# Grazioso Salvare Dashboard Project

## Overview

The Grazioso Salvare Dashboard Project is a full-stack web application developed using Python and the Dash framework. This dashboard allows users to interact with a MongoDB database containing the Austin Animal Center Outcomes data. Users can filter data by various rescue types and view the data via an interactive data table, a geolocation map, and additional charts. The dashboard also includes the Grazioso Salvare logo (linked to [www.snhu.edu](https://www.snhu.edu)) and a unique identifier (my name).

## Required Functionality

* **Interactive Data Table**: Displays the complete (unfiltered) dataset from MongoDB.
* **Filtering Options**: Users can filter the dataset by rescue types:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster Rescue or Individual Tracking
  + Reset (to return to the unfiltered state)
* **Dynamic Charts**:
  + A **geolocation map** (using Dash Leaflet) that shows the location of the selected animal based on its latitude and longitude.
  + A **secondary chart** (a pie chart) displaying breed distribution.
* **Branding & Identification**: Includes the Grazioso Salvare logo (clickable, linking to [www.snhu.edu](https://www.snhu.edu)) and my unique identifier (“Andrei Shostak – CS-340 Dashboard”).
* **Interactive Updates**: All widgets update dynamically based on user input.

## Tools & Technologies

* **Python**: The primary programming language.
* **Dash & Plotly**: Used to create the interactive web application (view and controller components).
* **Dash Leaflet**: Provides geolocation (mapping) functionality.
* **MongoDB**: Serves as the data model, offering a flexible, JSON-like document storage that integrates seamlessly with Python.
* **PyMongo**: The official Python driver for MongoDB used for CRUD operations.
* **Pandas**: For data manipulation and converting MongoDB query results into DataFrames.
* **Base64 & OS Libraries**: Used for encoding and displaying the logo image.

## Rationale for Tool Choices

* **MongoDB** was selected due to its schema-less design, flexibility, and native support for JSON-like documents—making it ideal for the provided data set.
* The **Dash Framework** offers a powerful and flexible platform for building interactive web dashboards entirely in Python.
* **Plotly Express** is used for creating interactive charts quickly and with minimal code.
* **PyMongo** simplifies database connectivity and operations, making it easier to integrate with our Python dashboard.

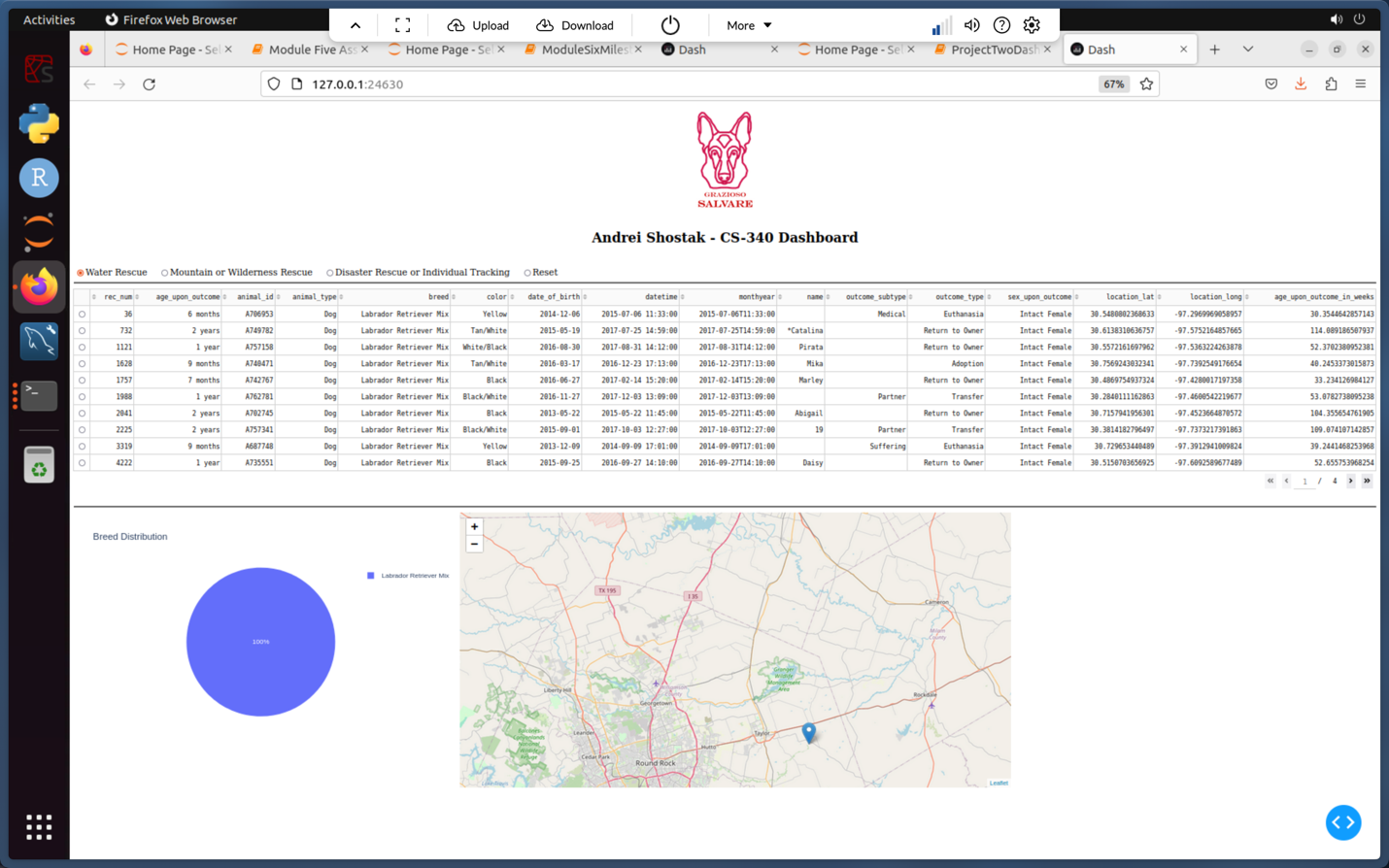
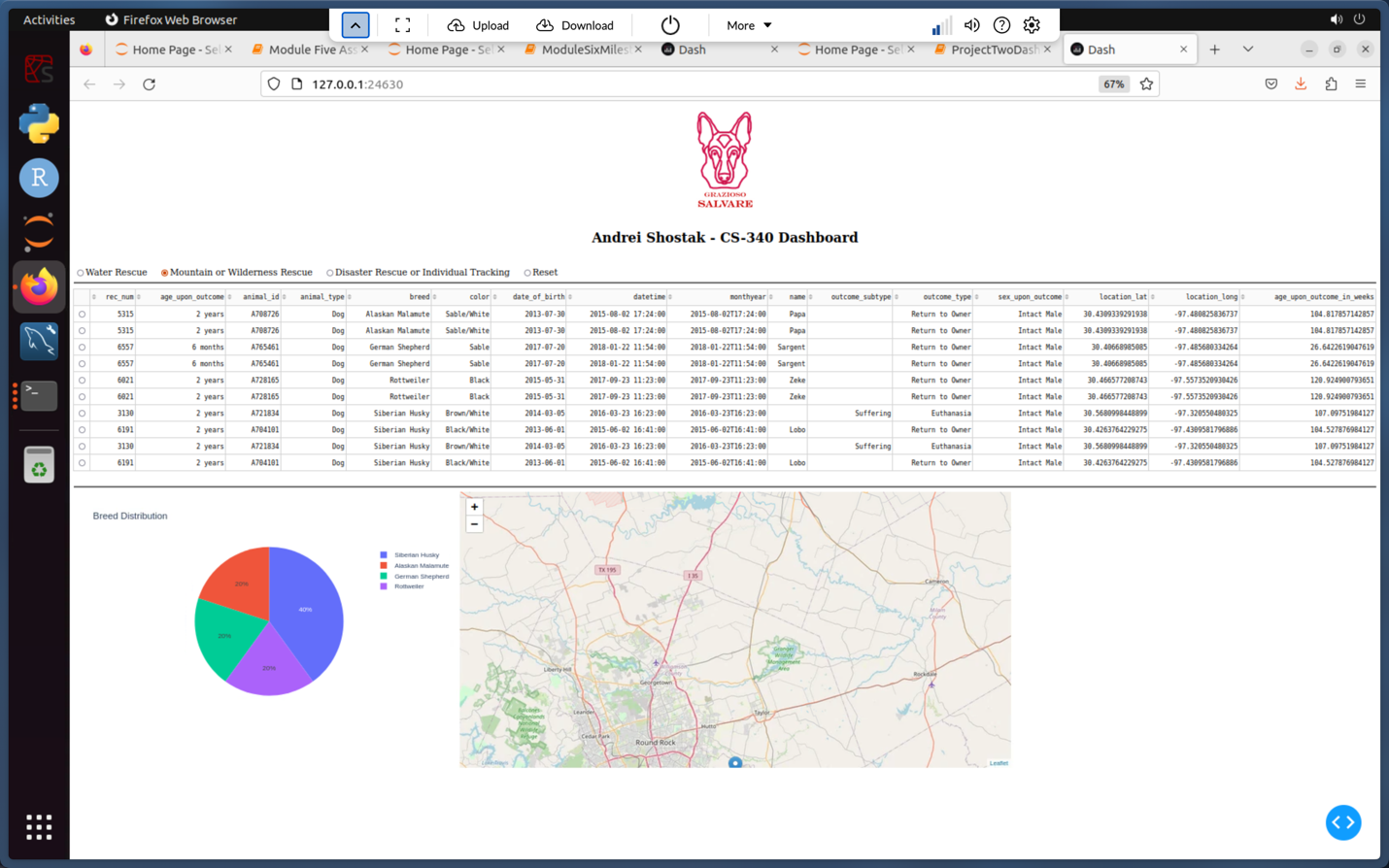
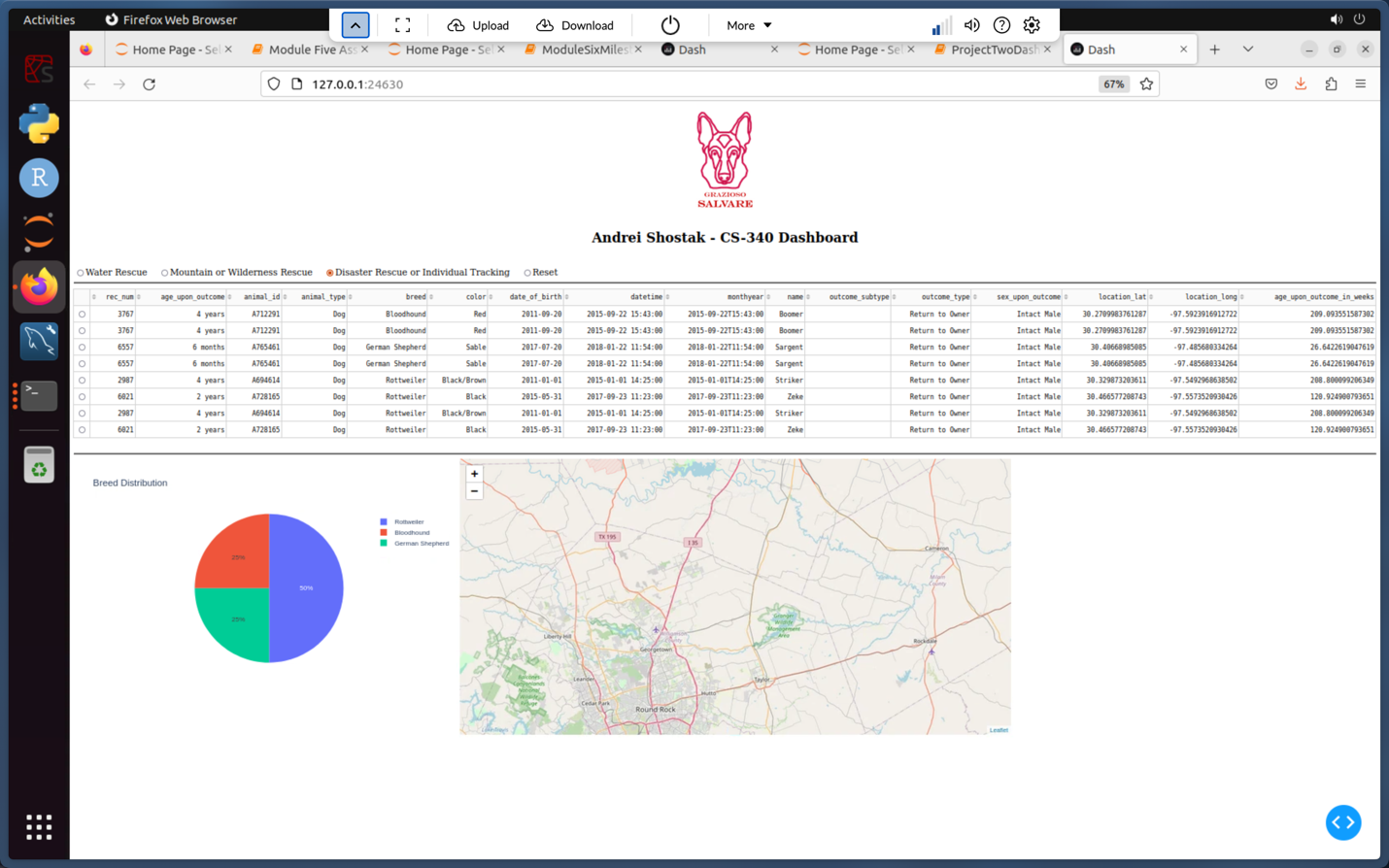
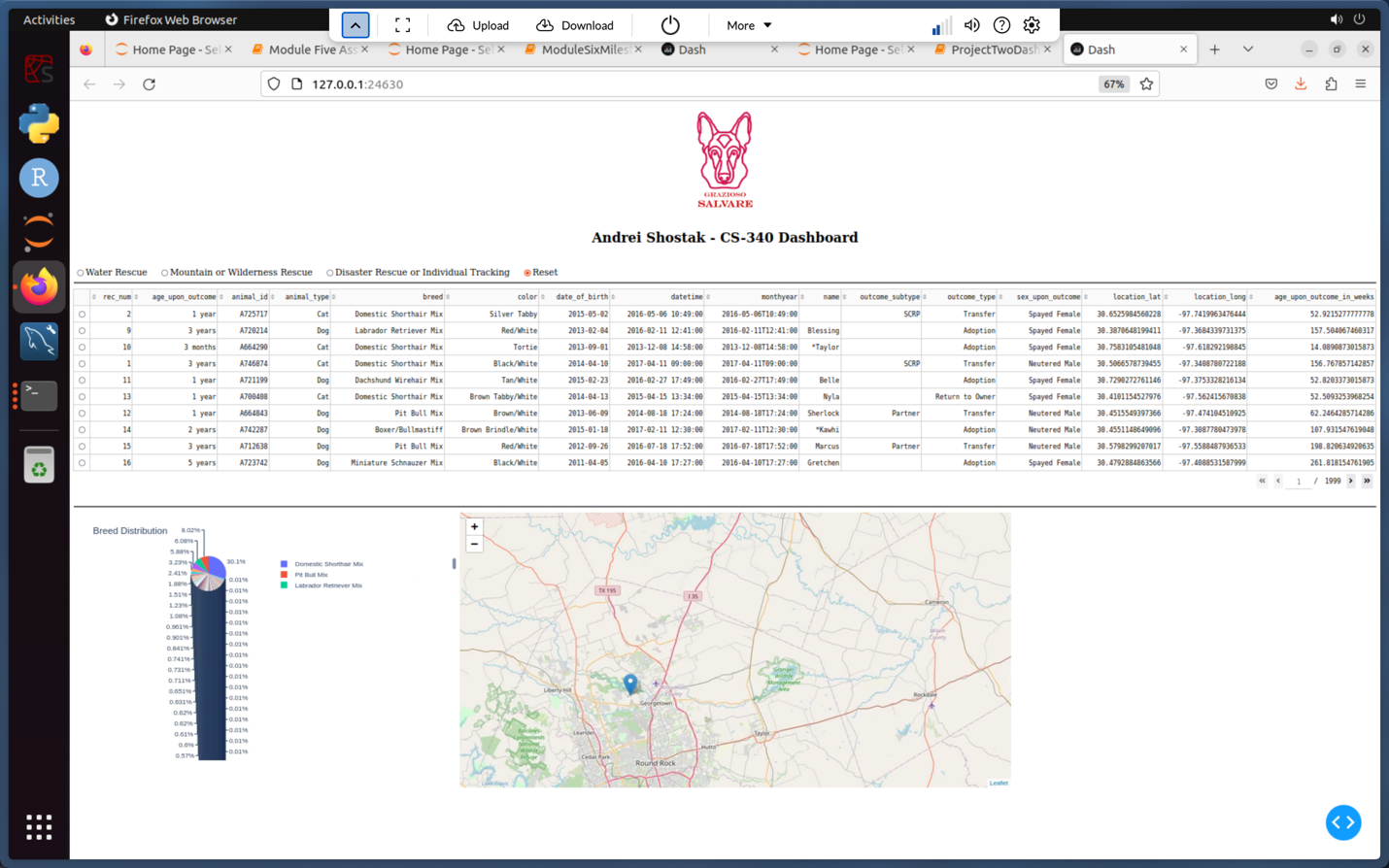
## Reproducing the Project

1. **Clone or Download the Repository**  
   Ensure the following files are present:
   * ProjectTwoDashboard.ipynb
   * animal\_shelter.py
   * grazioso\_logo.png
2. **Install Required Libraries**  
   Use pip to install necessary packages:

* pip install dash dash-leaflet jupyter-dash plotly pymongo pandas numpy

1. **Set Up MongoDB**  
   Import the Austin Animal Center Outcomes CSV into MongoDB and create the user aacuser with the appropriate credentials and roles.
2. **Run the Notebook**  
   Open ProjectTwoDashboard.ipynb in Jupyter Notebook and run all cells to launch the dashboard.
3. **Interact with the Dashboard**
   * Use the filter options to update the data table, pie chart, and geolocation map.
   * Verify that the dashboard displays the Grazioso Salvare logo (with link) and the unique identifier.

## Screenshots

* **Filtered: Water Rescue**:  
  
* **Filtered: Mountain or Wilderness Rescue**:  
  
* **Filtered: Disaster Rescue/Individual Tracking**:  
  
* **Reset State**:  
  

## Project Steps and Challenges

1. **Dashboard Setup**:
   * Created the basic layout using Dash components (DataTable, RadioItems, map, pie chart).
   * Integrated the Grazioso Salvare logo and added my unique identifier.
2. **Data Integration**:
   * Utilized the existing animal\_shelter.py CRUD module to fetch data from MongoDB.
   * Converted the data to a Pandas DataFrame and removed the \_id column to avoid serialization issues.
3. **Interactive Filtering**:
   * Developed callbacks to update the DataTable based on selected filter options.
   * Implemented dynamic updates for the pie chart and geolocation map based on filtered data.
4. **Challenges Encountered**:
   * **Authentication and Connection Issues**: Resolved by ensuring credentials and connection parameters were correctly set.
   * **Data Mapping Errors**: Fixed by verifying DataFrame column names and using them in callbacks instead of numeric indices.
   * **User Interface Tweaks**: Iteratively refined the layout for better readability and usability.
5. **Overcoming Challenges**:
   * Debugged by printing DataFrame shapes and sample rows.
   * Added error handling in callbacks to provide clear user feedback.
   * Consulted documentation for Dash, PyMongo, and MongoDB for best practices.

## Resources & References

* [Dash Documentation](https://dash.plotly.com/)
* [Dash Leaflet Documentation](https://dash-leaflet.herokuapp.com/)
* [MongoDB Documentation](https://docs.mongodb.com/)
* [PyMongo Documentation](https://pymongo.readthedocs.io/)
* [Plotly Express Documentation](https://plotly.com/python/plotly-express/)

## Conclusion

This project demonstrates the integration of a MongoDB database with a Python-based dashboard built using the Dash framework. The solution allows Grazioso Salvare to interact with and visualize animal shelter data, enabling the identification of suitable dogs for rescue training. The dashboard is designed for ease of use, scalability, and effective data visualization, and the accompanying documentation provides clear instructions for reproduction and maintenance.