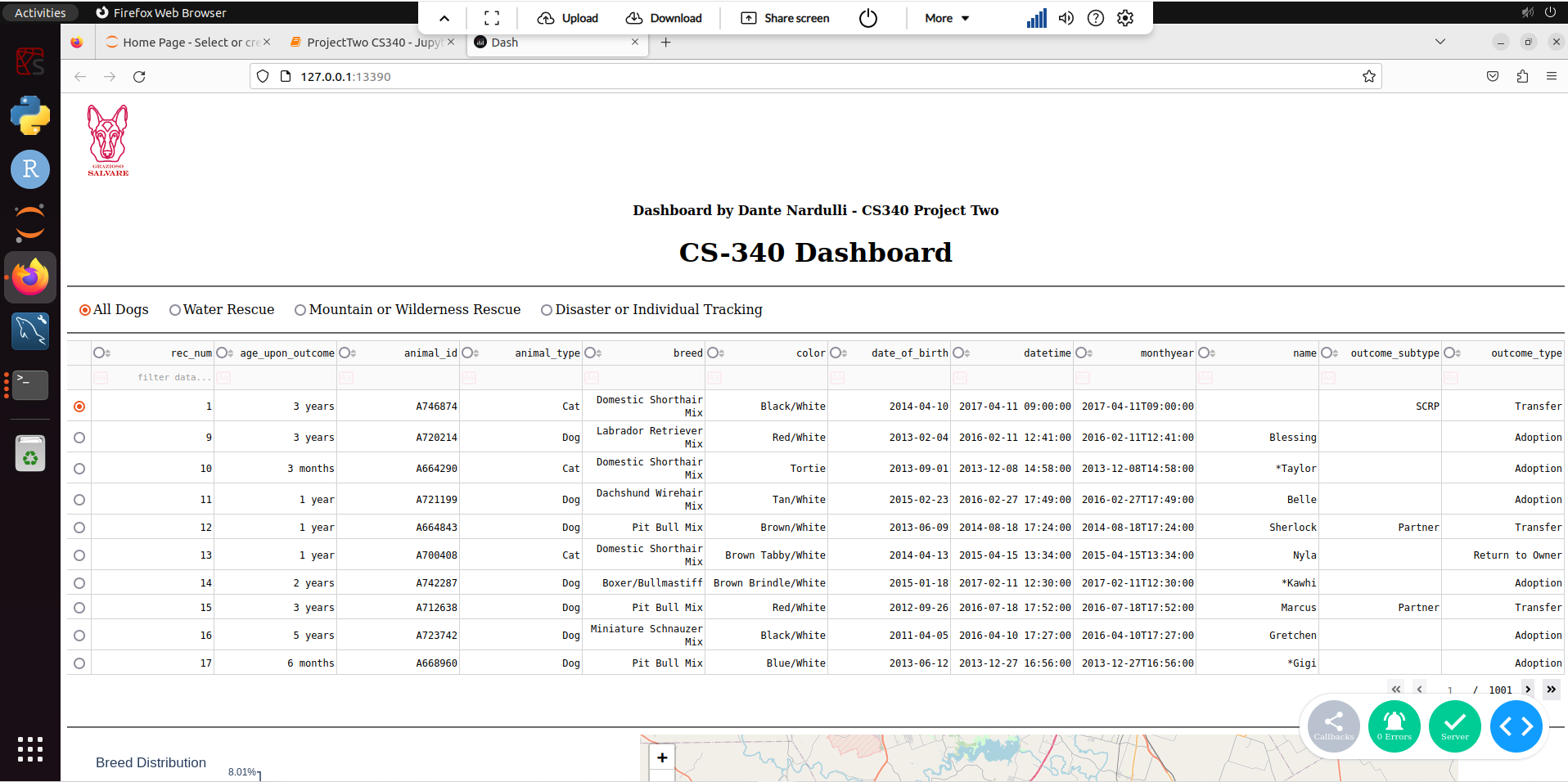
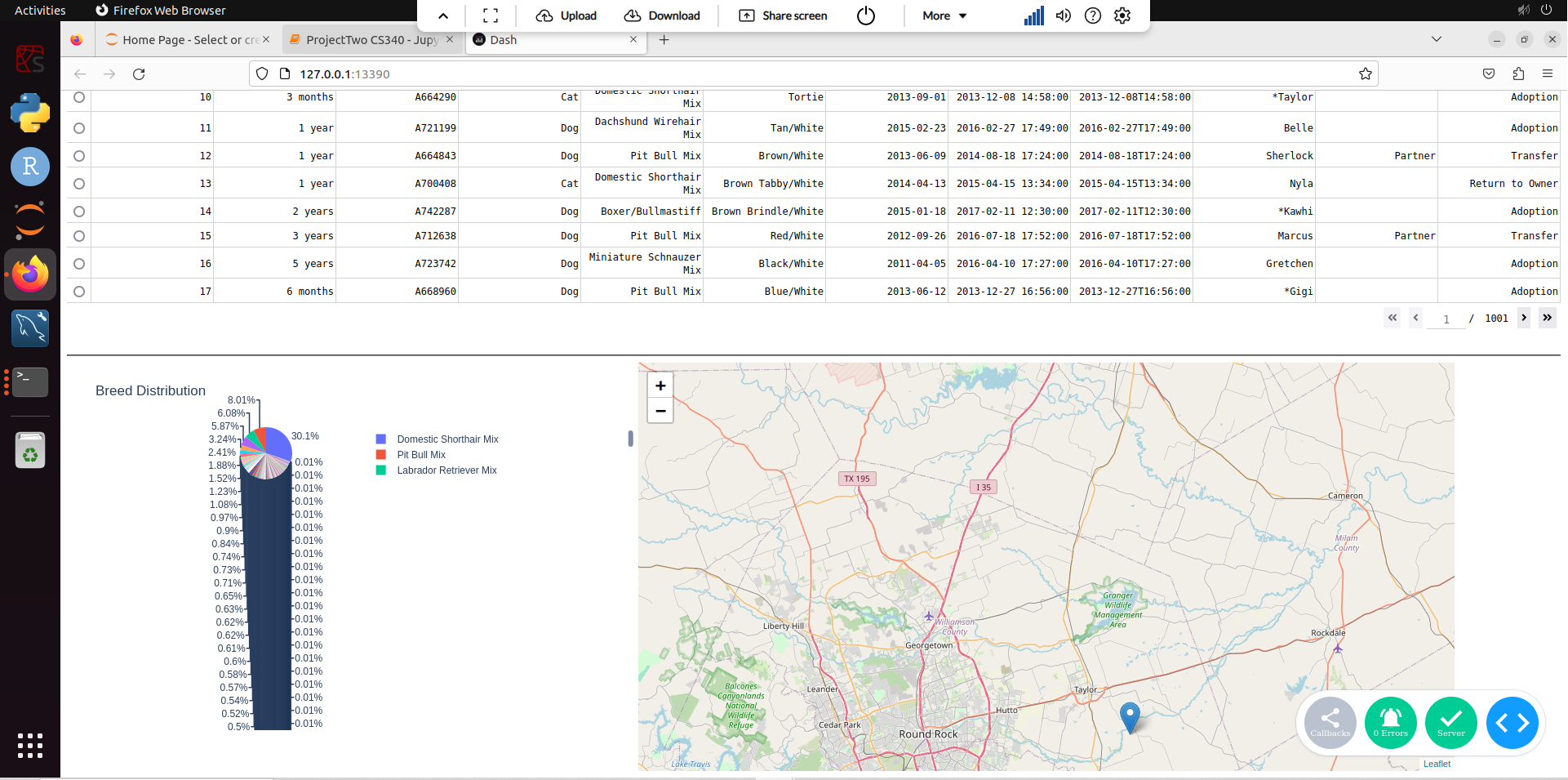
CS340 Project 2

**Functionality**

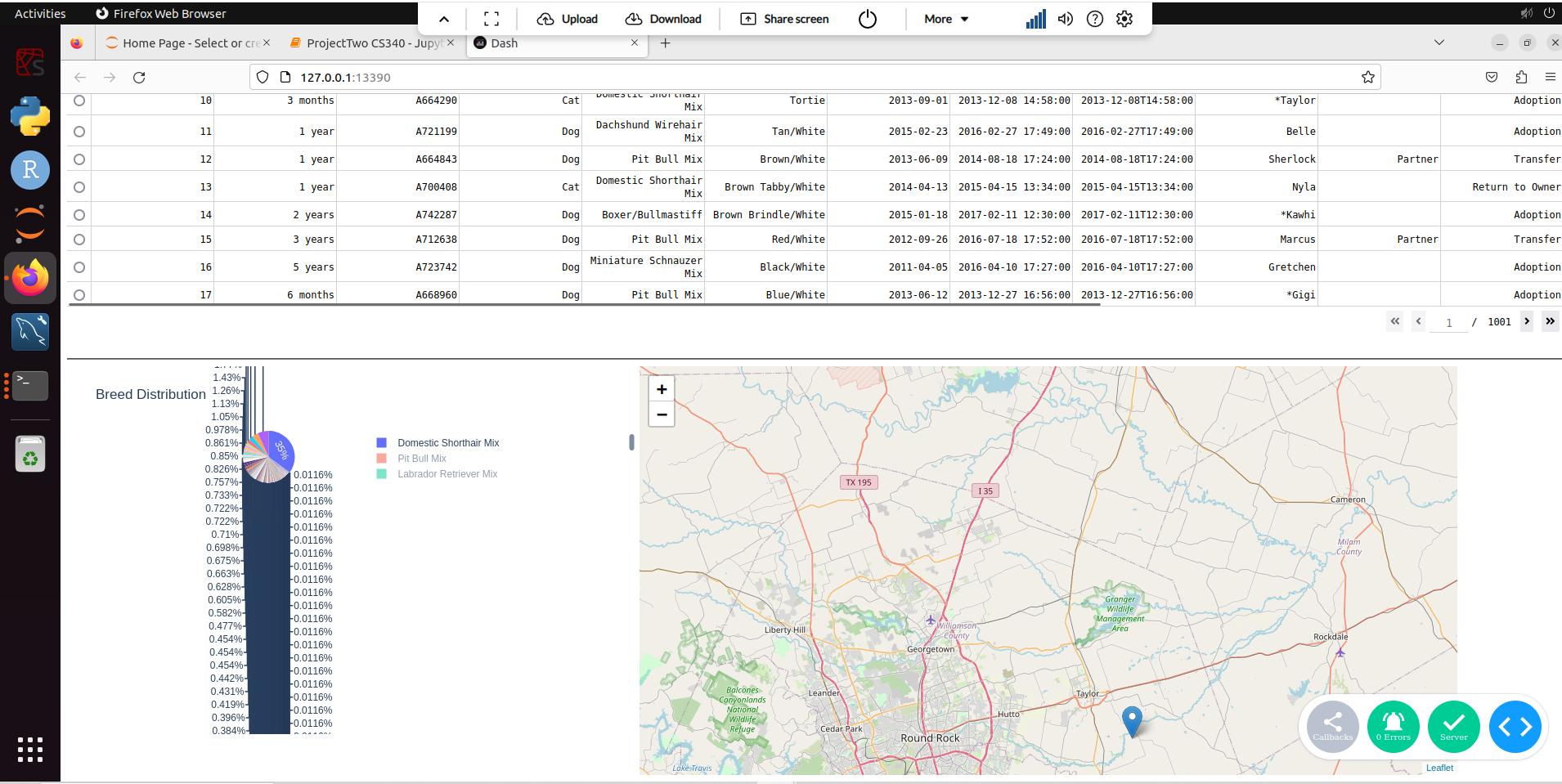
For this project, I created an interactive interface for users to view and filter data about available dogs in shelters. The interface has a data table that displays an unfiltered list of shelter animals, which can be dynamically updated based on user-selected filters. I included interactive filters that allow users to select data based on the dog’s breed, age, and rescue type (water rescue, mountain, or wilderness rescue, etc.). Also included is a geolocation map that displays a map of Austin, Texas, with markers indicating the locations of available dogs, which can be clicked to view details about the dog. Lastly, I included a pie chart that visualizes the distribution of dog breeds on the main interface and each filtered rescue page.

**Screenshots**Main Interface:

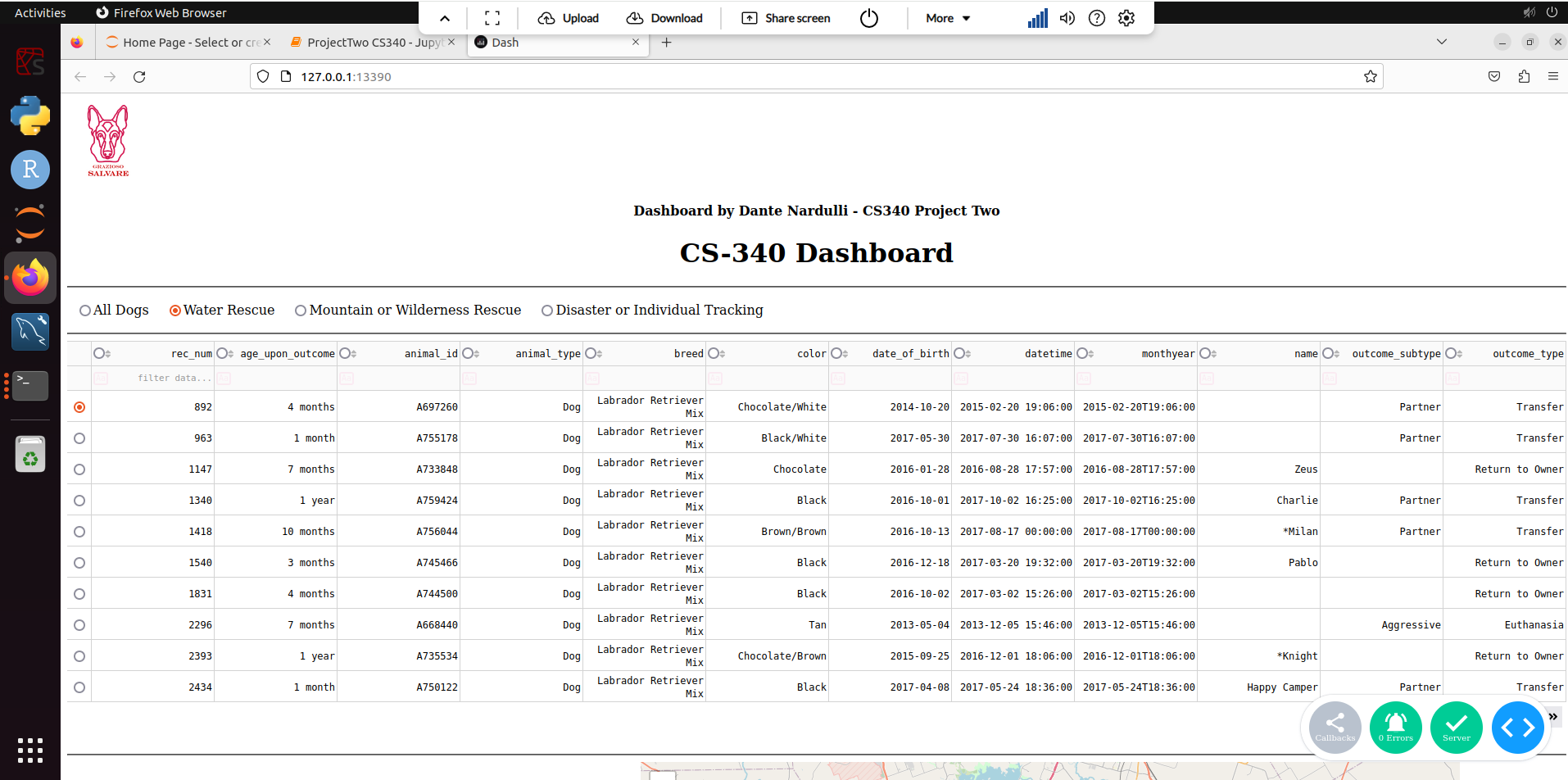


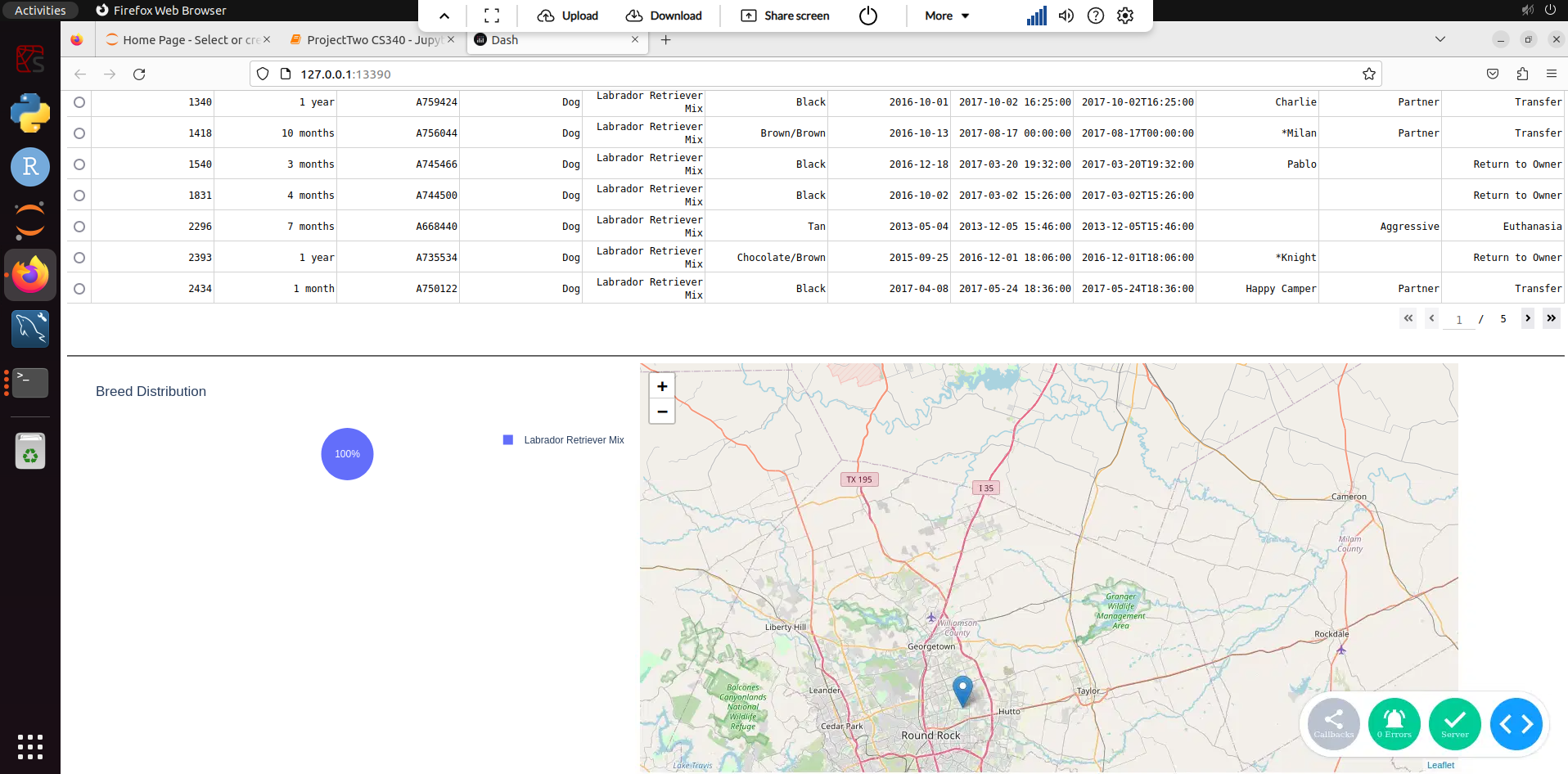
Map and Pie Chart:  


Adaptable Pie Chart:

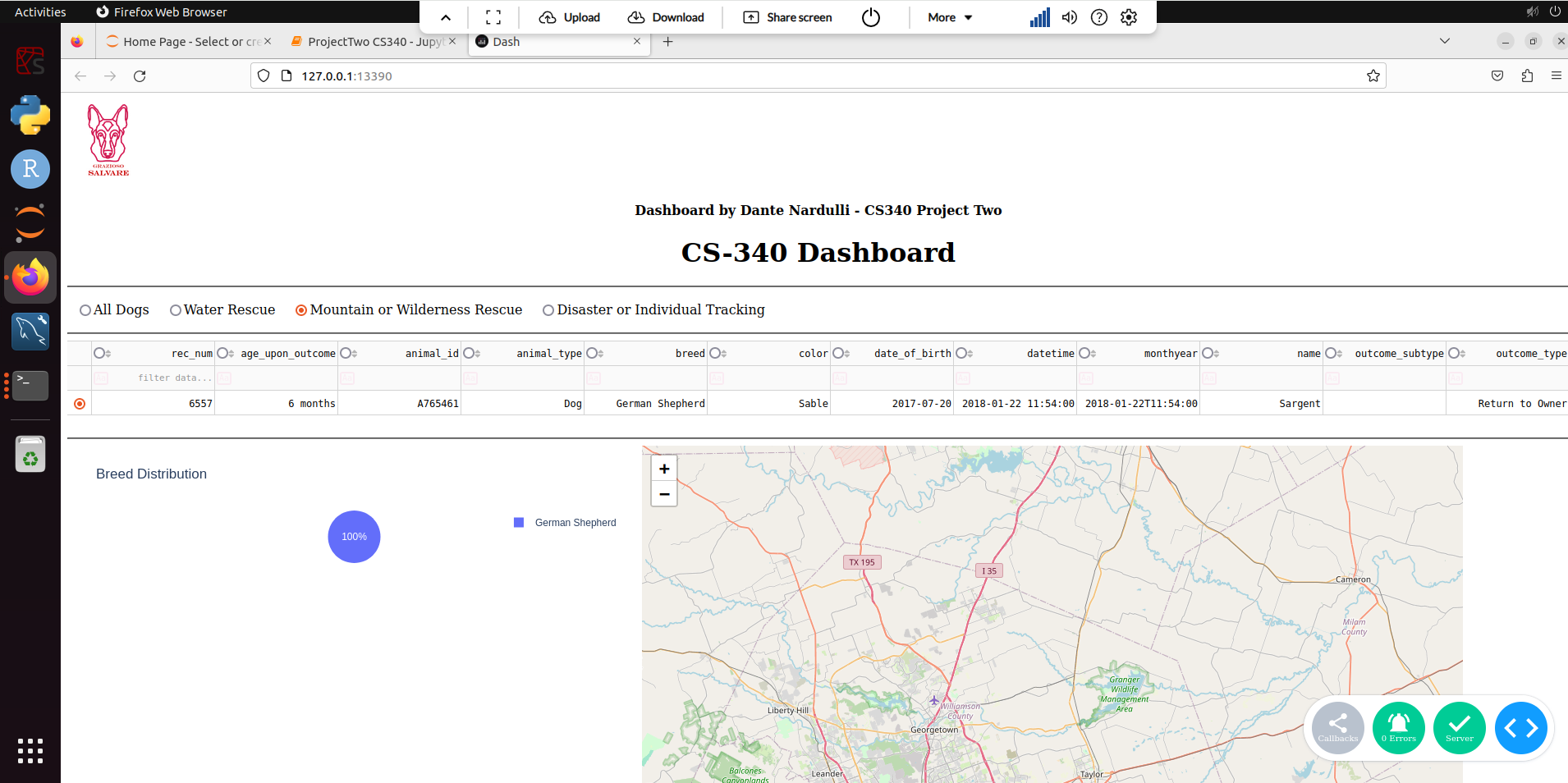


Rescue filter (Water):

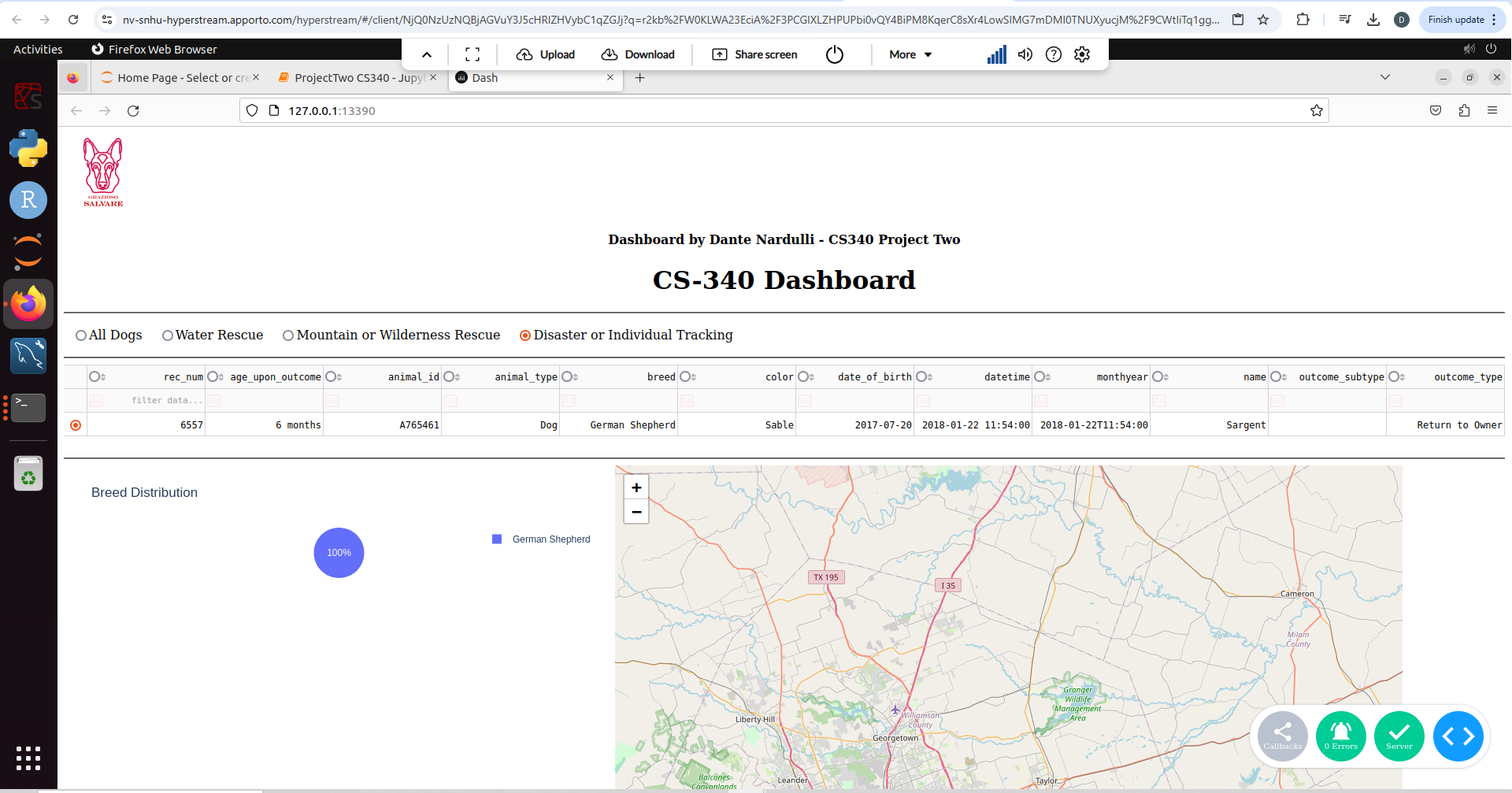




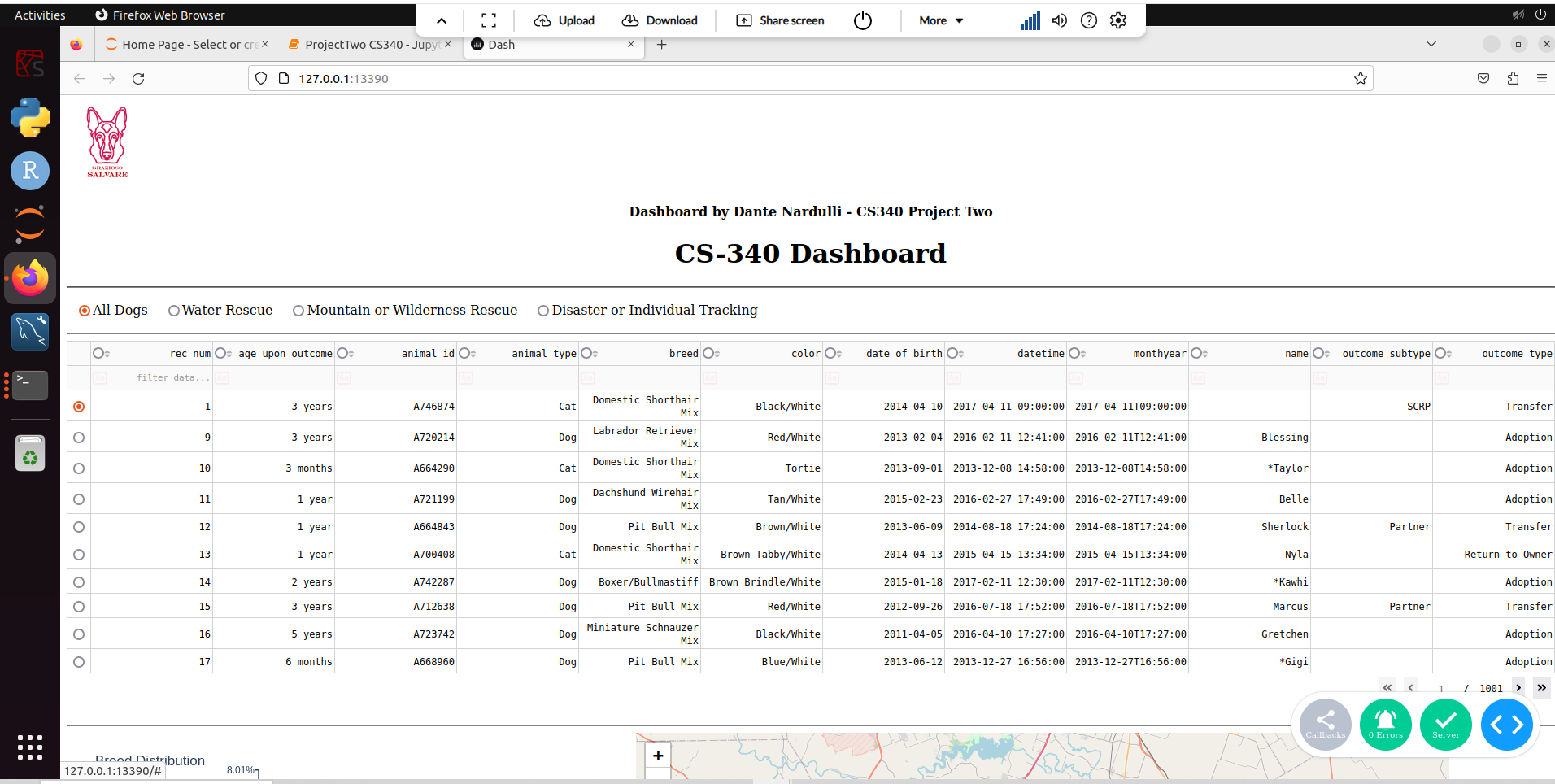
Rescue Filter(Mountain):



Rescue Filter (Tracking):



Reset to Main Interface:



**Tools and Tech Used**

Dash is a python framework used for building web applications with a simple and declarative syntax. Dash handles the view and controller logic for the dashboard, connecting the user interface components. Dash is a powerful tool for building interactive, web-based data visualizations with minimal configuration. It provides a clean way to set up interactive components with minimal boilerplate code, which was ideal for this project. Dash callbacks handle user inputs and update the view accordingly. These callbacks interact with MongoDB to retrieve filtered data and pass it to the view.

MongoDB is a NoSQL database used to store animal shelter data. It allows for easy storage and retrieval of JSON-like documents, which is a natural fit for storing structured animal data. It was very easy to store and filter data by dog breed, age, rescue type, etc. MongoDB has a great choice due to its flexible data storage. It integrated seamlessly with Python using PyMongo and MongoEngine, which I used for CRUD functionality. MongoDB is highly scalable, making it suitable for storing large amounts of data that may grow over time.

Plotly was utilized because it is a graphing library used to generate visualizations such as pie charts and maps. Plotly integrates well with Dash and provides a wide range of interactive graphing capabilities, including geospatial mapping.

I used Dash Leaflet to integrate maps into Dash applications. This library was used to display a map of Austin with markers for available dogs in different shelters. It provides interactive maps with various customizable features.

**Project Steps**

The first step was to setup the database. using the MongoDB database with the animal shelter data. A Python module was created to provide CRUD functionality for querying and manipulating the database. The second step I created the dashboard using Dash’s HTML and component libraries, including dash\_table and dash \_leaflet for the map. After that I focused on interactivity with filters were added using Dash components like radio buttons and drop-down menus. Callbacks were set up to filter the data and update the map and data table accordingly. For the fourth step I made a pie chart was added to visualize the distribution of dog breeds, and a geolocation map was integrated to show the locations of the available dogs. The last step I made sure the dashboard was thoroughly tested to ensure all components function as expected, with user interactions updating the view dynamically.

**Challenges**

I ran into a lot of issues and the majority came with writing clear code. Jupyter notebook was very strict on indentation and many times would not tell me which line was messed up which made it even more confusing trying to troubleshoot. The solution was just thorough testing and debugging to ensure the code would run as expected. Another issue I ran into was handling user input and callbacks. To solve this issue, I implemented checks to ensure that user inputs (such as selected rows or filter values) are not “none” before proceeding with updates to the dashboard. This ensures that the application doesn’t crash when no data is selected.

**Conclusion**

This dashboard provides Grazioso Salvare with a user-friendly interface to interact with shelter data, filter it based on various attributes, and visualize the results on an interactive map. The project successfully integrates MongoDB as the backend database with Dash as the frontend framework, delivering a functional and scalable solution.