Jael Andre

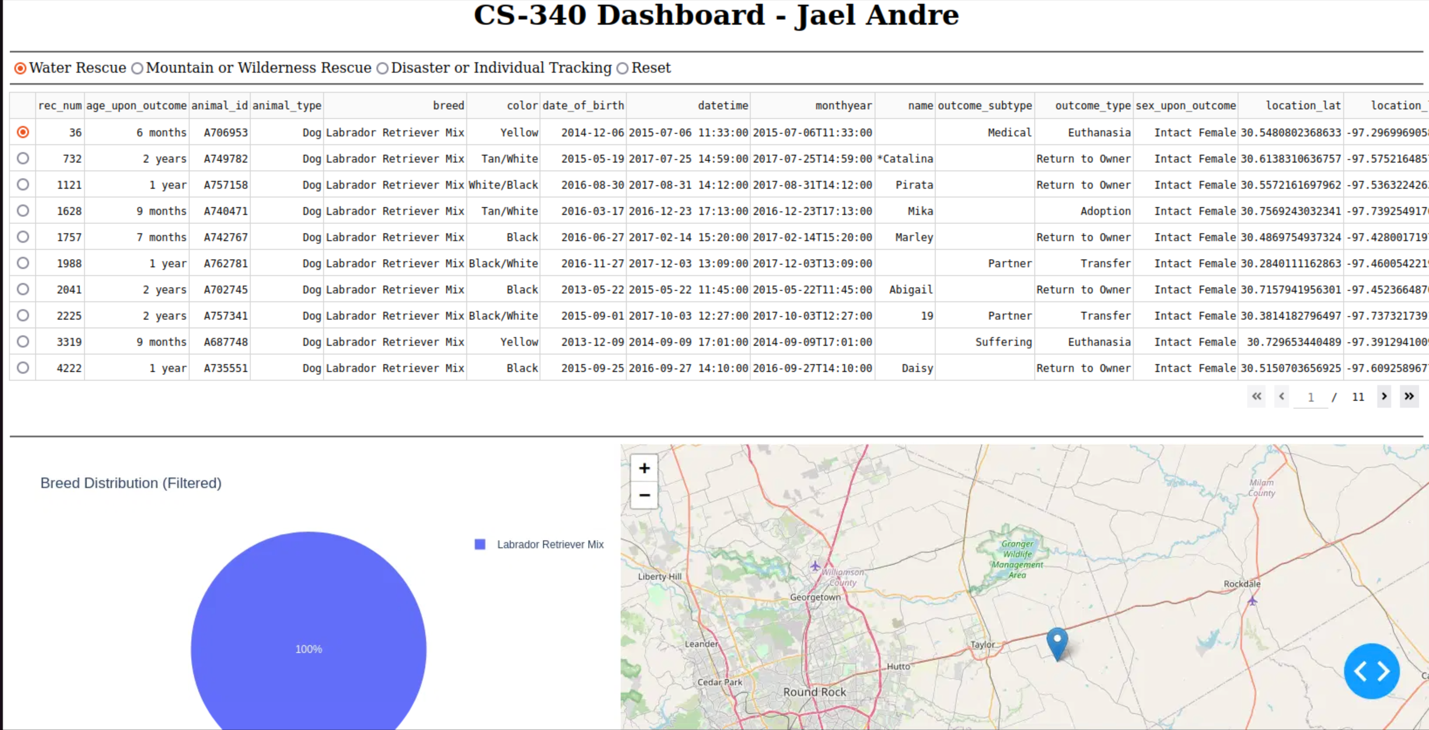
CS 340

Module 7 Project Two

A screenshot of a dashboard

Description automatically generatedProfessor Morris   
  
  
  
  
1. Dashboard Appearance

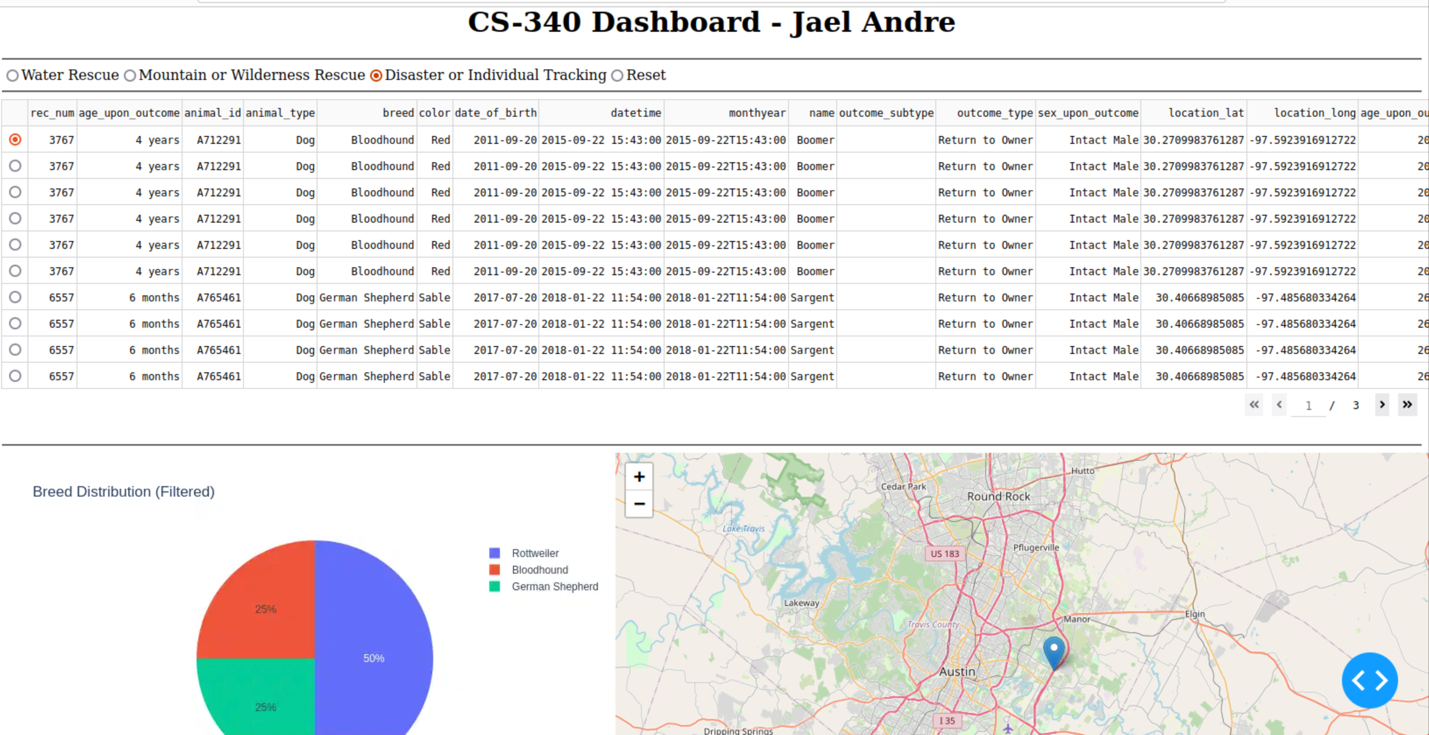
2. Water Rescue Filter



3. Mountain or Wilderness Rescue

A screenshot of a computer

Description automatically generated

4. Disaster or Individual Tracking

5. Reset (All Filters Reset with no data show)

A screenshot of a computer

Description automatically generated

In this project, I developed a web-based dashboard using Python, Dash, and MongoDB to visualize and interact with the Austin Animal Center Outcomes dataset. The goal was to create an interactive dashboard that allows users to filter data based on various rescue types, such as water rescue, mountain or wilderness rescue, and disaster or individual tracking. To achieve this, I utilized Dash to build the front-end and MongoDB to store and query the data. The dashboard features a data table that dynamically responds to the filter selections, a pie chart that visualizes breed distribution, and a geolocation map that updates based on the selected data. When the "Reset" filter is chosen, the pie chart resets to an empty state, visually indicating that no data is currently selected.

I chose MongoDB as the database solution because of its flexibility in handling unstructured data, which was ideal for storing the animal records in the dataset. MongoDB's JSON-like structure integrates seamlessly with Python, and the PyMongo driver made it easy to perform CRUD (Create, Read, Update, Delete) operations. By using MongoDB as the model component, I could quickly query and filter large datasets based on various attributes such as breed, age, and location. MongoDB's document-based nature allowed me to efficiently manage and retrieve complex datasets without the need for complex relational database schemas.

For the front-end, I used Dash, which allowed me to build the entire web application in Python. Dash provides a powerful framework for developing interactive, web-based visualizations, making it easy to connect the data model (MongoDB) with the view (the dashboard interface). The interactive filter buttons, data table, pie chart, and map were all constructed using Dash's intuitive components, and callbacks were used to enable real-time updates when users interacted with the dashboard. Dash's ability to handle data visualization through libraries like Plotly made it the perfect choice for this project. Additionally, the flexibility of Dash ensured that I could create a user-friendly interface with smooth transitions between different data filters.

Throughout the project, I faced a few challenges, such as ensuring that the pie chart would reset correctly and display an empty state when no data was selected. This required careful management of the callbacks and condition handling in the graph function. Another challenge was integrating the MongoDB query results with Dash’s data table and ensuring that the map updated correctly when rows were selected. These issues were resolved through iterative testing and debugging, allowing me to ensure the dashboard behaved as expected.

The tools I used include PyMongo for database interaction, Dash for building the web application, and Plotly for generating the charts and visualizations. These tools were ideal for this project because they allowed me to work entirely in Python while achieving a high level of functionality and interactivity. The project was deployed and tested successfully, and I captured screenshots of the running dashboard, demonstrating its core functionality.