**CS 340 Project Two: Grazioso Salvare Dashboard**

## About the Project

*This project is a* ***Dash-based web application*** *that enables users to* ***interactively visualize and manage animal shelter data*** *stored in a* ***MongoDB database****. The dashboard provides an* ***intuitive interface*** *for filtering, searching, and analyzing rescue dog data for* ***Grazioso Salvare****, an animal rescue training company.*

*The application includes the following functionalities:*

* ***Interactive Data Table:*** *Displays animal shelter records with sorting, filtering, and row selection.*
* ***Geolocation Map:*** *Shows the location of selected rescue dogs on an interactive Leaflet map.*
* ***Breed Distribution Chart:*** *Visualizes breed proportions using a dynamic pie chart.*
* ***Rescue Type Filters:*** *Allows users to filter animals based on their suitability for:*
  + *Water Rescue*
  + *Mountain/Wilderness Rescue*
  + *Disaster or Individual Tracking*
  + *Reset (Show All)*

## Motivation

*Animal shelters manage* ***large datasets*** *related to* ***adoption status****,* ***medical history****, and* ***rescue suitability****. To streamline data analysis, this project integrates* ***MongoDB****,* ***Dash****, and* ***Plotly*** *to create a* ***scalable****,* ***user-friendly dashboard*** *for efficient* ***data-driven decision-making****.*

## Getting Started

*To get a local copy up and running, follow these steps:*

1. ***Install******MongoDB*** *and ensure the database server is running.*
2. ***Set up a MongoDB user*** *(****aacuser****) with authentication.*
3. *Install the required Python libraries:*

* pip install pymongo dash plotly dash-leaflet pandas numpy

1. *Run the dashboard.*

* python dashboard.py # Or use Jupyter Notebook for the JupyterDash version

1. *Open* ***http://127.0.0.1:27701/*** *in a web browser to access the dashboard.*

## Installation Requirements

*Ensure you have the following tools installed:*

1. *Python 3.x*
2. *MongoDB*
3. *Required Python libraries*
4. *Jupyter Notebook*
5. *A Web Browser (for viewing the interactive dashboard)*

## Usage

*This module allows users to interact with an* ***animal shelter database*** *using a* ***Dash-powered web application****. Users can:*

* ***Filter*** *rescue animals by type (Water, Mountain, Disaster).*
* ***View and search*** *animal records in an interactive* ***data table****.*
* ***Select an animal*** *to see its* ***location on a map****.*
* ***Analyze breed distribution*** *through a* ***dynamic pie chart****.*

**Dashboard Features**

*The web application provides a* ***graphical interface*** *for interacting with the database using the following components:*

1. ***Filter by Rescue Type***

* *Users can* ***select a rescue type*** *(Water, Mountain, Disaster) using radio buttons.*
* *The table updates dynamically to* ***show only relevant animals****.*

1. ***Interactive Data Table***

* *Displays* ***real-time data*** *from MongoDB.*
* *Supports* ***sorting****,* ***filtering****, and* ***pagination****.*
* *Clicking a row updates the* ***map*** *and* ***pie chart****.*

1. ***Geolocation Map***

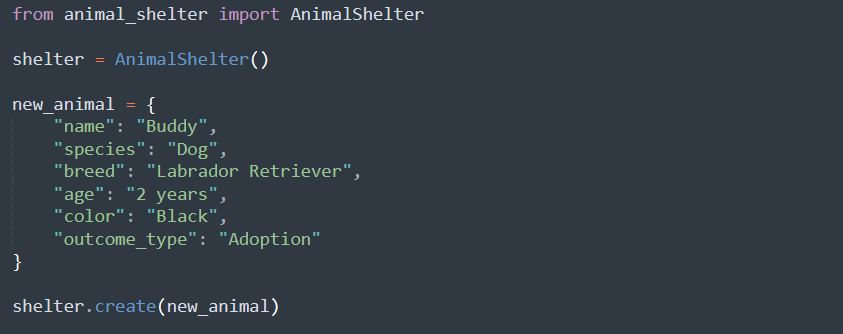
* *Shows the* ***selected animal’s location*** *on an interactive Leaflet map.*
* *Updates dynamically based on table selection.*

1. ***Breed Distribution Pie Chart***

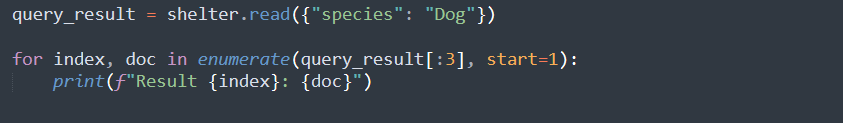
* *Displays a* ***breakdown of dog breeds*** *within the selected rescue type.*
* *Updates automatically when filtering data.*

### Code Example

***Create a new Animal record:***



***Read records from the database:***

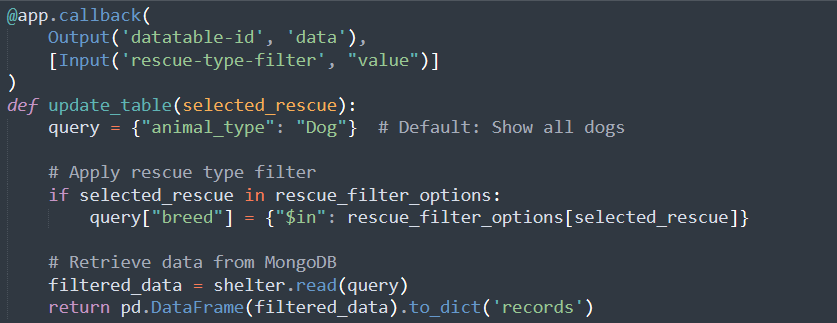
**

***Update an Animal record:  
***

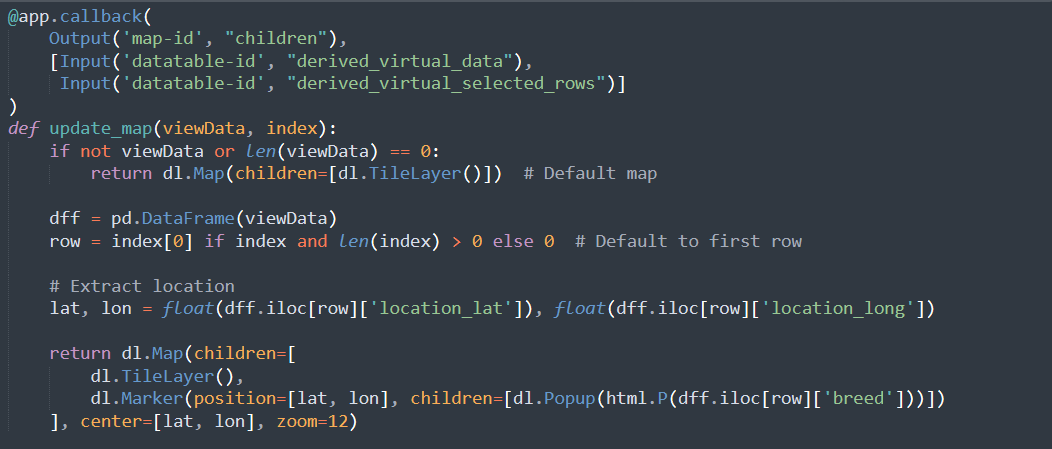
***Delete an Animal record:***



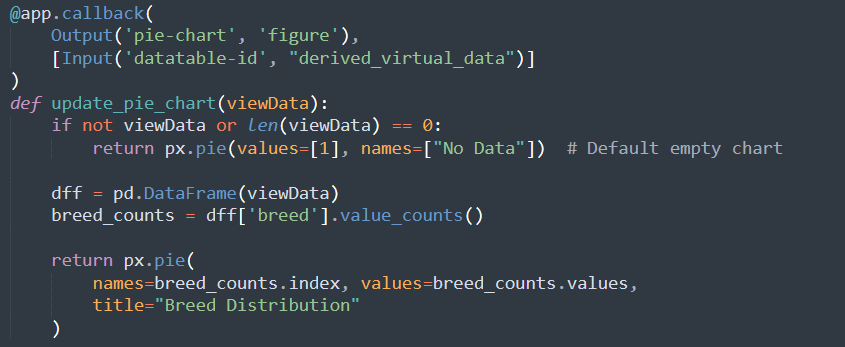
**Fetching Data from MongoDB Based on Filters:**



**Updating the Map Based on Selected Animal:**



**Generating a Pie Chart for Breed Distribution:**



**Development Tools and Rationale**

**This project was built using:**

* ***Python****: Chosen for its ease of scripting and* ***Dash*** *web framework integration.*
* ***Dash (Plotly):*** *Used to create the* ***interactive dashboard*** *for visualizing and filtering shelter data.*
* ***MongoDB****: A* ***NoSQL database*** *ideal for storing* ***flexible*** *and* ***dynamic*** *animal records.*
* ***PyMongo****: A Python library used to interact with MongoDB.*

**Why MongoDB?**

* ***Schema-less Structure****: Allows flexible data storage.*
* ***Indexing for Faster Queries****: The* ***breed*** *field is indexed to improve search performance.*
* ***Scalability****: NoSQL structure enables handling large datasets efficiently.*

**Dash Callbacks and MongoDB Queries Used**

Instead of manual CRUD operations, the project utilizes **Dash callbacks** that dynamically retrieve, filter, and visualize data.

* **Filtering Rescue Type Selection**

Uses **find()** queries in MongoDB to filter animals based on rescue type selection.

* **Displaying Animal Data in the Table**

Retrieves and displays data dynamically in a Dash **DataTable**.

* **Map and Pie Chart Updates**

Updates animal **location on the map** and **displays breed** distribution in a pie chart using Plotly.

**Testing the Application**

To verify that the **Dash application** functions correctly, follow these steps:

1. **Launch the Dashboard**

Run the Dash application and verify the MongoDB connection.

1. **Filter by Rescue Type**

Select different rescue types and confirm that the **table updates** dynamically.

1. **Select an Animal in the Table**

Verify that selecting an animal **updates the map location correctly**.

1. **Check Breed Distribution Pie Chart**

Confirm that the **pie chart updates** based on the selected data.

**Challenges and Solutions**

**1. Data Filtering & Querying**

**- Issue:** Initially, filters did not return expected results due to incorrect query syntax.

**- Solution:** Used `"$in": []` for breed filtering and verified breed names in MongoDB.

**2. Geo-Location Mapping**

**- Issue:** Some animals were not showing correctly on the map.

**- Solution:** Ensured `location\_lat` and `location\_long` were converted to numeric and handled missing data.

**3. Dash Layout Adjustments**

**- Issue:** The pie chart and map were not aligned correctly.

**- Solution:** Used `display: flex` and column layout to align them horizontally.

**References**

**MongoDB Documentation:** https://www.mongodb.com/docs/

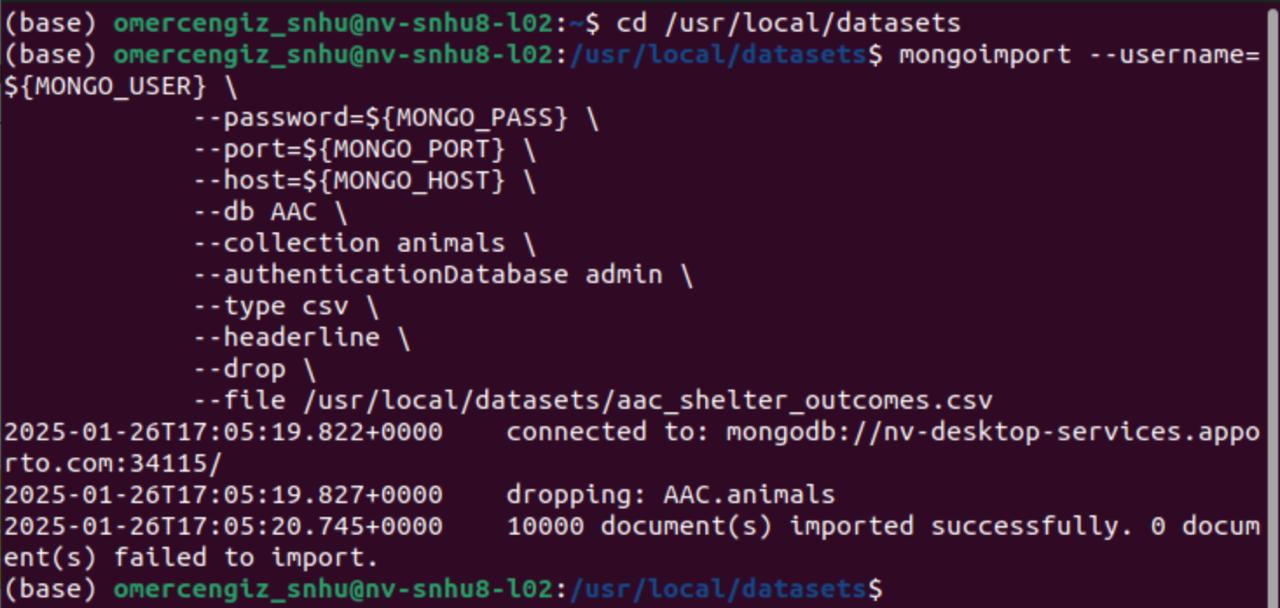
**Dash by Plotly:** https://dash.plotly.com/

## Screenshots

## *Data Import and Indexing*

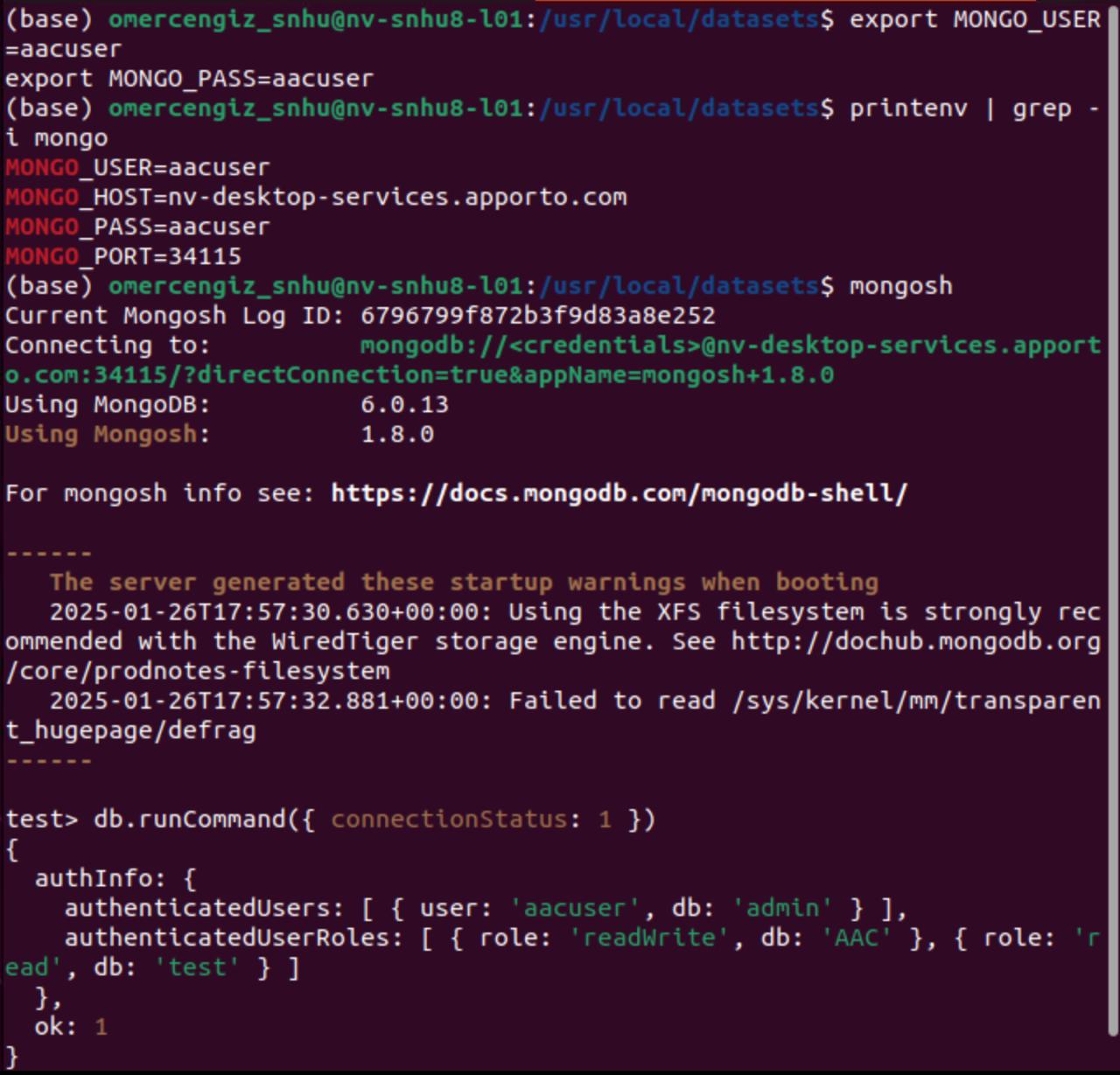
## *Shows how data is loaded into MongoDB.*

## *Demonstrates index creation.*



## *User Authentication*

## *Confirms aacuser login with proper credentials.*

****

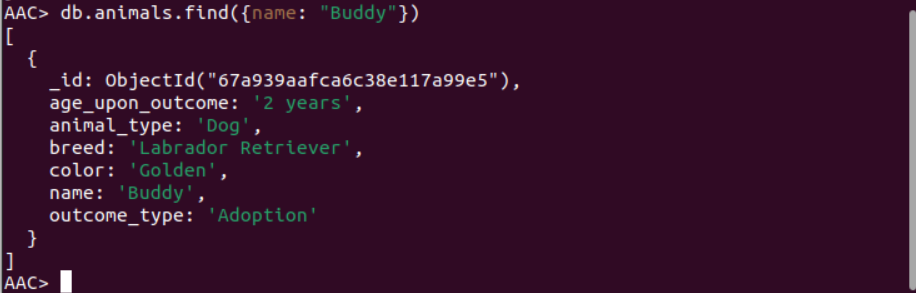
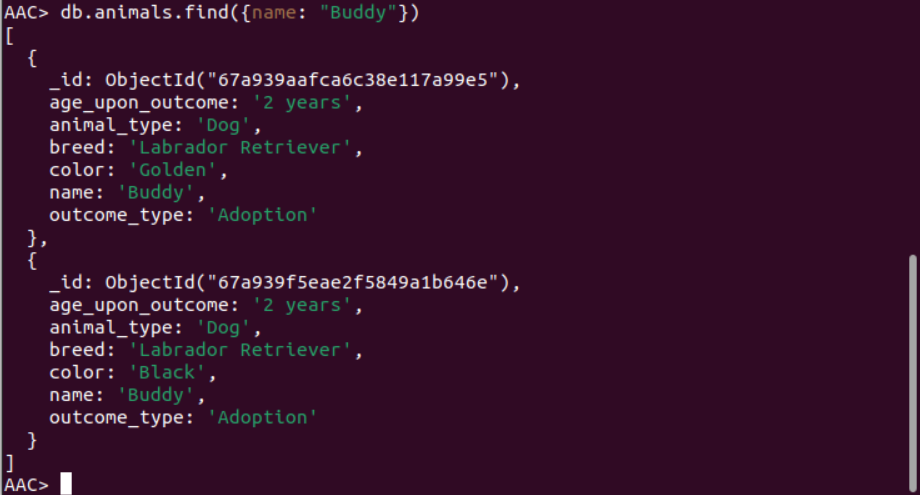
****

## *CRUD Operations*

## *Screenshots for Create, Read, Update, Delete operations.*

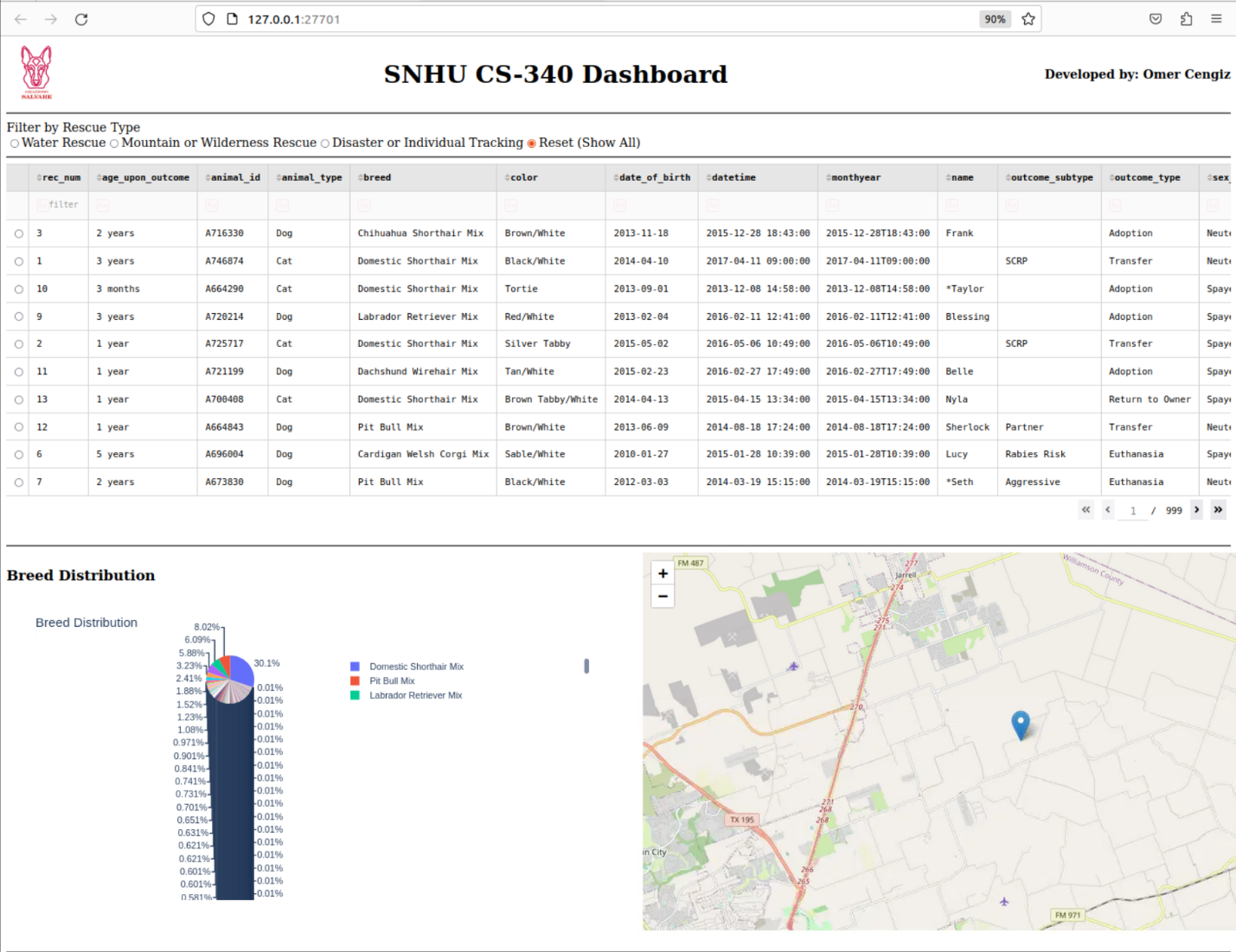
## *Shows before and after results for update() and delete().*

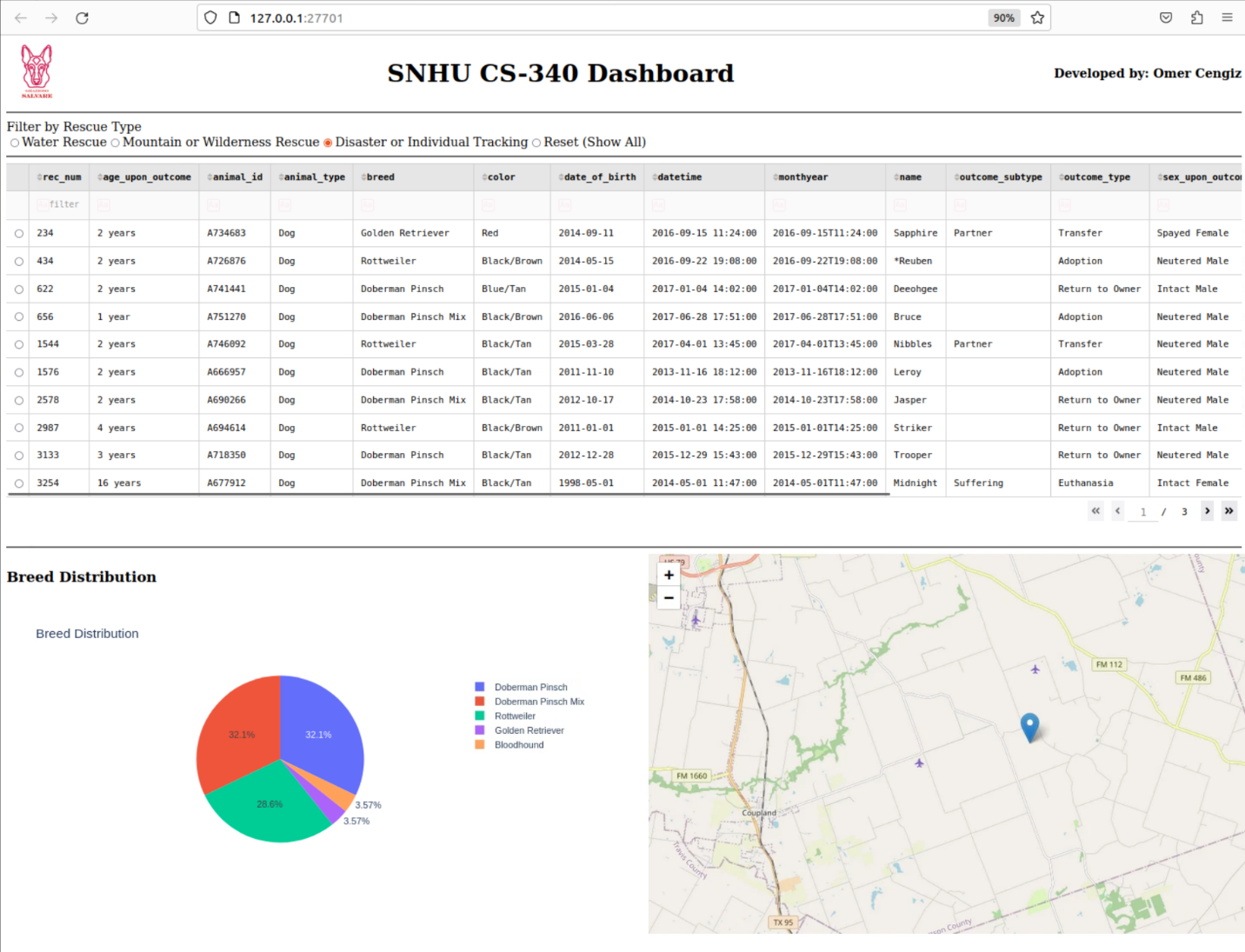
****

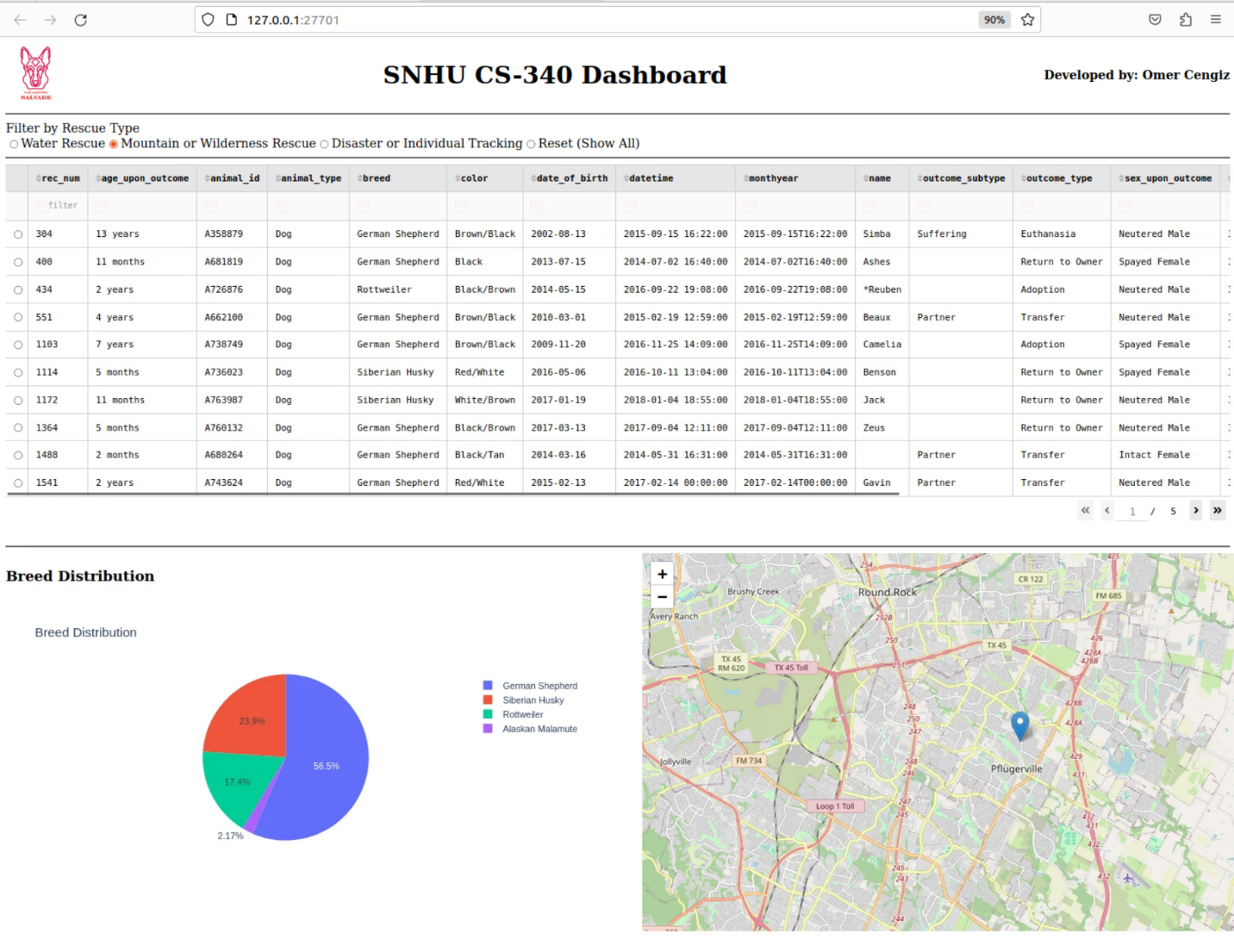


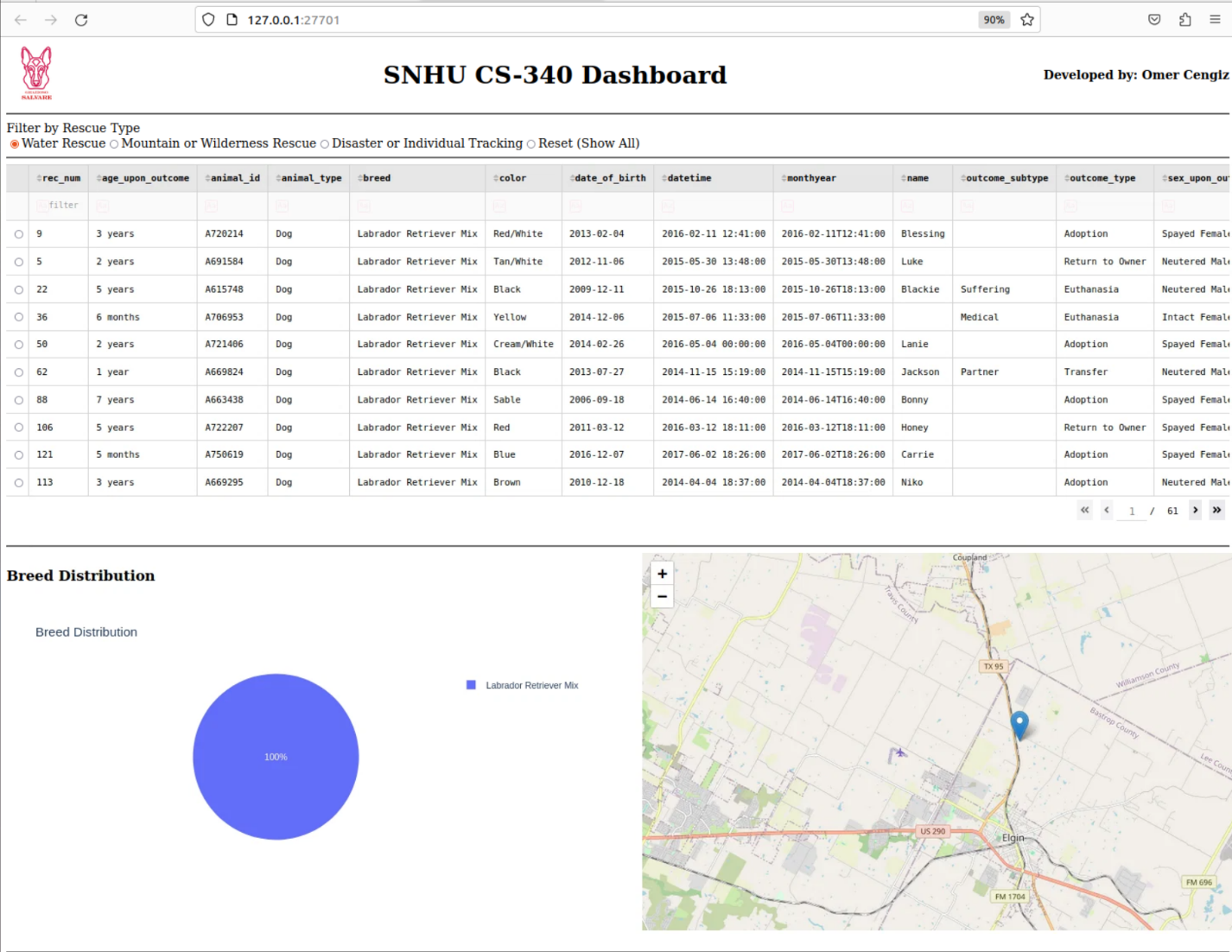
1. **Testing the Application**

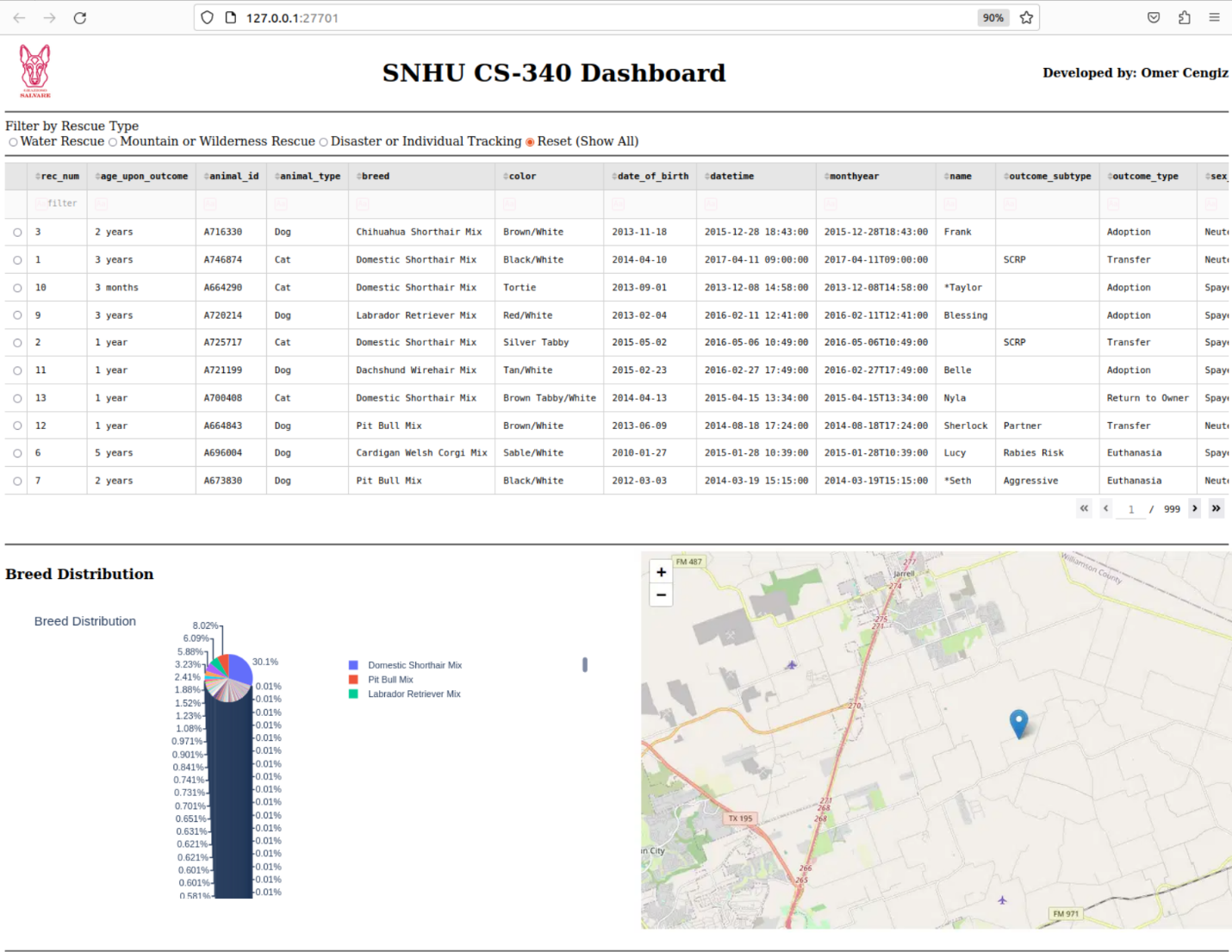
**Dashboard in its default state (unfiltered)**

****

**Dashboard with Water Rescue filter applied**

**Dashboard with Mountain Rescue filter applied**

**Dashboard with Disaster Rescue filter applied**

**Dashboard after Reset (showing all data)**

**RESTful API and MVC Consideration**

* *The* ***RESTful API approach*** *allows seamless* ***CRUD operations via HTTP requests****.*
* *The project follows the* ***Model-View-Controller (MVC)*** *pattern, where:*
  + ***Model****: MongoDB database structure.*
  + ***Controller****: Python CRUD module.*
  + ***View****: (To be implemented in Project Two) will be a Dash-based dashboard.*

## Contact

Author: Omer Cengiz