**CS 340 README Project Two Steven Armenta**

*Use this template to complete your README file. When completing the template, keep the headings as they are so that your document has a clear organization. Remove the italicized prompt text after you have completed each section for a polished final document.*

**About the Project/Project Title**

*This project consists of a full-stack web application designed for Grazioso Salvare to identify suitable rescue dogs from animal shelter data. The application includes a Python CRUD module (animal\_shelter.py) that interfaces with a MongoDB database, and a Dash-based interactive dashboard that allows users to visualize and filter the data according to specific rescue dog requirements.*

**Motivation**

*This application was developed to help Grazioso Salvare efficiently identify dogs from local animal shelters that match specific criteria for search-and-rescue training. By providing a user friendly dashboard interface with filtering capabilities for different rescue types (water rescue, mountain/wilderness rescue, and disaster/individual tracking); the company can quickly identify potential rescue dogs based on breed, age, and other characteristics.*

**Getting Started**

*To begin this program, follow these steps:*

***Prerequisites***

* *Python 3.x*
* *MongoDB*
* *Access to animal shelter data*

**Installation**

*A current version of Python to run both the .py and the .ipynb files*

1. *Ensure you have Python installed (version 3.x recommended).*
2. *Install the required dependencies using pip:*

*pip install jupyter-dash dash dash-leaflet plotly pandas pymongo*

1. *Set up MongoDB with appropriate authentication:* 
   * *Create an admin account with full privileges*
   * *Create an 'aacuser' account with appropriate access rights to the database*
2. *Import the animal shelter dataset into MongoDB*
3. *Ensure the MongoDB service is running*

**Usage**

*This Python module has CRUD Operations to interact with the animal shelter database.*

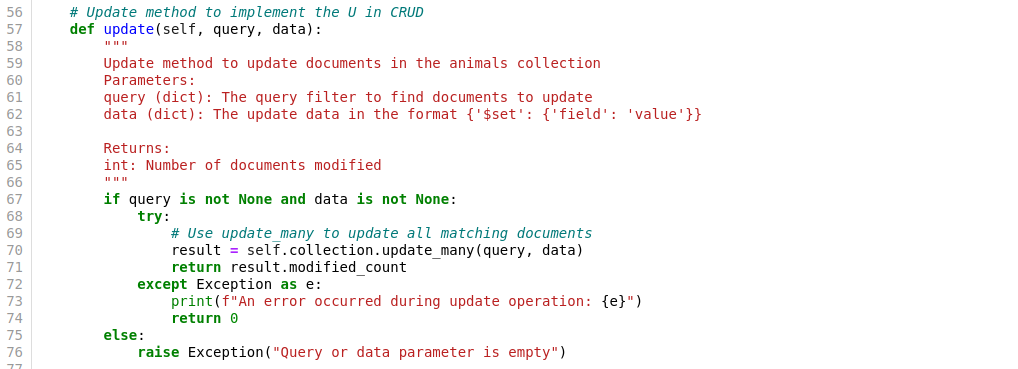
**CRUD Functionality**

*The module includes the following functions:*

1. ***Create:*** *Inserts new records into the database*
2. ***Read:*** *Finds records within the database based on specific queries*
3. ***Update:*** *Updates existing records that match specific criteria*
4. ***Delete:*** *Removes specified records from the database*

**Code Example**

Code example below shows the Update method to implement the U in CRUD.



**Here's an example of the dashboard filter functionality:**



**Tool Selection and Rationale**

***MongoDB***

*MongoDB was selected as the database component for this project for several important reasons:*

1. ***Document-Oriented Storage****: MongoDB's document model is ideal for storing animal data with varying attributes and characteristics. Each animal record can have different fields without requiring schema modifications.*
2. ***Query Flexibility****: MongoDB provides powerful querying capabilities that make it easy to filter animal records based on multiple criteria simultaneously (breed, age, sex, etc.).*
3. ***Python Integration****: Through PyMongo, MongoDB offers seamless integration with Python, allowing for efficient data manipulation using Pandas DataFrames.*
4. ***Scalability****: As the animal shelter database grows, MongoDB can scale horizontally to accommodate increased data volume.*
5. ***JSON-like Documents****: MongoDB's BSON format aligns well with Python dictionaries, making data transformation between the database and application layer intuitive.*

***Dash Framework***

*The Dash framework was chosen for the view and controller components because:*

1. ***Python-Based****: Dash is built on Flask, Plotly.js, and React.js, but is accessible entirely through Python, enabling development of a full-stack application.*
2. ***Interactive Visualizations****: Dash provides built-in components for creating interactive charts, graphs, and maps that update in real-time.*
3. ***Callback System****: Dash's Input/Output callback system enables reactive programming patterns that automatically update the UI when a user interacts with the application.*

**Development Steps**

1. ***Database Setup****: Configure MongoDB and import the animal shelter dataset provided.*
2. ***CRUD Module Development****: Created and tested the animal\_shelter.py module with create, read, update, and delete functionality.*
3. ***Dashboard Development****:* 
   * *Set up basic Dash application structure.*
   * *Implemented data loading from MongoDB using the CRUD module.*
   * *Created interactive filter options based on rescue dog specifications.*
   * *Developed dynamic data table component.*
   * *Implemented pie chart for breed distribution visualization.*
   * *Added geolocation map with marker functionality.*
4. ***Integration and Testing****: Connected all components and tested the full functionality of the application.*
5. ***Documentation****: Complete by creating a comprehensive README documentation.*

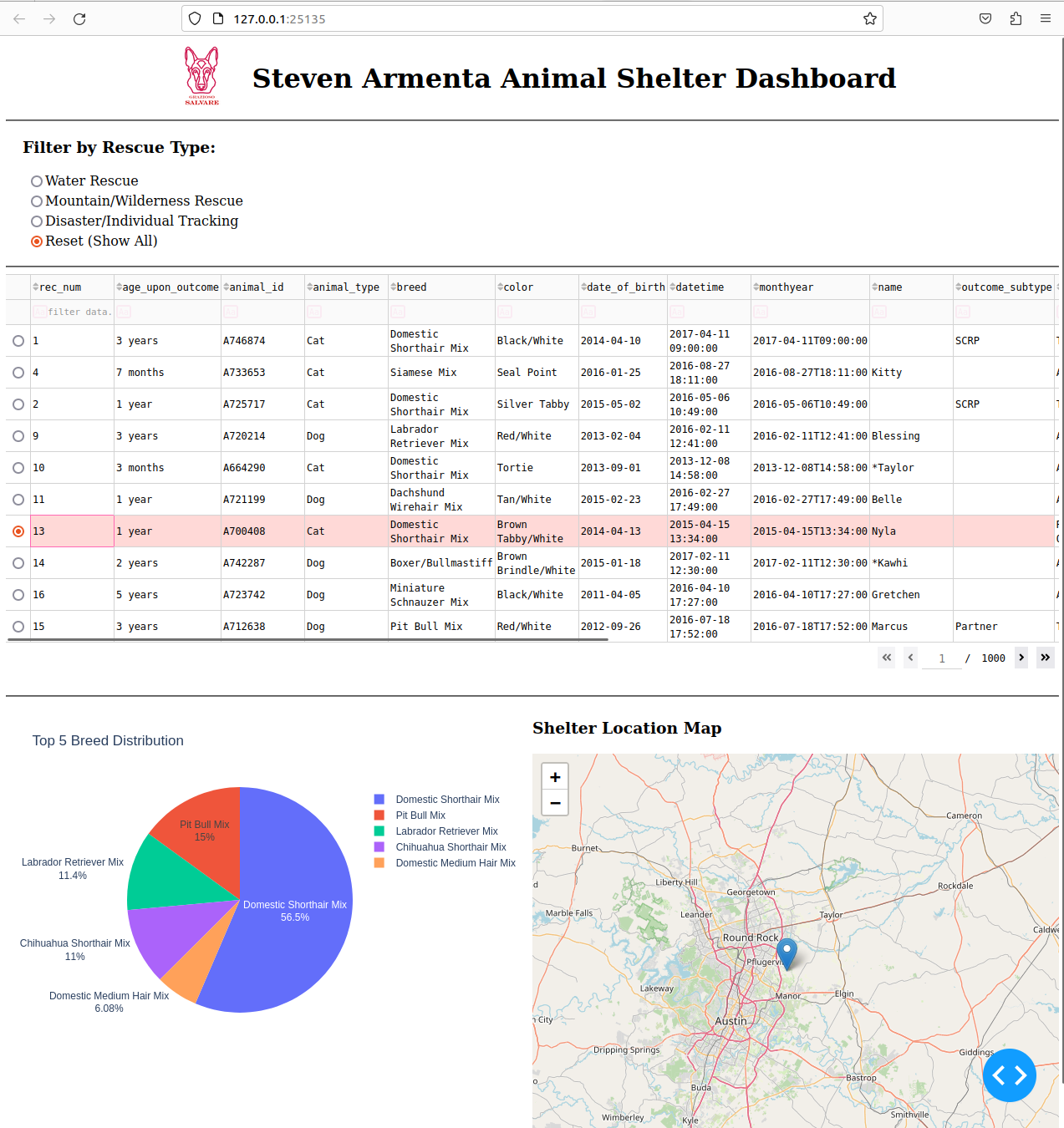
**Challenges and Solutions**

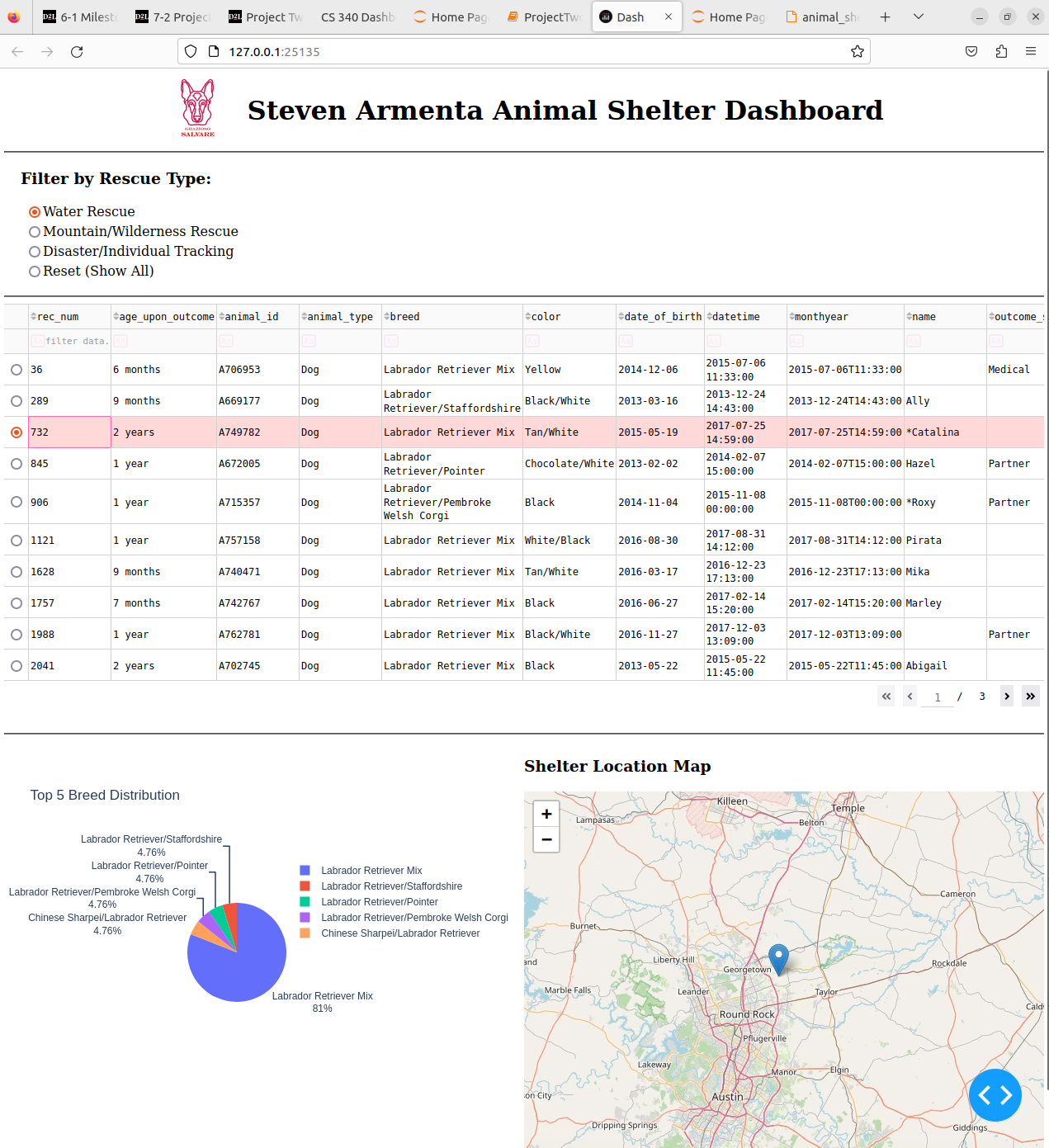
***Challenge 1:*** *The first challenge I experienced was being able to get the pie chart to work correctly. This was difficult because when I first implemented the chart percents kept showing up on the pie chart, so it looked not user friendly. The solution was to update limit the chart to display only the 5 most popular dog breeds improving performance and readability.*

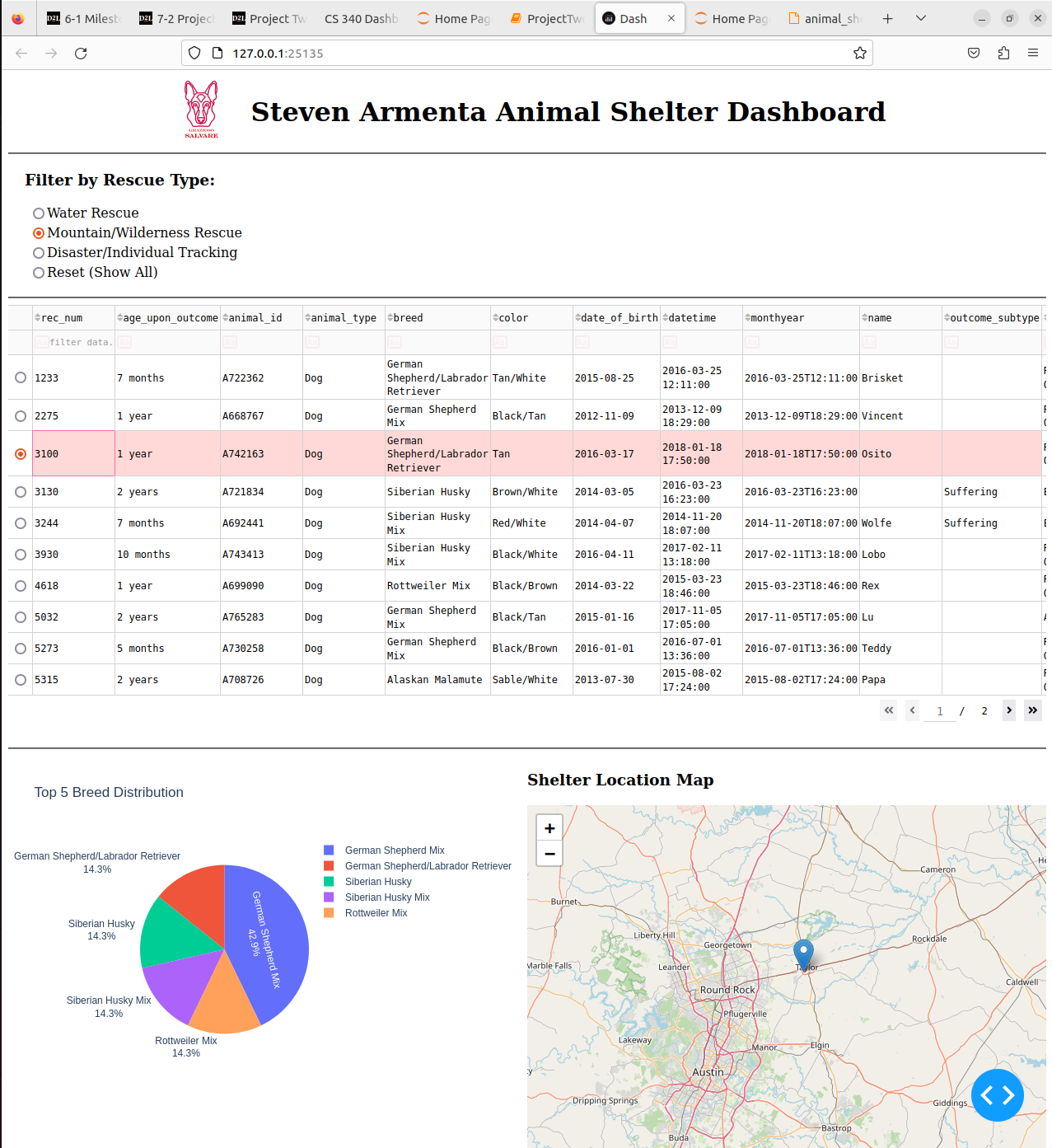
***Challenge 2****: Syntax Errors and getting the Map to Dynamically Update was a challenge I encountered frequently. The solution to the syntax errors were to read over my code carefully and for the Dynamic Map update issue I implemented a callback function that properly handles the derived virtual data and selected rows, and includes fallback options when data is not available.*

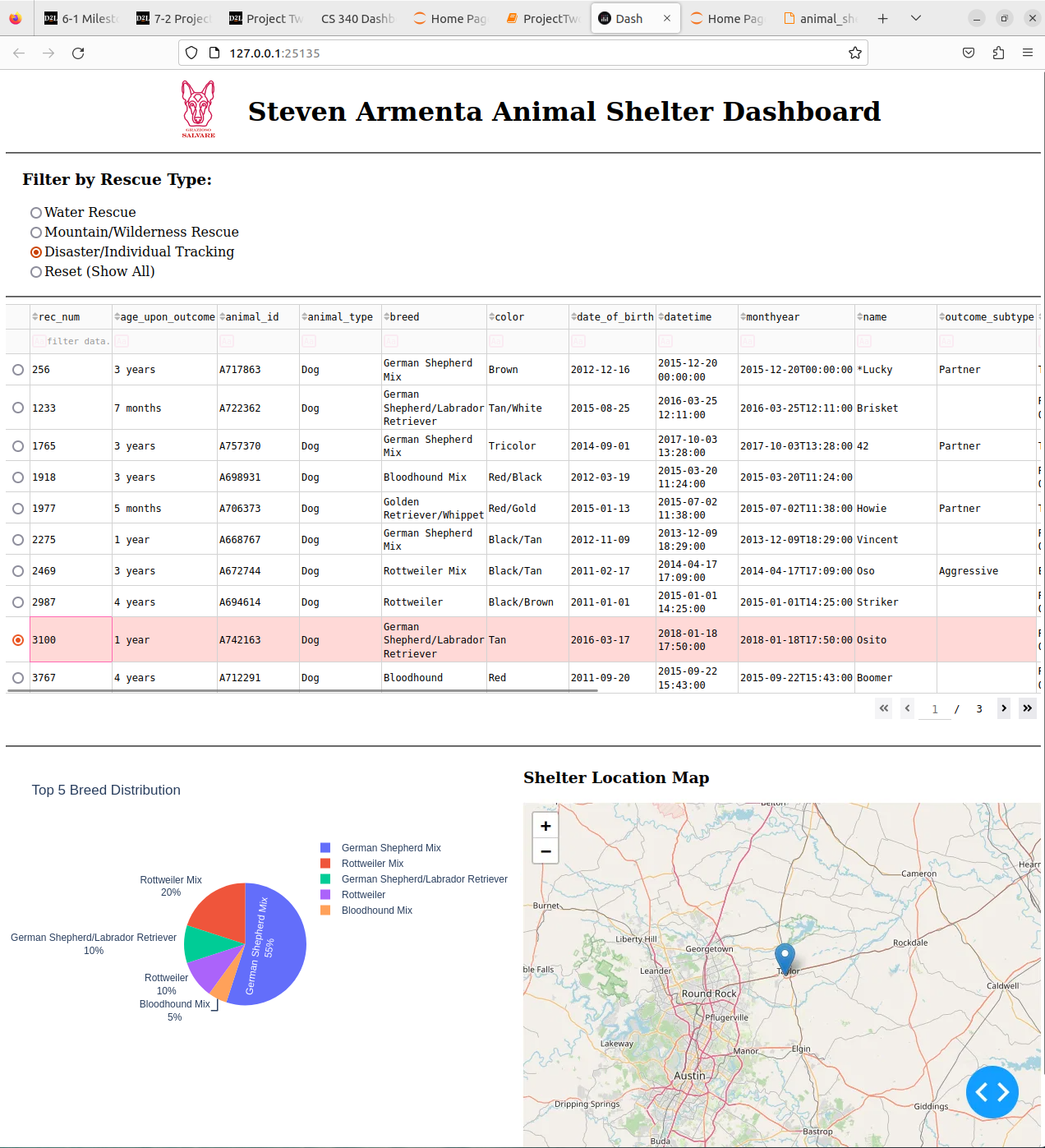
**Screenshots**

**Unfiltered Dashboard View**

**

**Water Rescue Filter Applied**

**Mountain/Wilderness Rescue Filter Applied**

**Disaster/Individual Tracking Filter Applied**

**Contact**

Your name: Steven Armenta