# CS 340 README Template

**Grazioso Salvare Rescue Training**

This project implements a full-stack dashboard application for Grazioso Salvare, a company specializing in training rescue dogs. The application enables staff to identify candidate animals from the Austin Animal Center Outcomes dataset using interactive tools.

The project includes a Python CRUD module using PyMongo to interact with a MongoDB database, and a web dashboard built with Dash and Dash Leaflet that visualizes and filters animal records. Features include interactive filtering, geolocation mapping, and breed distribution charts. The goal is to provide an intuitive, user-friendly interface for data exploration that minimizes the need for manual queries or raw data analysis.

## Motivation

This solution was developed to meet Grazioso Salvare’s need for identifying adoptable animals that meet criteria for search-and-rescue training, automating and simplifying access to shelter outcome data, and enabling staff without technical expertise to explore and filter large volumes of data efficiently.

## Getting Started

To reproduce this project, you must have access to a MongoDB database populated with Austin Animal Center animal outcomes data. Create a user account in MongoDB (e.g., aacuser) with appropriate read/write permissions. Download the project files, including animal\_shelter.py (the CRUD module) and ProjectTwoDashboard.ipynb (the main dashboard). Open the notebook in JupyterLab or Jupyter Notebook and run it. The dashboard will launch inside the notebook interface using JupyterDash.

## Installation

This project requires Python 3.x, MongoDB, PyMongo, Dash, JupyterDash, and Dash Leaflet. Python can be downloaded from [python.org,](https://www.python.org/) and MongoDB from [mongodb.com](https://www.mongodb.com/). The remaining libraries can be installed using pip.

## Usage

The application follows the Model-View-Controller (MVC) architecture. The model is represented by the animal\_shelter.py module, which contains CRUD functions to interact with the MongoDB database using PyMongo. The view and controller are provided by Dash. Users interact with radio buttons, data tables, and charts in the interface, while Dash callbacks handle the controller logic, updating the visuals based on user input. The application includes predefined MongoDB queries that filter animals based on their suitability for different rescue types, using attributes like breed, age, and sex.

### Code Example

Python CRUD File

A screenshot of a computer code

AI-generated content may be incorrect.

## 

## Required Functionality

The dashboard enables users to view all animal data in a paginated, sortable table and to filter by Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking, or Reset (to show all records). A pie chart dynamically reflects the breed distribution based on the filtered data, and a Leaflet-based map displays the location of the selected animal from the table. (Screenshots below)

## Challenges and Resolutions

One common challenge was ensuring the logo image appeared correctly. This was solved by using base64 encoding for inline rendering in JupyterDash. Another challenge was the pie chart becoming cluttered when displaying all breeds; this was resolved by adding logic to emphasize the top five breeds while aggregating rare ones. Errors with layout and component nesting were fixed by wrapping chart outputs properly in dcc.Graph. To prevent map or table crashes when no row is selected, additional error handling was added to ensure safe defaults.

## Tools and Rationale

Python 3.x was selected for its ease of use and wide support for data visualization and web development libraries. MongoDB was used for its flexibility in handling unstructured animal data and for its native compatibility with Python through the PyMongo driver. Dash was used to create the front-end layout and controller logic using pure Python, making it accessible to data scientists and developers alike. Dash Leaflet provided geolocation functionality, and JupyterDash allowed for running the app directly inside a notebook during development and demonstration.

## Links

* Mongo DB <https://www.mongodb.com/docs/>
* PyMongo Documentation<https://pymongo.readthedocs.io/>
* Dash by Plotly <https://dash.plotly.com/>
* Dash Leaflet <https://dash-leaflet.herokuapp.com/>
* Python <https://www.python.org/>

### Screenshots

*Testing File*A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

*MongoDB Import*

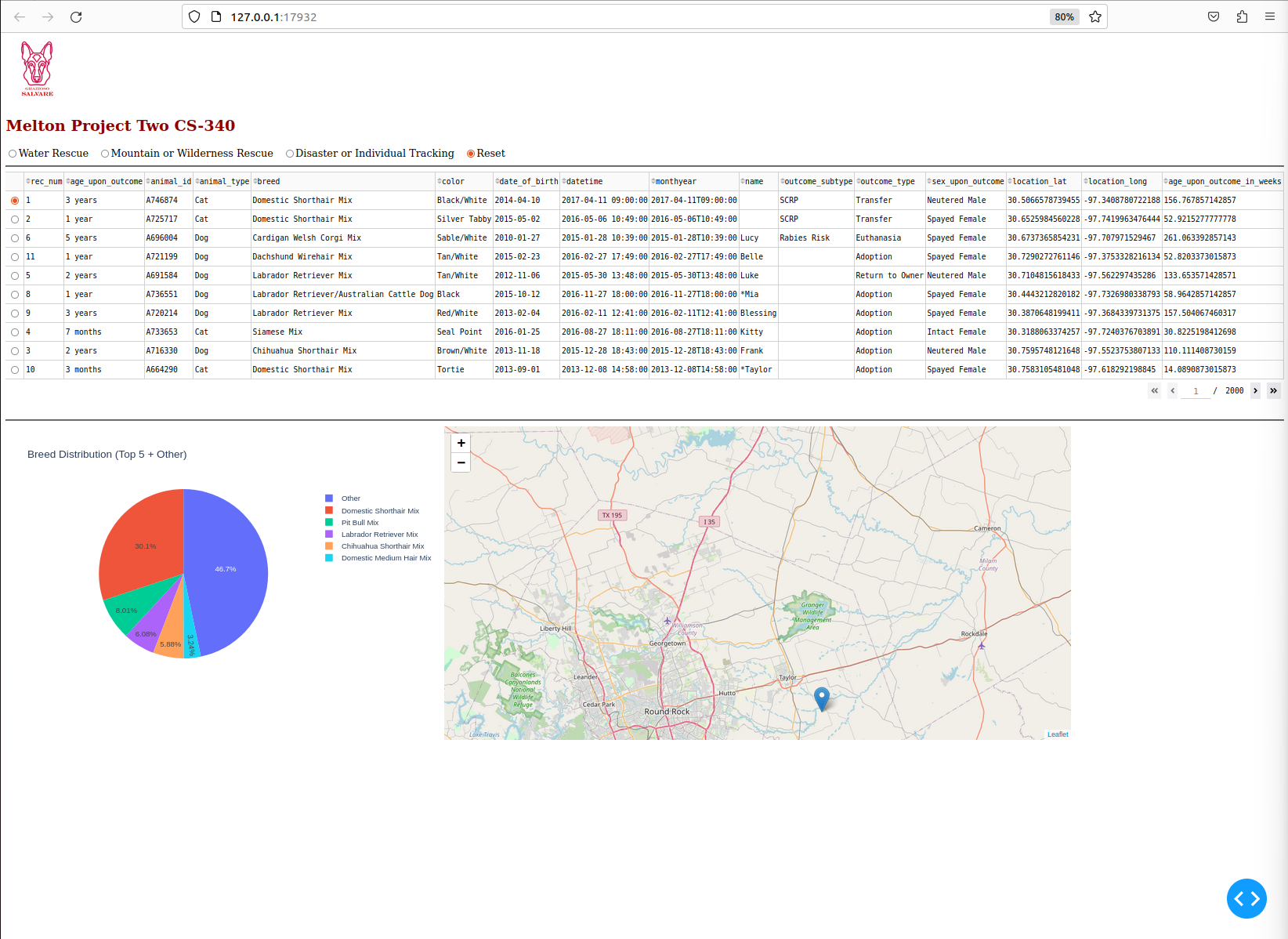
A screenshot of a computer screen

AI-generated content may be incorrect.

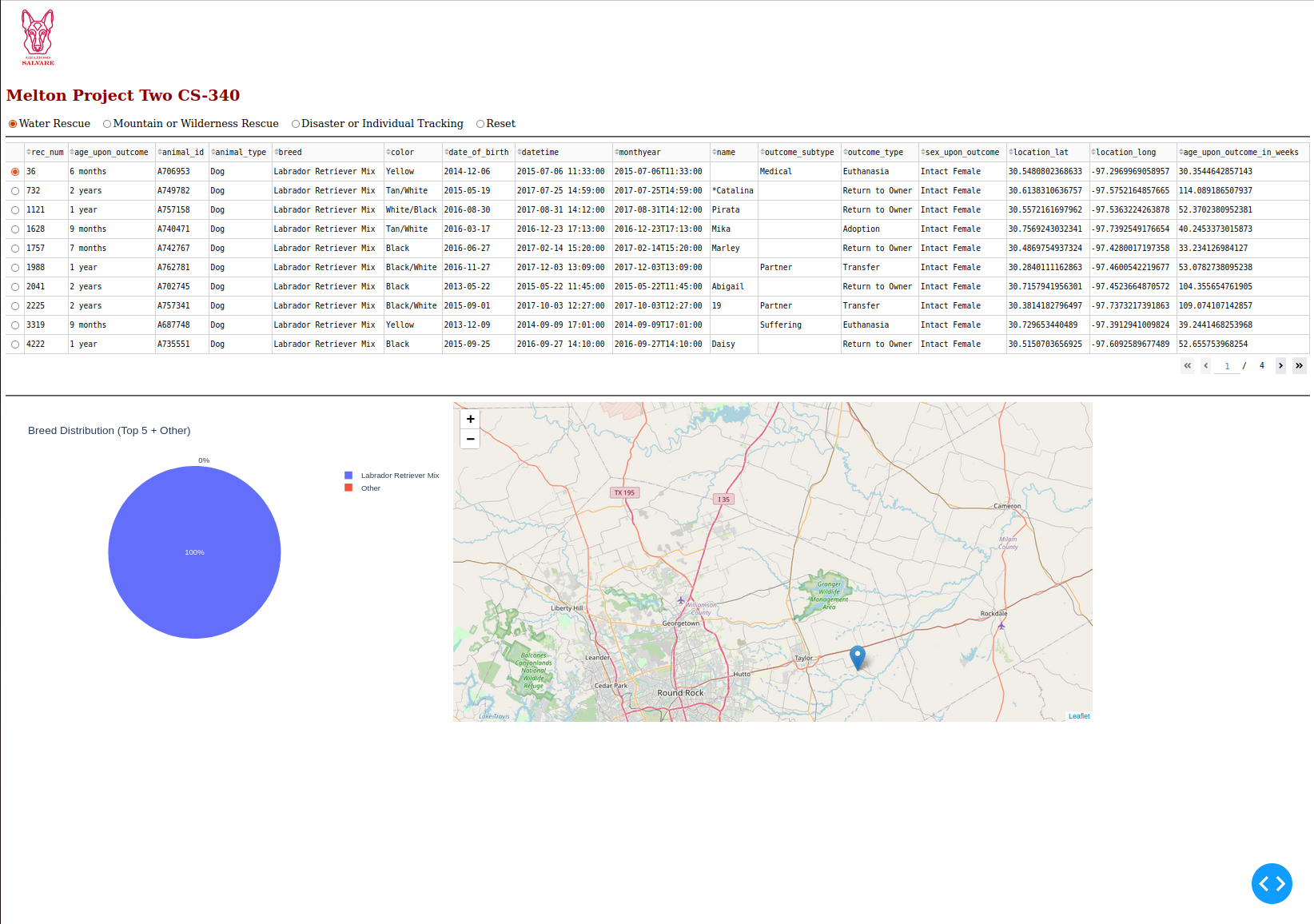
*MongoDB Authentication*A screenshot of a computer program

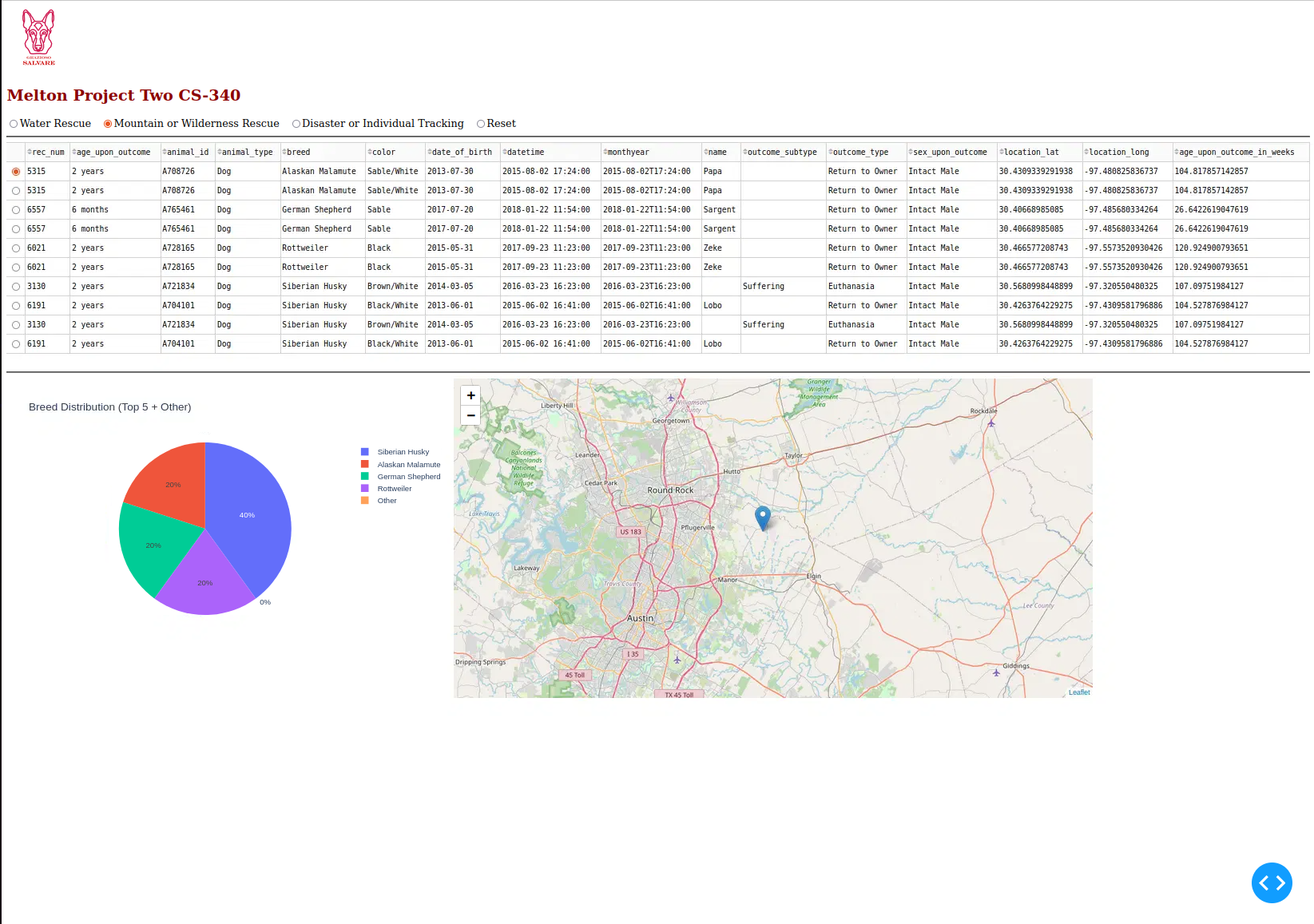
AI-generated content may be incorrect.

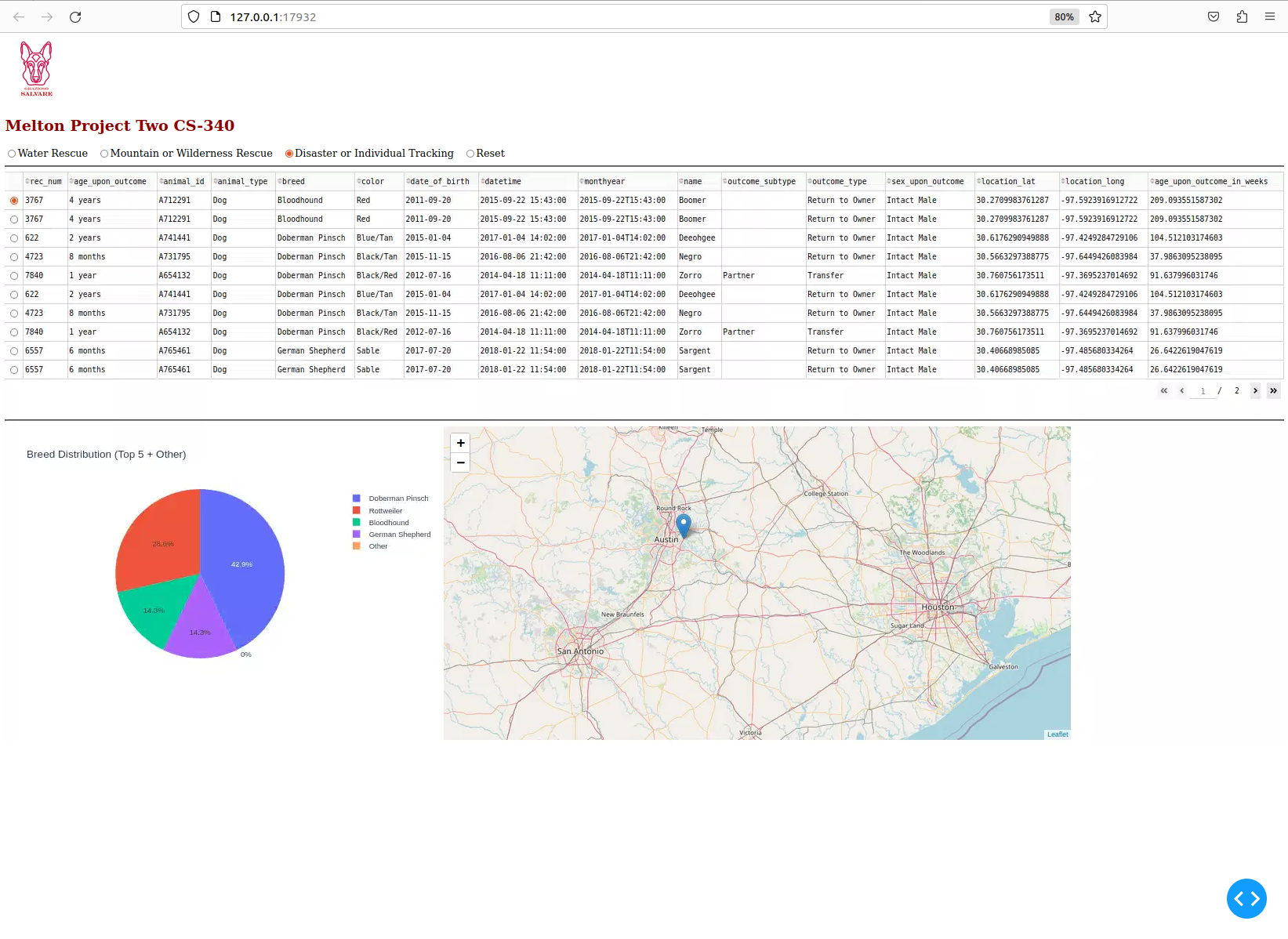
*Dashboard Default View*

**

*Dashboard Water View*

**

*Dashboard Mountain Rescue View*

*Dashboard Disaster View*

## Contact

Your name: Dexter Melton