# CS 340 Project Two README

## About the Project

This dashboard was developed for Grazioso Salvare to assist in identifying shelter dogs suitable for specialized rescue training. It interfaces with a MongoDB database containing animal data from the Austin Animal Center and allows users to filter animals based on rescue type.

The dashboard includes the following required features:

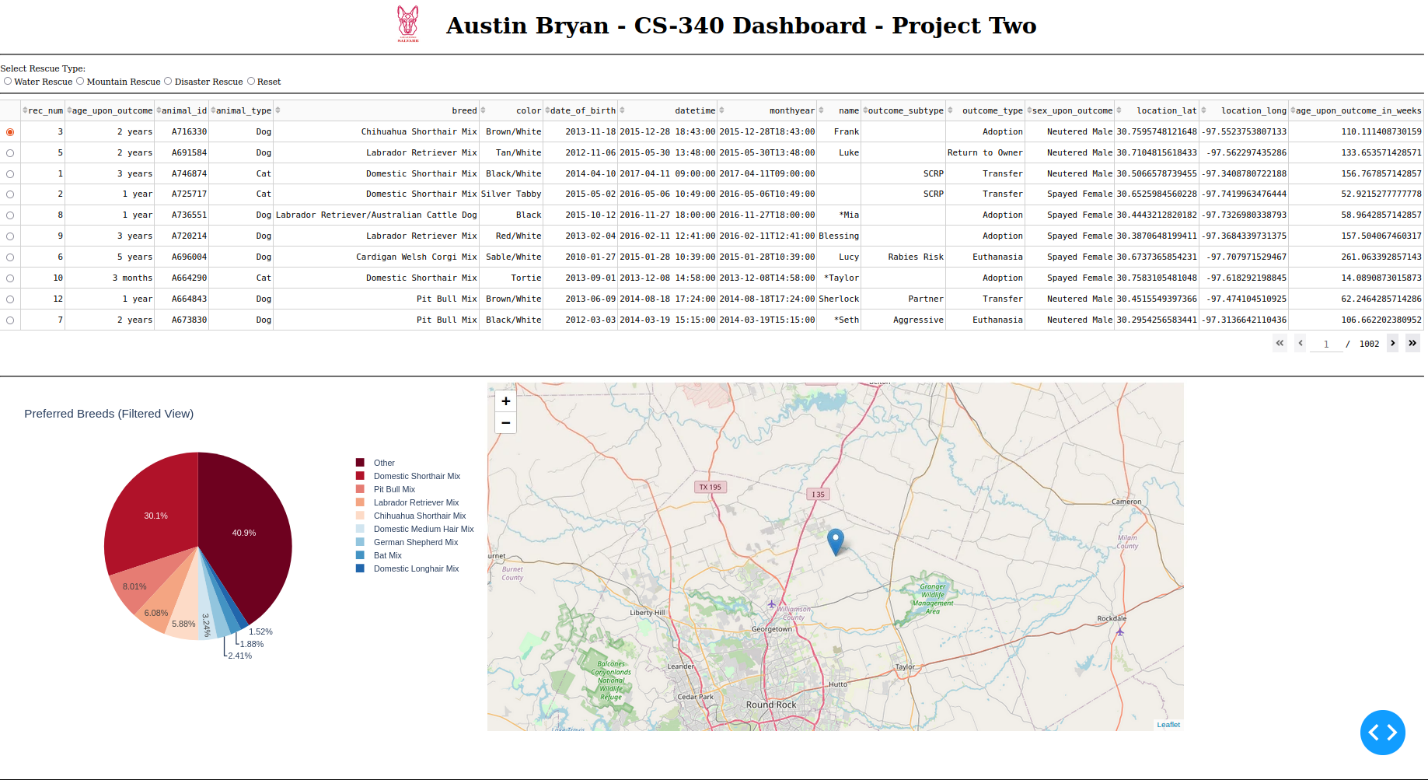
* **Rescue-Type Filtering:** A set of radio buttons lets users filter dogs by Water Rescue, Mountain Rescue, Disaster Rescue, or view all results (Reset). Each filter corresponds to specific breed, age, and sex criteria provided by the client.
* **Interactive Data Table:** The table updates dynamically based on the selected filter. It supports pagination and sorting to improve usability.
* **Geolocation Chart:** A map displays the location of the selected animal using latitude and longitude from the dataset.
* **Breed Distribution Chart:** A pie chart shows the breed distribution of filtered animals, aggregating smaller categories under "Other" for clarity.
* **Branding and Identification:** The dashboard includes the Grazioso Salvare logo (linked to SNHU) and a unique identifier with the developer’s name.

These features fulfill all functional requirements outlined in the project brief and the Dashboard Specifications Document.

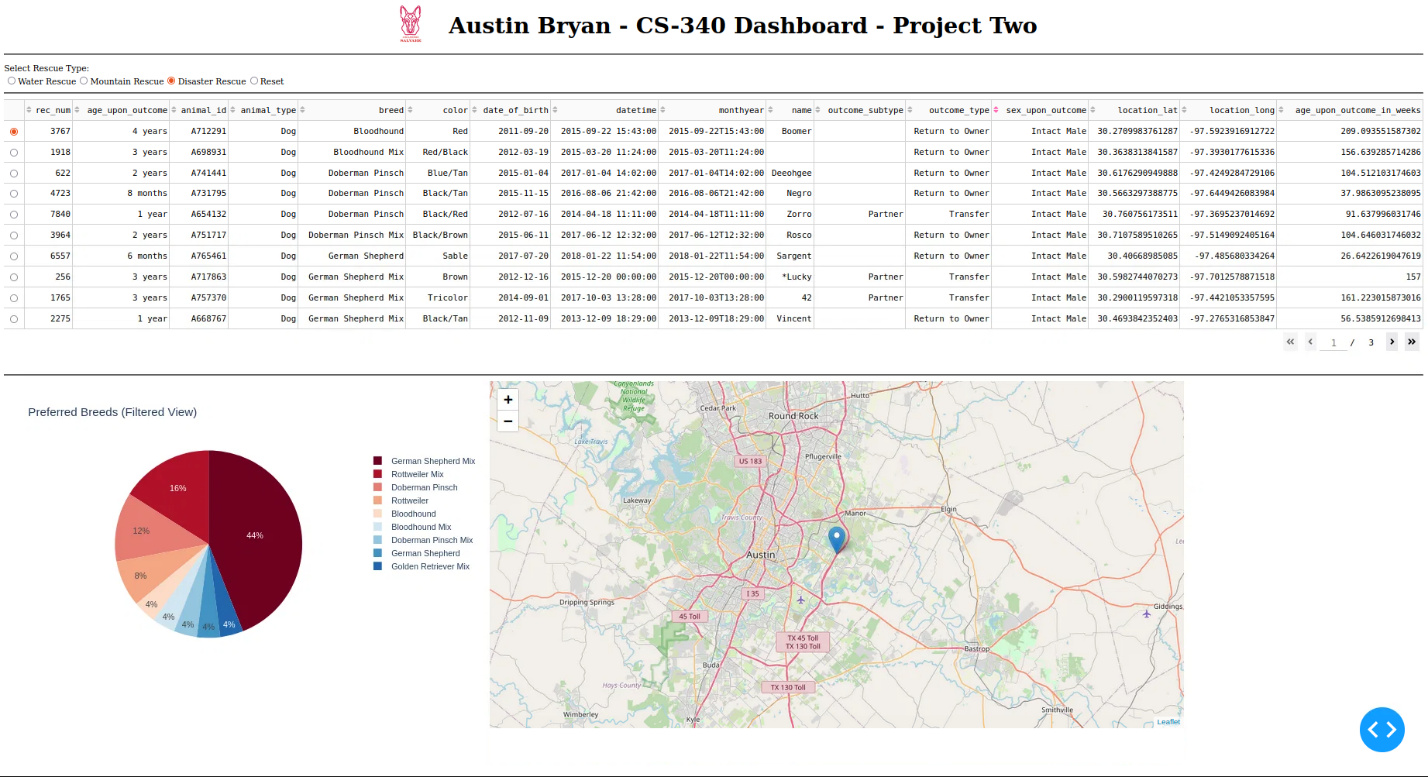
**Screenshots Demonstrating Functionality**

Screenshot 1: Initial dashboard state

*The dashboard opens with no filter selected, displaying all dogs in the database.*

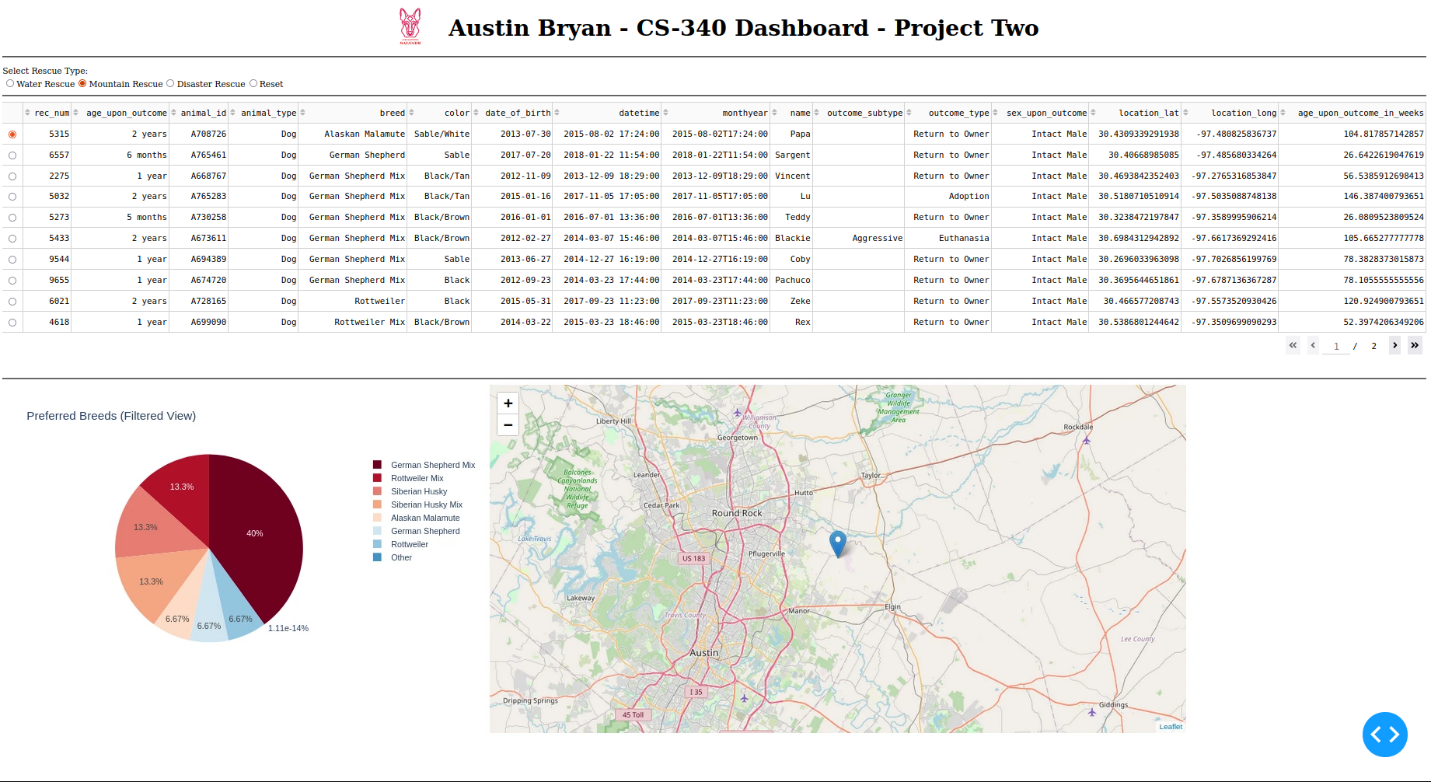


Screenshot 2: Disaster Rescue filter applied

*Displays intact male Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, and Rottweiler breeds aged between 20 and 300 weeks.*

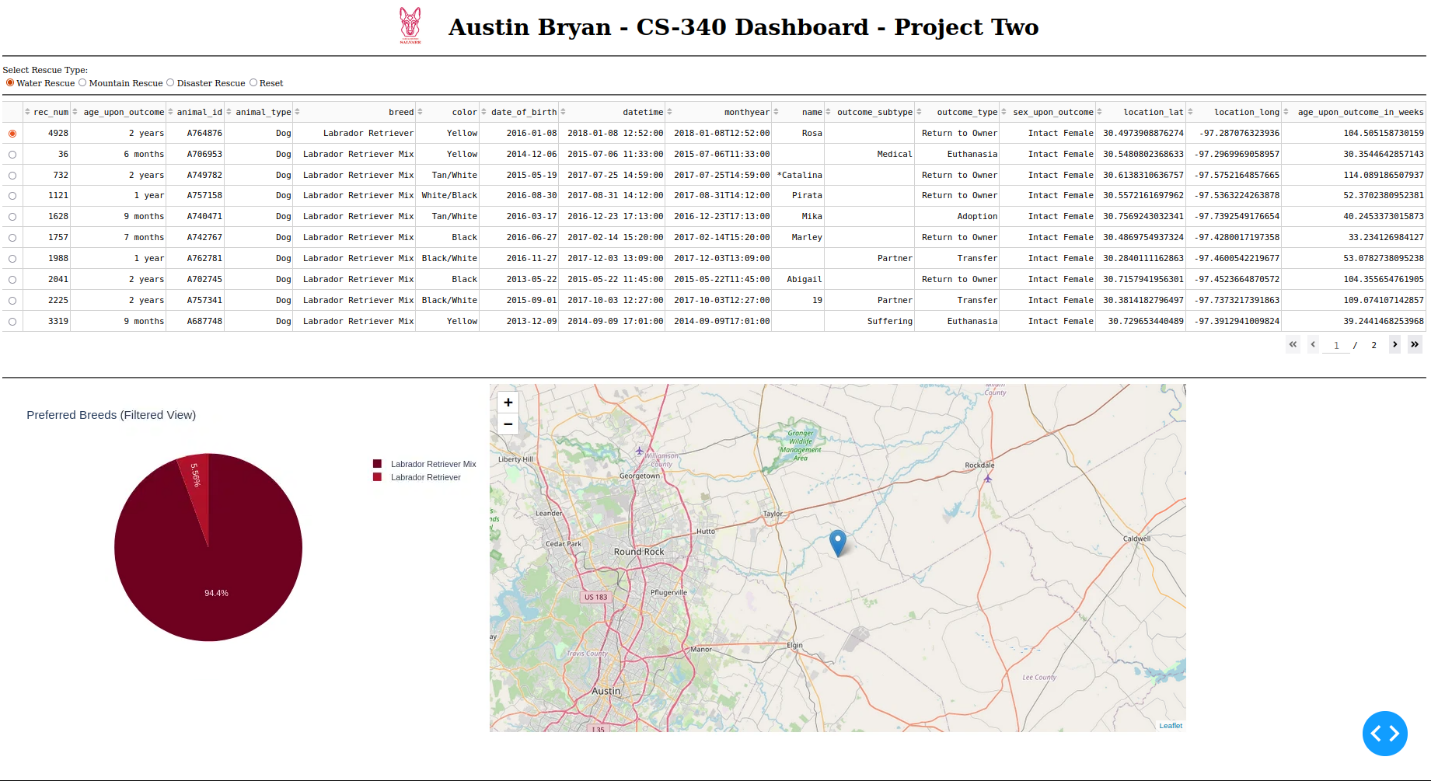
Screenshot 3: Mountain Rescue filter applied

*Displays only intact male German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, and Rottweiler breeds within the specified age range.*



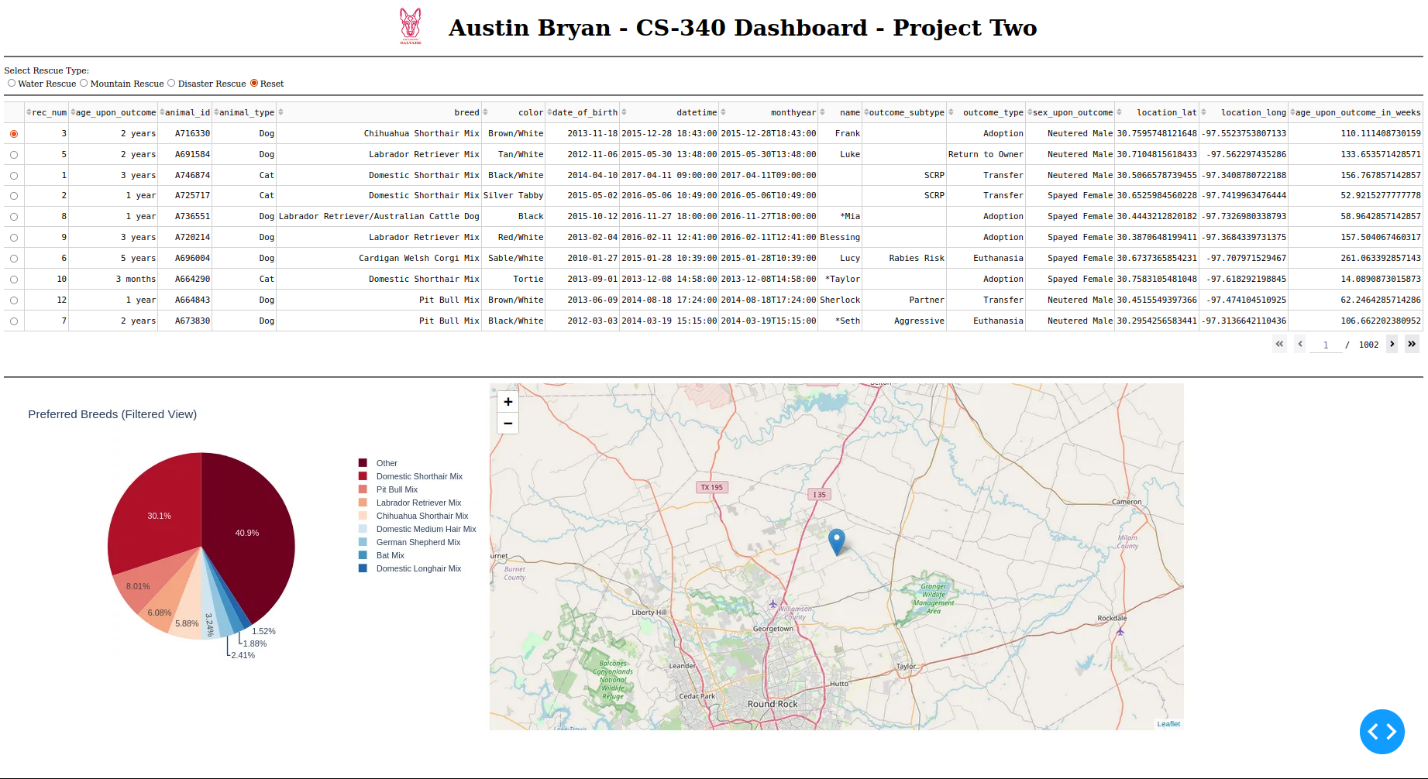
Screenshot 4: Disaster Rescue filter applied

*Filters the data to show only intact female Labrador Retriever, Chesapeake Bay Retriever, and Newfoundland breeds within the specified age range.*



Screenshot 5: Reset view

*Returns the dashboard to its unfiltered state, displaying the full dataset again.*



**Tools and Technologies Used**

This project was developed using a variety of tools and libraries selected for their relevance to modern web-based data applications, ease of integration, and suitability for rapid development in a Python environment:

Python 3.10

Python served as the core programming language due to its readability, rich ecosystem, and strong support for both web development and data manipulation.

MongoDB

MongoDB was used as the backend database to store and query animal outcome records from the Austin Animal Center. Its document-based structure was well-suited for the unstructured and nested nature of the data, and its integration with Python through the pymongo library enabled fast prototyping and flexible querying.

Dash (by Plotly)

Dash was chosen to build the web dashboard interface. It allows for the creation of interactive web applications entirely in Python, combining HTML components, data tables, and charts. Dash made it possible to connect frontend interactivity (such as rescue-type filters) directly to MongoDB queries with minimal overhead.

Dash Leaflet

Used to implement the geolocation chart, Dash Leaflet provided a highly customizable and responsive map component to display animal locations based on latitude and longitude fields in the dataset.

Pandas

Pandas was used for efficient manipulation of MongoDB query results. Its DataFrame structure simplified filtering, sorting, and preparing data for display in both the data table and charts.

Plotly Express

For dynamic visualizations, Plotly Express was used to create a pie chart representing breed distribution in the filtered dataset. It allowed for polished, interactive, and client-friendly charting without the need for complex JavaScript code.

Jupyter Notebook / JupyterDash

The dashboard was built within a Jupyter Notebook using JupyterDash, allowing for an integrated development experience within the Apporto virtual lab environment provided by the course.

**Resources Used**

* [MongoDB Python Driver (PyMongo) Documentation](https://pymongo.readthedocs.io/en/stable)  
  Used for understanding how to connect to MongoDB and execute CRUD operations in Python.
* [Dash by Plotly Documentation](https://dash.plotly.com/introduction)  
  Used to build the interactive dashboard, including radio buttons, data tables, and callbacks.
* [Dash Leaflet Documentation](https://dash-leaflet.herokuapp.com/)  
  Used to create the interactive map to show the location of selected animals.
* [Plotly Express Documentation](https://plotly.com/python/plotly-express)  
  Consulted for creating the breed distribution pie chart.
* SNHU CS-340 Project Guidelines  
  Provided by the course to define functionality, design expectations, and grading rubric.

## Project Development Process

## The following steps were taken to complete the Grazioso Salvare dashboard project:

## Setup and Database Integration

## The project began with the creation of a MongoDB database using the provided CSV dataset. Using the pymongo library, a Python module was created to support full CRUD (Create, Read, Update, Delete) functionality. Authentication was configured, and the database was connected to the dashboard via a class-based interface.

## Developing the Dashboard Framework

## A Jupyter Notebook was set up using JupyterDash to serve as the development environment. Core libraries such as Dash, Plotly Express, and Dash Leaflet were imported to support visualization and interactivity. Initial tests were conducted to confirm that the dashboard could render correctly.

## Building the Unfiltered Data Table

## An interactive data table was implemented using dash\_table.DataTable. This table dynamically displayed animal records from MongoDB. Key features such as pagination, sorting, and column highlighting were added to enhance usability. Additional precautions were taken to remove the \_id field, which can crash the table due to its ObjectId type.

## Filter Logic and Rescue Type Queries

## A radio button component was added to allow users to filter dogs by rescue type: Water Rescue, Mountain Rescue, Disaster Rescue, or Reset (unfiltered). Each option triggered a corresponding query in the CRUD module, which returned relevant results based on breed, age, and sex criteria. The data table updated automatically in response.

## Chart Implementation

## A pie chart was created using Plotly Express to display breed distribution for the filtered dataset. The pie chart includes a dynamic fallback to group uncommon breeds into an “Other” slice if too many categories are returned. Color palettes were selected to align with accessibility and branding considerations.

## Geolocation Map Integration

## A map was created to show the selected animal’s location. Map markers were updated dynamically when a row in the data table was selected. Column-based indexing was used to prevent crashes and increase code clarity.

## Branding and Polish

## The dashboard was branded to meet client requirements. The Grazioso Salvare logo was embedded at the top of the layout and linked to the SNHU website. A personal identifier ("Austin Bryan") was included in the header for authorship. Style customizations were added to improve layout, spacing, and readability.

## Testing and Screenshots

## Each feature of the dashboard was tested thoroughly. Screenshots were captured to show:

## The default unfiltered view

## Filtering by each rescue type (Water, Mountain, Disaster)

## The reset functionality

## These screenshots serve as visual proof of successful implementation and are included in this README

**Challenges Encountered and Solutions**

Throughout the development of the dashboard, several challenges were encountered and systematically addressed:

1. **Filtering by Breed Names**:  
   The dataset contained a large number of breed combinations and naming variations (e.g., "Labrador Retriever Mix", "Newfoundland/Labrador Retriever", etc.). Writing reliable queries without regex was particularly tedious, especially for water rescue filters. The solution was to manually inspect the list of unique breeds in the database and construct explicit $in filters for all valid combinations, even when results did not always return every expected breed (e.g., some breeds did not appear in water rescue results despite being included in the filter).
2. **Positioning the Logo**:  
   Aligning the Grazioso Salvare logo alongside the header text while ensuring accessibility and responsiveness was tricky. This was solved using a horizontal Flexbox layout in Dash’s html.Div with alignItems and gap styling to create a balanced visual header.
3. **Pie Chart Display Overload**:  
   When too many breeds were represented in the filtered dataset, the pie chart became crowded and unreadable. To resolve this, the top breeds (those >1.5%) were separated from the less common ones, which were grouped into a single “Other” slice. This approach made the chart more informative and visually digestible.
4. **Color Selection for Visual Consistency**:  
   In order to closely match the client-provided dashboard mockup, a browser eye dropper tool was used to extract the exact color palette. These color values were then hardcoded into the pie chart using Plotly’s color\_discrete\_sequence parameter.
5. **Screenshot Clarity and UI Feedback**:  
   Initially, the dashboard defaulted to the “Reset” filter, making the differences between "Reset" and the initial unfiltered state unclear in screenshots. To make filtering behavior more intuitive and screenshots more meaningful, the application was modified to launch with no selection pre-set, ensuring the user initiates the first filter explicitly and all views are a direct result of user interaction.

## Contact

Your name: Austin Bryan 2798128