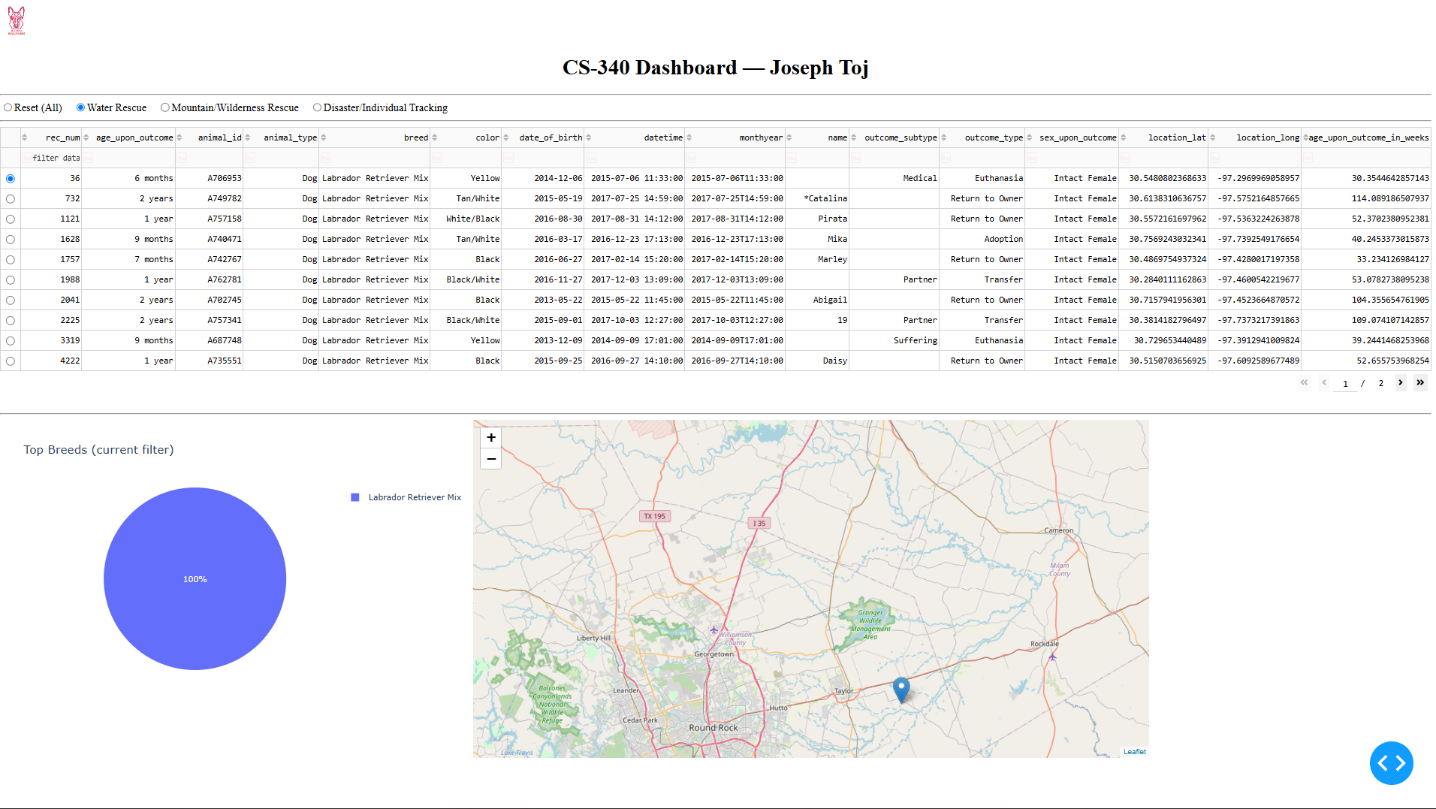
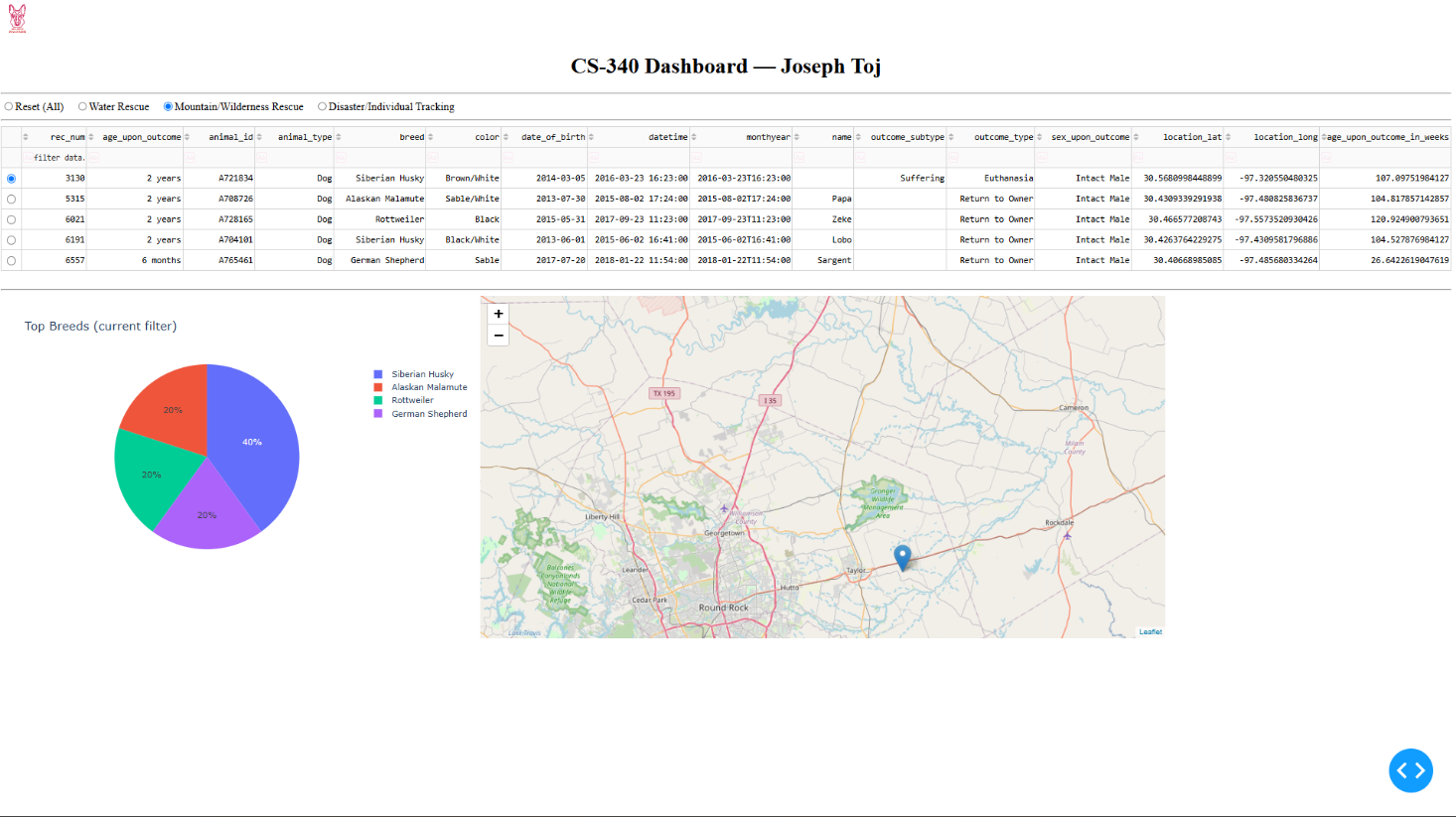
Grazioso Salvare Dashboard – CS 340 Project Two

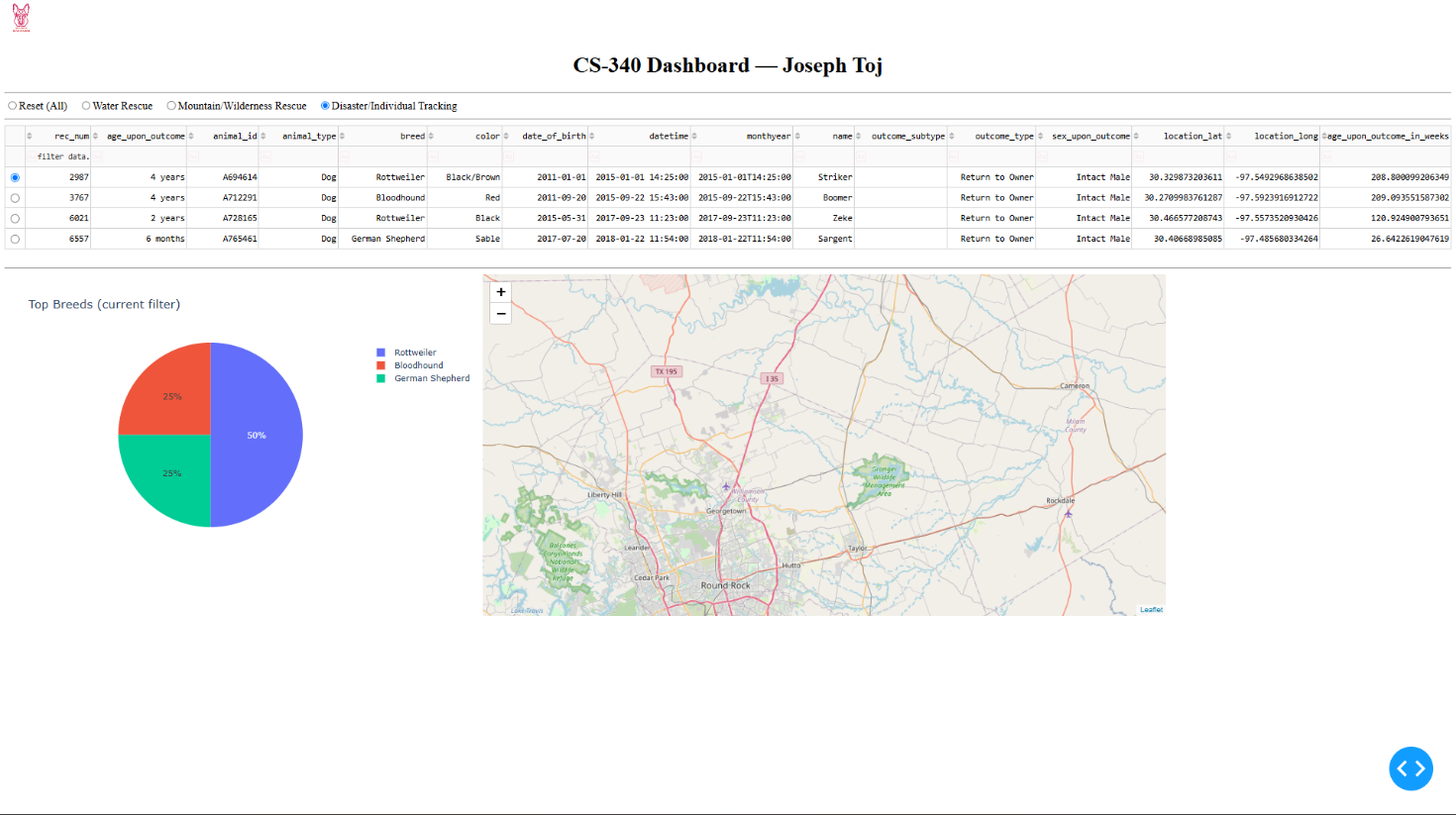
Joseph Toj

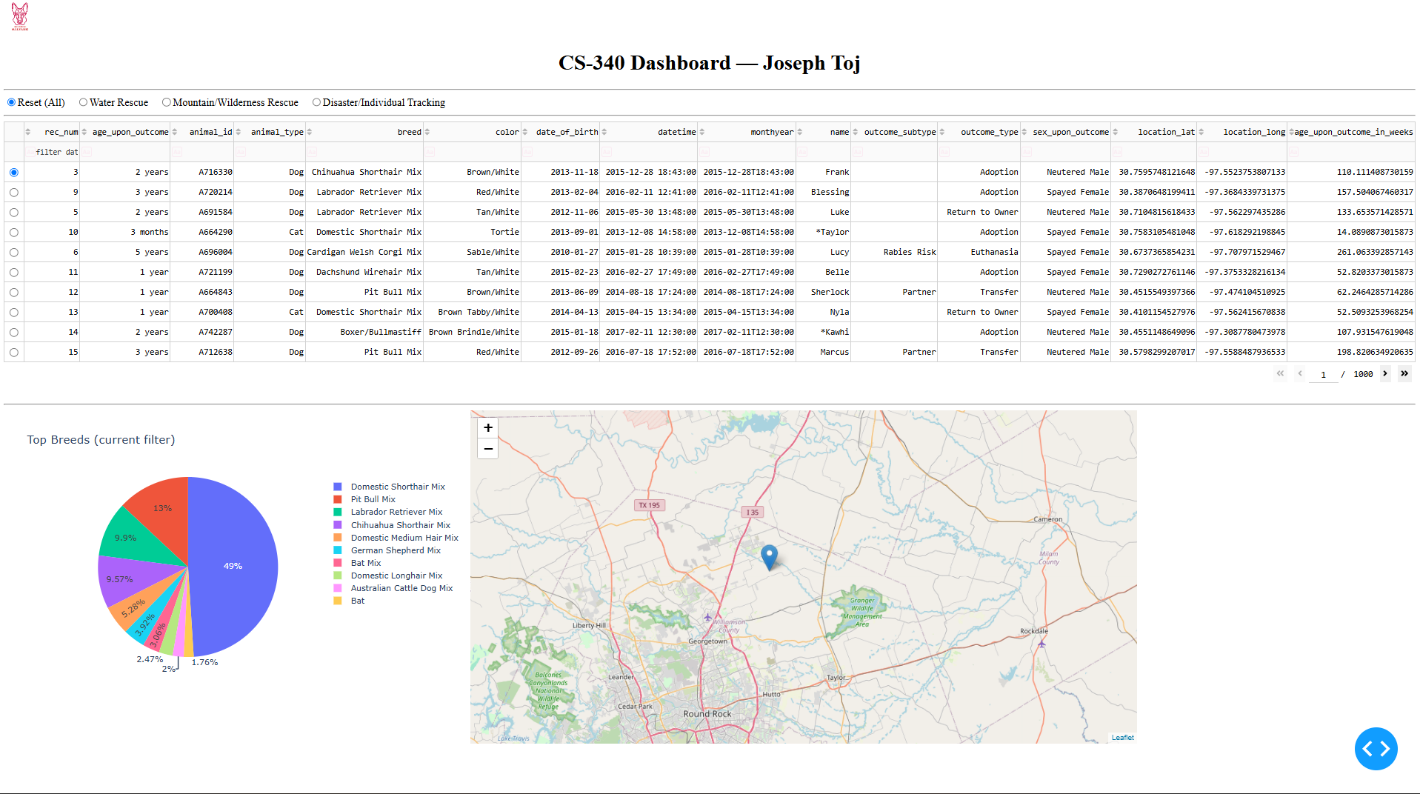
A screenshot of a computer

AI-generated content may be incorrect.









1) Required Functionality (with Proof)

This solution provides an interactive, MongoDB-powered dashboard for Grazioso Salvare:

Interactive options (radio items) to filter the Austin Animal Center Outcomes dataset by Water Rescue, Mountain/Wilderness Rescue, Disaster/Individual Tracking, and Reset (All).

An interactive DataTable that reacts to the chosen filter (pagination, sorting, filtering is on).

A geolocation chart (Leaflet) that pins the currently selected row (first row is selected by default).

A second chart (pie) that updates according to the same filtered, visible table data.

The Grazioso Salvare logo and my UID are presented in the dashboard header.

Screenshots / Screencast (insert images below each bullet):

Starting State (Reset/All) — [Image]

Water Rescue — [Image]

Mountain/Wilderness Rescue — [Image]

Disaster/Individual Tracking — [Image]

Reset (All) — [Image]

2) Tools Used and Rationale

MongoDB (Model):

A document database that “fits naturally with semi-structured, JSON-like records”. It has a Python driver that plays well with the CRUD module, and enables quick “retrieve all” as well as filtered queries, with projection. Supports flexible schema for the fields I added to this dataset (location\_lat, location\_long, age\_upon\_outcome\_in\_weeks).

Dash (View & Controller):

Dash provides the UI widgets (radio items, DataTable) as well as the reactive callbacks (controller logic), all in Python. The DataTable makes pagination/sorting/filtering available with very little code, and the callbacks are used to update the Leaflet map and the pie chart when filters change, or the row selection updates.

dash-leaflet (Geolocation View):

dash-leaflet was used to “pin” the selected animal’s coordinates, with marker, tooltip (breed), and popup (name).

3) Why MongoDB for the Model

Natural fit for JSON: Plays well with Python dicts and Pandas.

Flexible schema: Accepts additional fields I added for this project (coordinates, age in weeks).

Simple queries: Filter logic maps easily to Mongo query operators ($in, $gte, $lte), fast server-side filtering.

4) Dash Framework (View + Controller)

View: Dash layout includes all widgets (Logo + UID, radio filters, DataTable, charts).

Controller: Dash callbacks connect widgets to data updates:

Filter → Table: The radio selection builds a Mongo query and reloads the table.

Table → Charts: The current table view powers the Leaflet map (selected row) and a pie chart (top breeds).

5) Reproduction Instructions

Prerequisites

Python environment with: dash, jupyter\_dash, dash-leaflet, plotly, pandas, pymongo.

Running MongoDB with the AAC/aac dataset loaded:

Database: aac

Collection: animals

User: joseph / password (has read access to aac)

Steps

Put ProjectTwoDashboard.ipynb and CRUD\_Python\_Module.py in the same directory.

Make sure the logo PNG is in code\_files/ (your notebook has a bit of a path-finder built in).

Open JupyterLab/Codio and run the notebook, from top to bottom.

Apply the radio filters to produce the four filtered states and capture required screenshots.

Zip the two source files and include this README (Word) with embedded screenshots.

6) Steps Taken to Complete the Project

Started from Module Six: unfiltered DataTable + Leaflet map driven by table selection.

Added radio filters for the three rescue categories + reset.

Implemented Mongo queries (controller) based on the Dashboard Specifications rescue breeds, sex, and age-in-weeks ranges.

Wired the DataTable to auto-update from Mongo when filters change.

Added a pie chart (top breeds) that re-computes from the visible table data.

Made sure the map updates as the selected row changes (default selects first row).

Branded the header with Grazioso logo and UID.

Captured screenshots of starting state and each filter state.

7) Challenges and How They Were Overcome

Mongo auth/DB naming: Made sure using the correct DB (aac) and user (joseph/password) in mongosh.

ObjectId issues in DataTable: Dropped the \_id column before passing data to the table.

Map indices/columns: Made sure the DataTable always selects a row by default, so the map always renders a pin.

8) Resources

Dash & Dash DataTable documentation

dash-leaflet documentation

MongoDB Python Driver (PyMongo)

Course-provided Dashboard Specifications Document and the AAC dataset