# AAC DATASET DASHBOARD README

## About the Project/Project Title

This is a Python dashboard built using Dash and MongoDB for Grazioso Salvare, a company that trains rescue dogs. The dashboard helps the client filter and view dogs from animal shelters in the Austin, TX area.

The dashboard was built in Jupyter Notebook and connects to a MongoDB database using a custom Python module (animal\_shelter.py) that supports basic CRUD operations.

## Features

Filters to select Rescue Type and Outcome Type

Reset button to clear filters

Data table showing matching animal records

Pie chart showing breed distribution

Geolocation map showing selected animal's location

## Motivation

This project provides an object-oriented interface for interacting with AAC shelter data. It demonstrates how a Python module can interface with a MongoDB NoSQL database.

## Files Included

ProjectTwoDashboard.ipynb: The main dashboard code

animal\_shelter.py: The Python CRUD class used to access MongoDB

## Getting Started

Follow these steps to get the project running on your Apporto virtual machine.

Open your Apporto environment and launch a terminal.

Import the dataset into MongoDB if it’s not already there.

Log in with:

Username: aacuser

Password: CSPassword123

Open Jupyter Notebook.

Navigate to and open ProjectTwoDashboard.ipynb.

Run each cell in the notebook to launch the dashboard.

## Installation

MongoDB: Used to store and manage animal shelter records.

Jupyter: Used for development and testing of the dashboard.

Dash / JupyterDash: Frameworks used to create the interactive dashboard.

VS Code (Optional): Recommended for editing .py files like animal\_shelter.py.

## Tools and Rationale

MongoDB: Used as the model because it's flexible and stores data in JSON-like documents, which is perfect for the varied structure of animal records. MongoDB also integrates well with Python via the pymongo library.

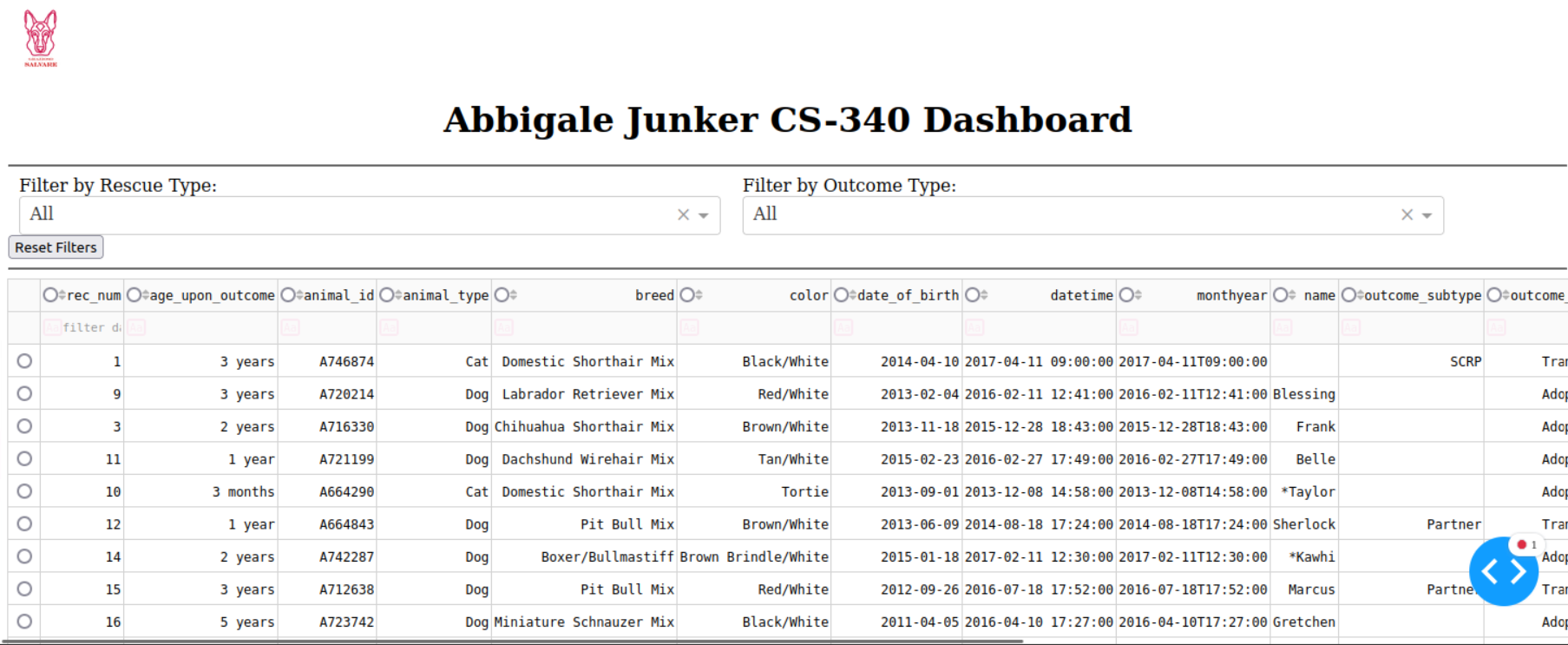
Dash/JupyterDash: Used to create the dashboard because it allows Python-based development of web apps. It's good for rapid development and connects easily with Plotly for charts and visualizations.

Pandas: Used to handle and display the MongoDB data in a readable format.

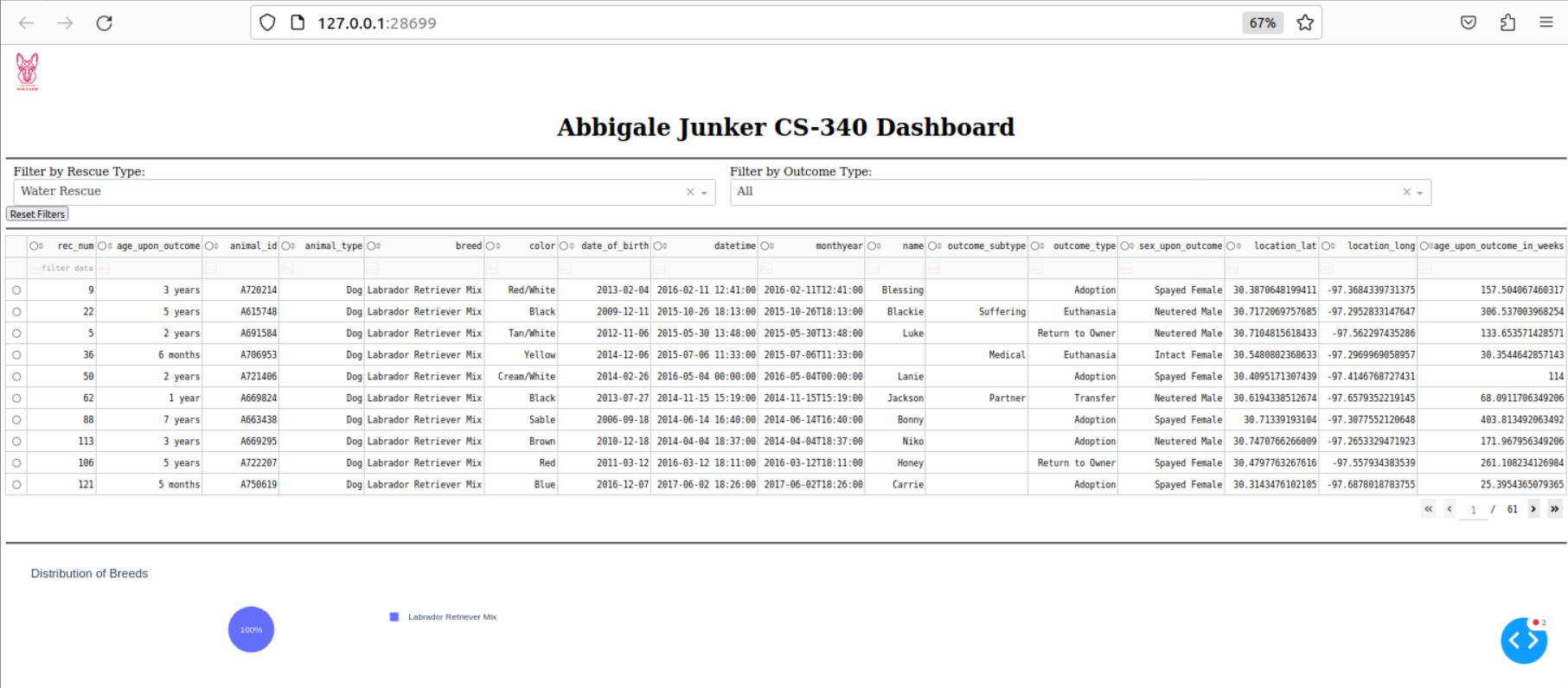
dash\_leaflet: Used to create the geolocation map.

### Dashboard Executions

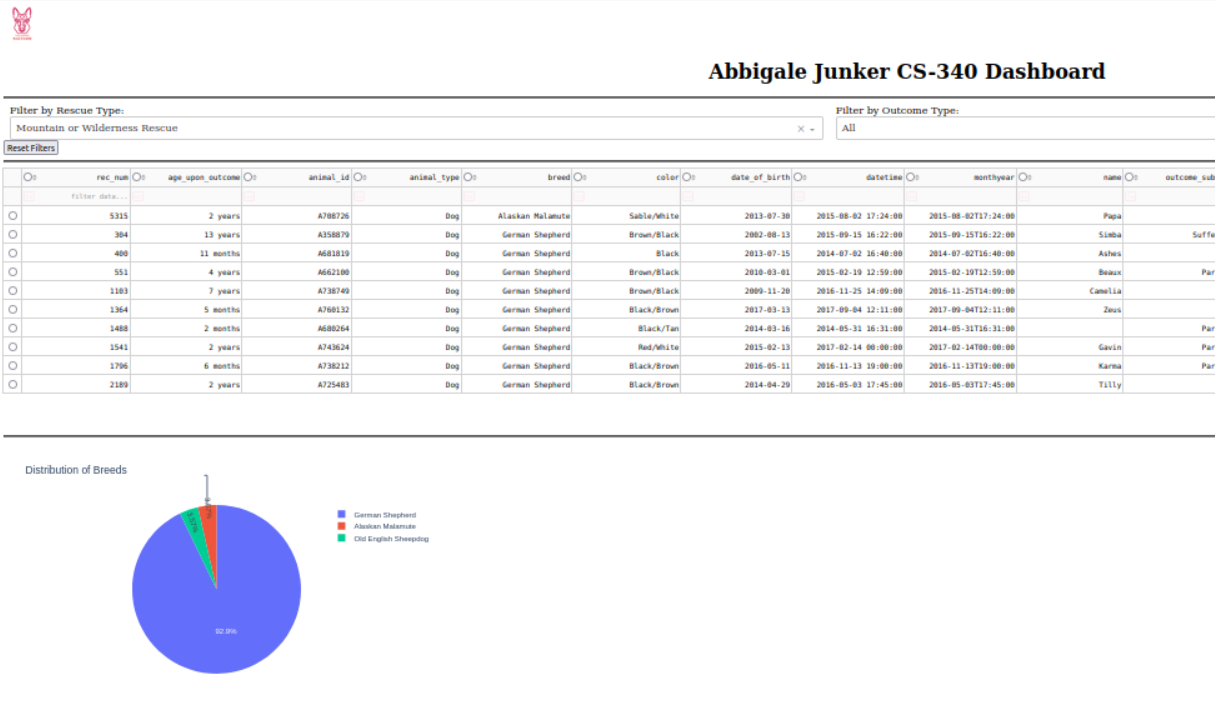
Unfiltered View



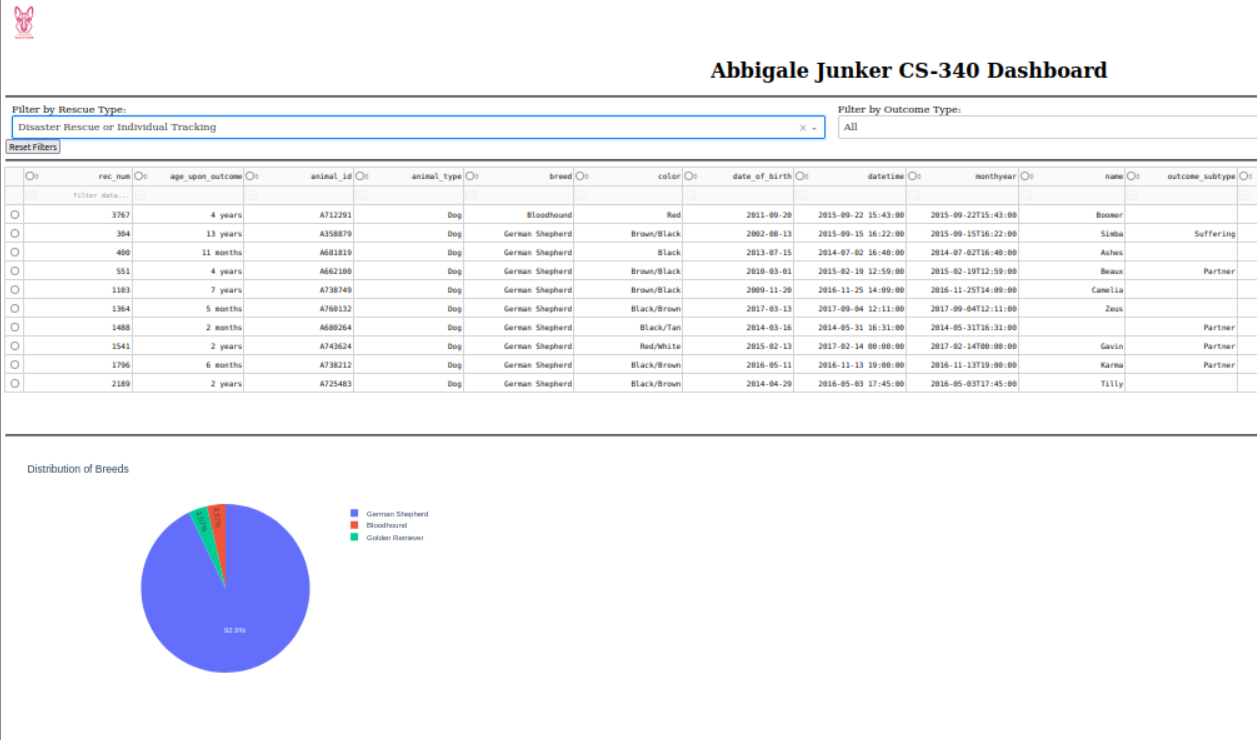
Water Rescue Type Filtering



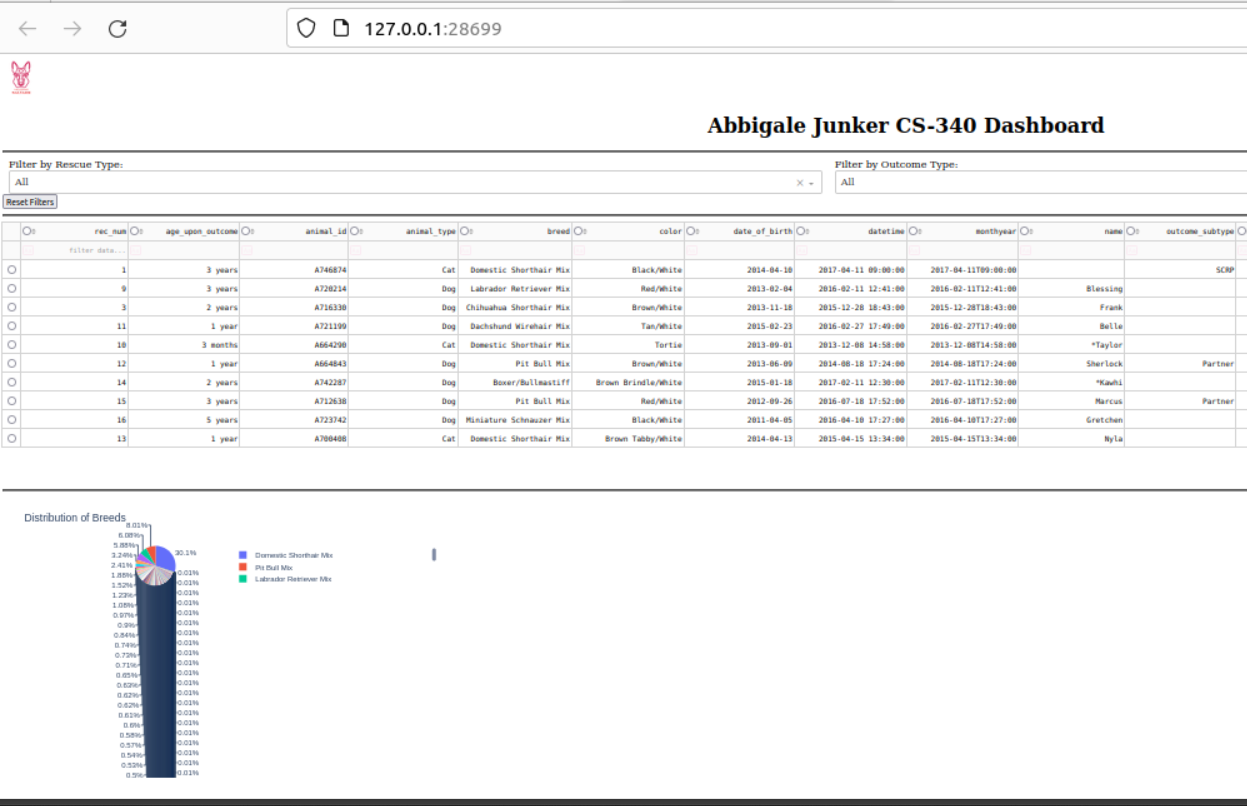
Mountain or Wilderness Rescue Type Filtering



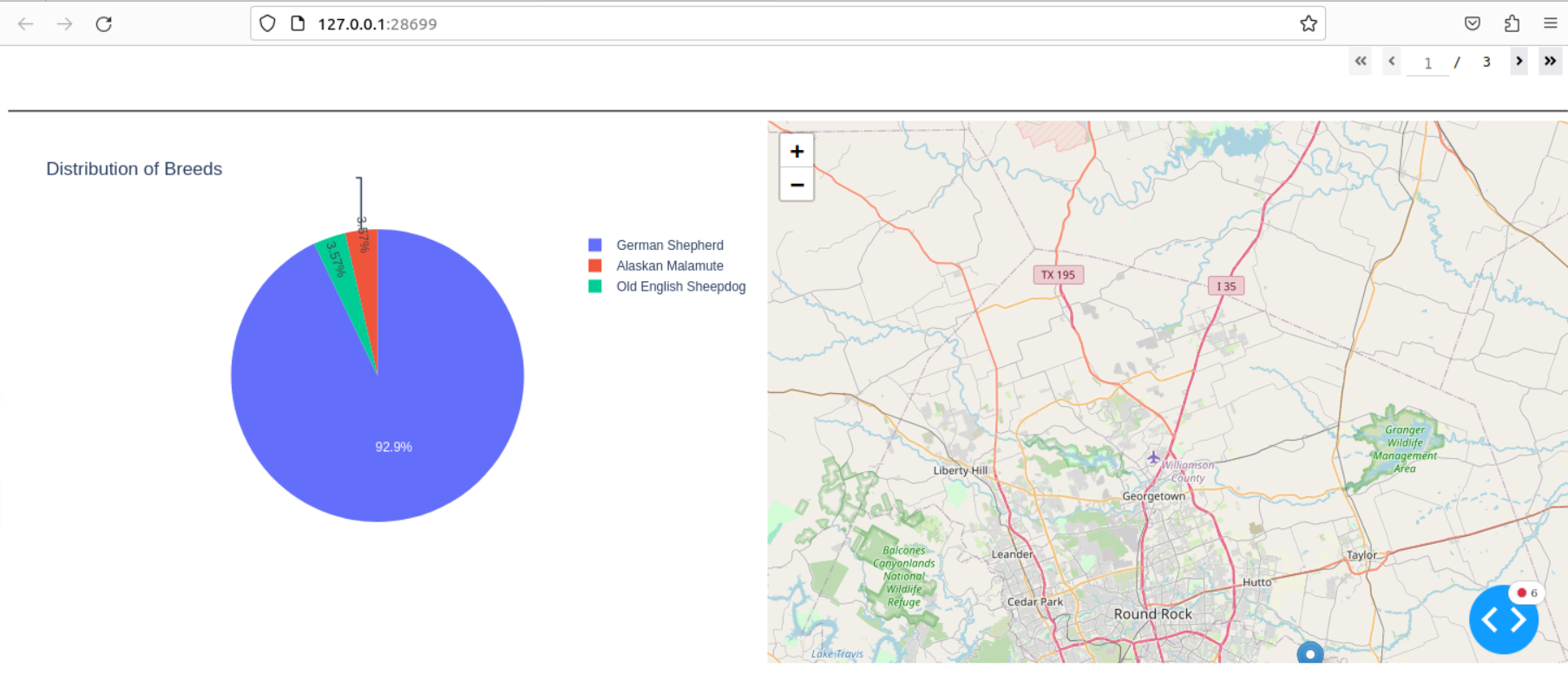
Disaster or Individual Tracking Type Filtering



Reset Button Clicked



Pie Chart and Geolocation Widget Example



### Challenges and Solutions

* \_id field crashing DataTable: MongoDB returns a special ObjectID in the \_id field. This was removed from the DataFrame to prevent crashes.
* Dropdowns freezing after Reset: Fixing the callback structure to separate filtering and resetting solved this.
* Data filtering not working: Required adjusting the filter logic and double-checking how the breed mapping was implemented for each rescue type. Separated reset filter callback from the dropdown filtering callbacks

## Contact

Your name: Abbigale Junker