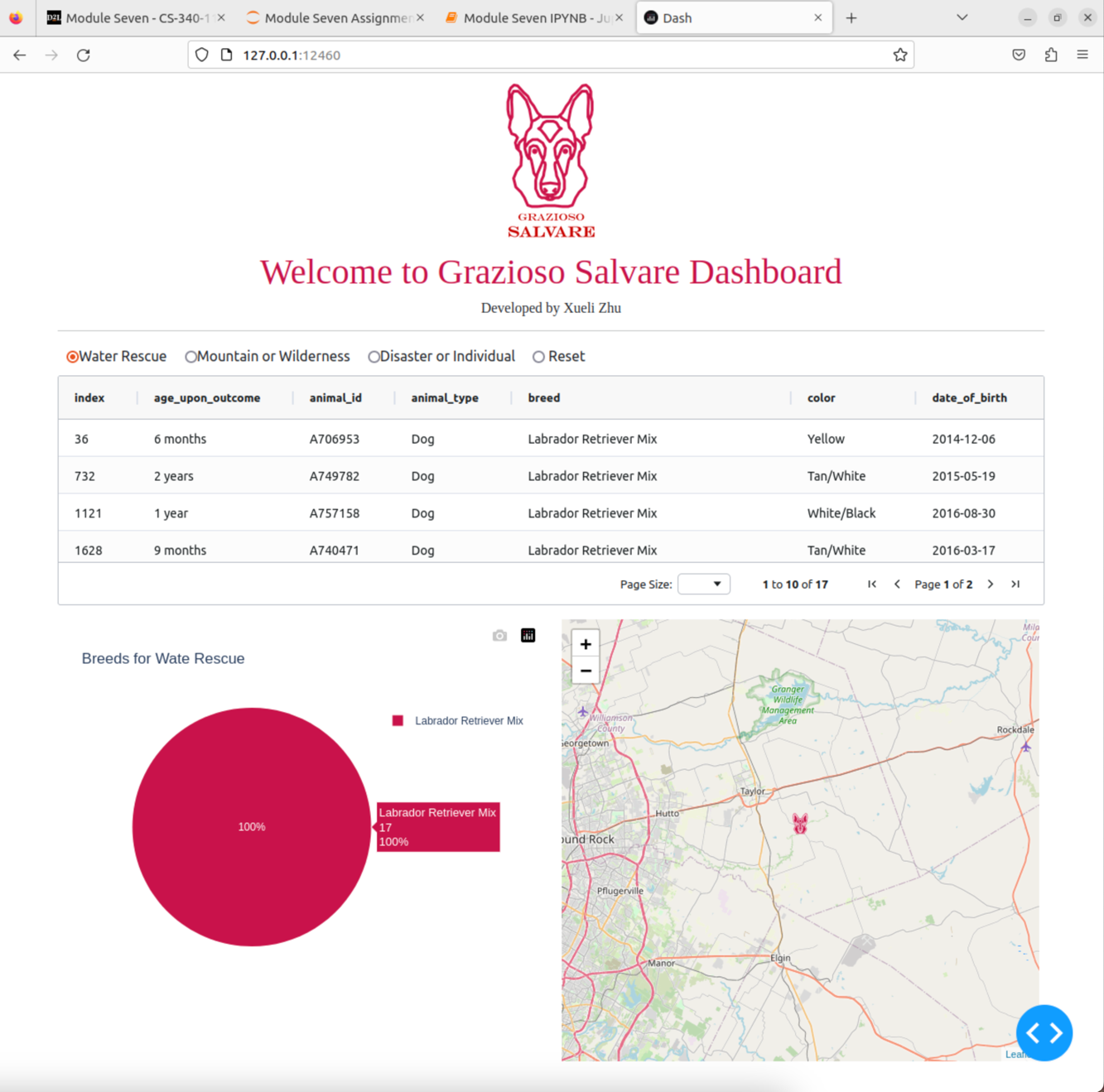
**Required functionality**

The project involves the creation of an interactive dashboard using Dash and Plotly where users can filter data regarding animal shelter outcomes. It provides a dynamic map with markers for specific breeds, a pie chart for breed distribution, and an option to filter results based on provided categories such as breed, age, and sex upon outcome. This functionality allows user to select filter options and dynamically update the data shown, such as breed types and their corresponding counts in the pie chart. Plus, custom drop pin is placed on the map for highlighting various shelter outcomes. Enhancing the visualization with a personalized touch that reinforces Grazioso Salvare’s brand identity. The pie chart will update dynamically depending on the filter choice, and the map markers add context about the location of the shelters. I have taken screenshots during testing and deployment that show map updates with different filter applications to make sure that the functionality of the dashboard works as expected, presenting the data correctly in every filter scenario.

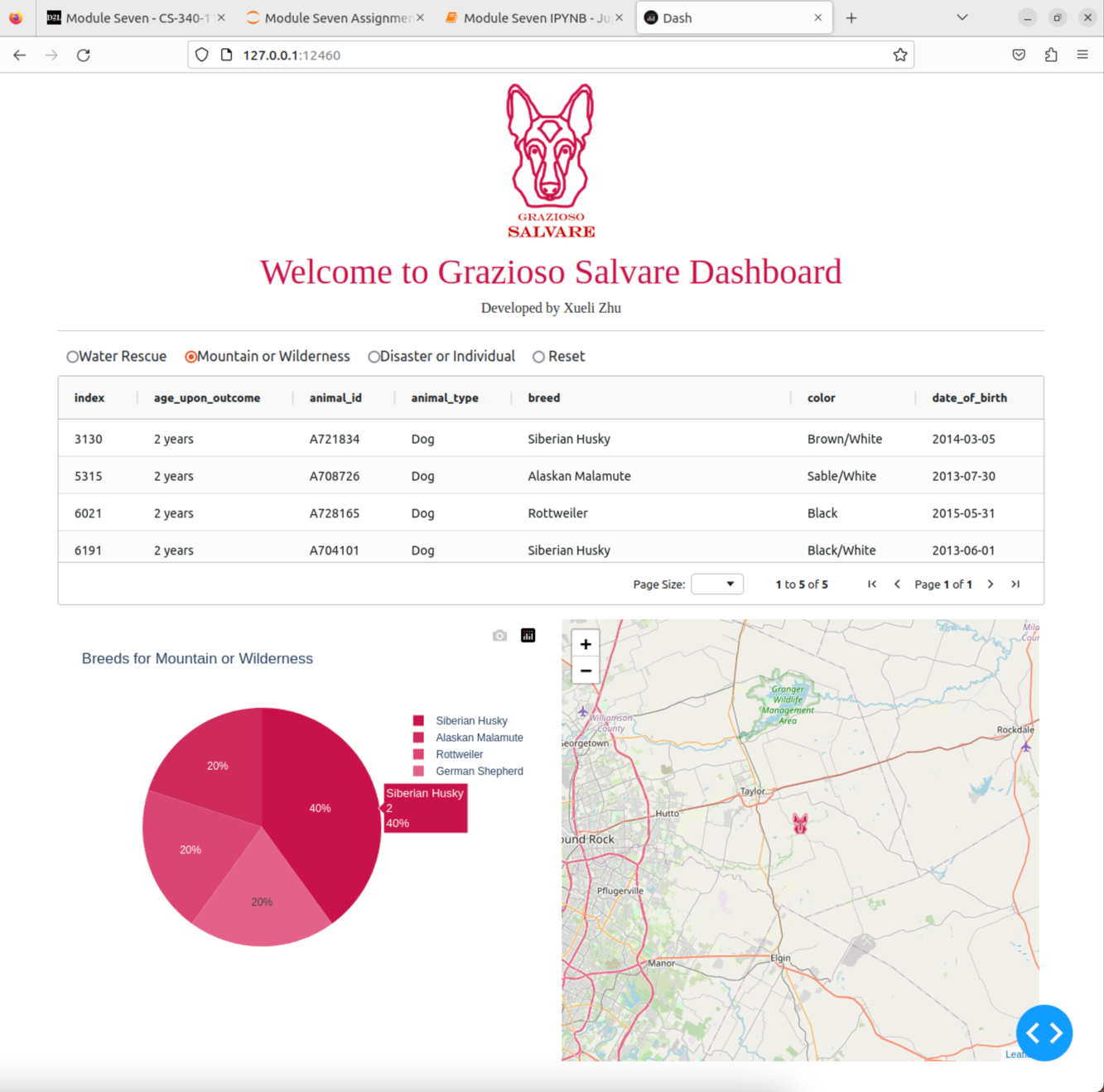
This is the program's main screen. At the top, you'll see the Grazioso Salvare logo with a tailored link leading to their website. Under the logo comes a title welcoming users, which is displayed in the same color as the logo to provide a sense of continuity in appearance. Then comes my name, followed by the data table that presents all the animals from the spreadsheet. A pie chart shows the ten most common breeds, displayed with ten custom colors that complement the shade of the logo to reinforce a consistent brand image. Finally, a map shows the location of the first animal listed, of course with a customized drop pin(their logo) to tie everything together.



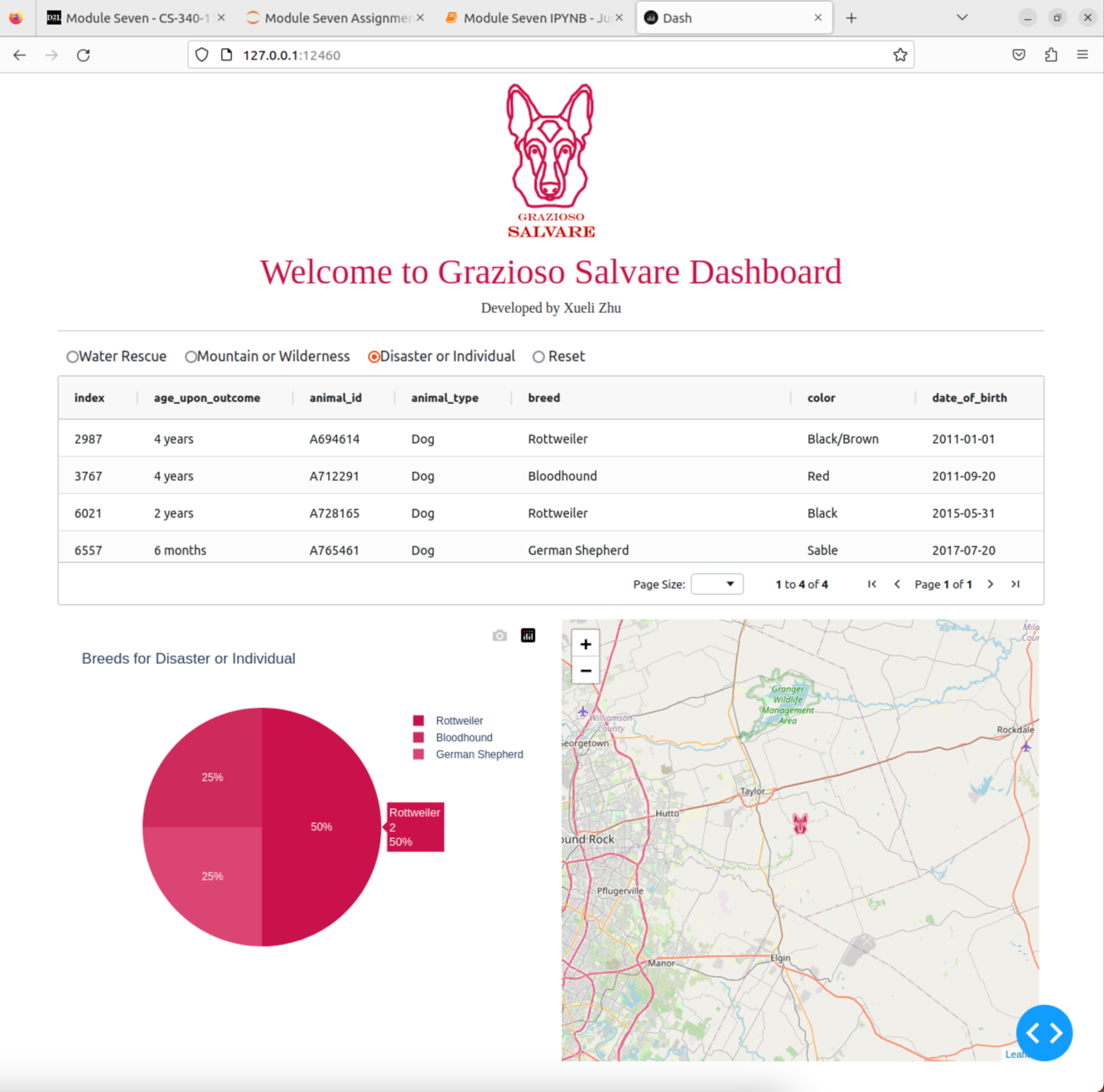
This screen displays the updated data table, which aligns with the specific requirements for water rescue dogs. The pie chart has also been updated to reflect the breed types and counts for water rescue dogs, replacing the previous chart showing the top ten most common breeds.



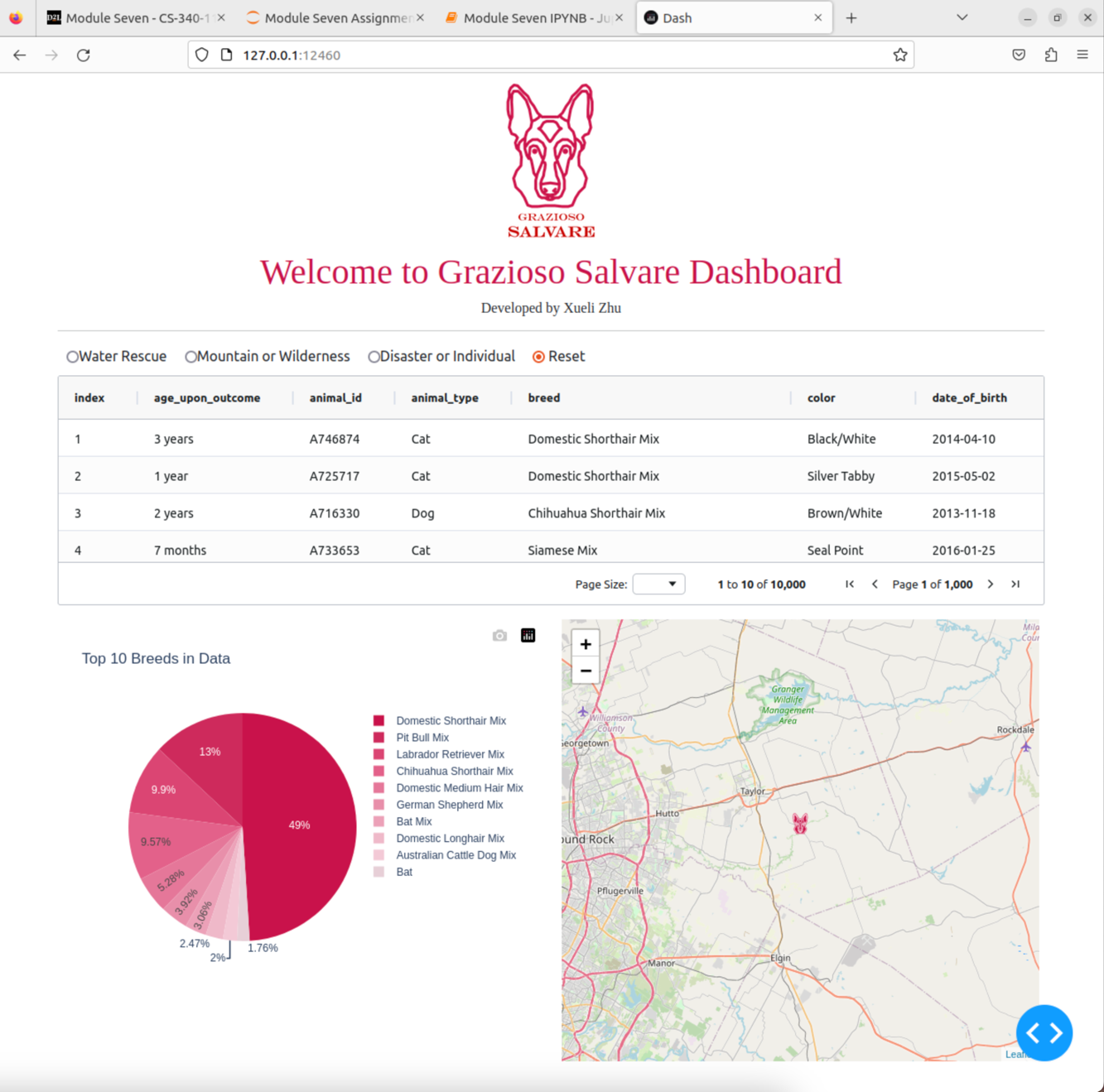
Same concept as above, with results for mountain or wilderness rescue dogs.



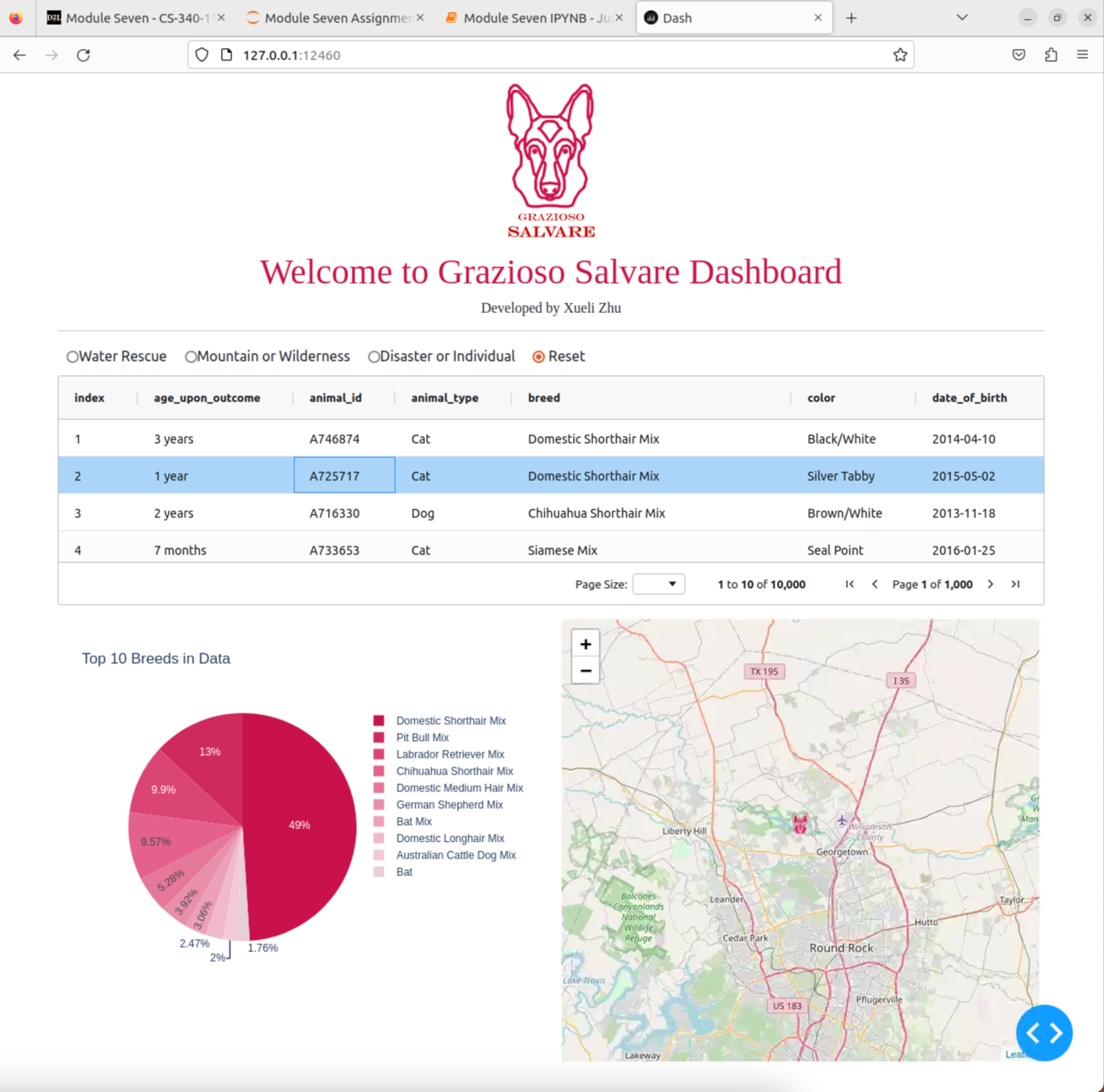
And this screen shows you the result of the disaster or individual rescue dogs.

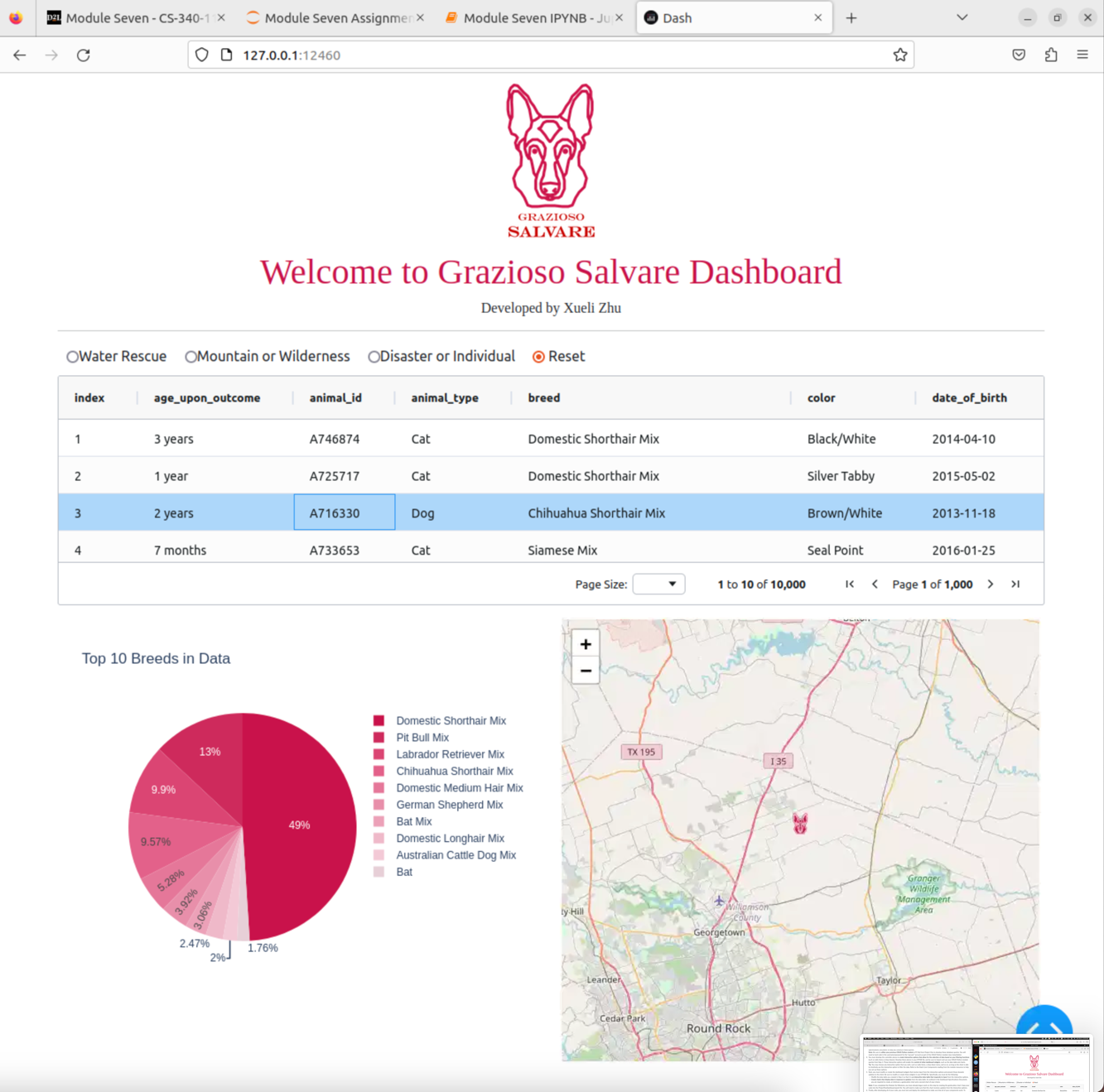


This screen demonstrates that by clicking 'Reset,' the data table will refresh to display all the animals from the spreadsheet once more, accompanied by an updated pie chart.



Below are two example screenshots showcasing the updated map location of the selected animal.





**Tools used**

It does include the MongoDB, Python, and Dash framework. For this solution, MongoDB was chosen as the model component due to its flexibility in handling unstructured data, scalability, and ease of integration with Python through the “pymongo” library. MongoDB's document-based structure makes it well-suited for managing complex datasets, such as the Austin Animal Center Outcomes Spreadsheet, without requiring rigid schema definitions. This enables dynamic querying and filtering of data, which is essential for our dashboard's interactive features. The Dash framework, developed by Plotly, was used to provide the view and controller structure for the web application. In Dash, the integration of Python with interactive web components, such as graphs and maps, is relatively simple. It also supports real-time updates and callbacks, it’s crucial for filtering and dynamically adjusting the displayed data. Because Dash is integrated with Plotly and Leaflet, visualizations are smooth and interactive for both the pie chart and map.

Resources used include Dash (<https://dash.plotly.com/>) and MongoDB (<https://www.mongodb.com/>).

**Steps**

Data was gathered and stored in MongoDB to start the project, which allowed flexible querying and filtering. The Dash framework was set up to create the interactive dashboard, integrating pie charts and maps. Filters were implemented to allow users to select breed categories, updating the data dynamically. Custom markers were added to the map to display shelter outcomes. Finally, the app was tested and deployed, ensuring functionality through visualizations and real-time interactions.

**Challenges**

The biggest challenge was filtering data in MongoDB with Python, especially when complex queries involving many conditions were to be filtered. Initially, MongoDB query syntax was challenging to filter data based on multiple fields and conditions. This was resolved by refining the queries to ensure that proper logical operations were applied. Further, I used Pandas to better manipulate data. This allows easier handling of difficult filtering and transformation jobs in an overall smooth and accurate processing of data.