Lasupe Xiong

CS 340

Instructor Reed Perkins

10/14/2025

Module 7 Project Two

**Project Overview**

This project concludes the construction of the Grazioso Salvare dashboard, an interactive online application that links to a MongoDB database to assist the customer in identifying dogs that are suitable candidates for various forms of rescue training. The dashboard leverages data from the Austin Animal Center Outcomes dataset and has simple settings that allow users to filter dogs by rescue type (water rescue, mountain or wilderness rescue, disaster, or individual tracking) and monitor their information in real time.

The application enables the client to:

* Browse and search all animal records.
* Filtering animals is based on rescue suitability.
* View dynamic geolocation maps showing animal locations.
* Analyze top dog breeds by frequency.

**Functionality**

The dashboard was developed in Python using Dash and follows the MVC design pattern:  
- Model: MongoDB serves as the backend data source. The crud.py module manages database operations using a read() function that retrieves all or filtered documents.  
- View: Dash HTML and DataTable components display data in an interactive table, along with a Leaflet-based map and Plotly bar chart.  
- Controller: Callback functions connect user input from filters to the model and view, ensuring data updates dynamically.

Key Features:  
- Interactive Filter Options: Drop-down menu and buttons for selecting rescue type.  
- Dynamic Data Table: Displays real-time data pulled from MongoDB via the CRUD module.  
- Geolocation Map: Displays animal locations using latitude and longitude fields.  
- Top 10 Breeds Chart: A bar chart that updates with current filter selections.  
- Unique Identifier: “LasupeXiong-ProjectTwo-2025” displayed on the dashboard.  
- Grazioso Salvare Logo: Branding element included per project requirements.

**Tools**

MongoDB / PyMongo | Backend database for animal records  
Dash (Plotly, Dash Leaflet) | Framework for interactive data visualization  
Pandas / NumPy | Data transformation and filtering  
Matplotlib | Data visualization support  
Python 3.11 / JupyterDash | Environment for interactive development and testing  
  
MongoDB was chosen because of its document-based structure and seamless integration with Python through PyMongo. Dash was used because it provides both the controller and view in a single framework, making it easy to link filters, tables, and visual charts interactively.

**Development**

1. Setup Database: Imported the aac\_shelter\_outcomes.csv dataset into MongoDB using mongoimport.  
2. Create CRUD Module: Implemented crud.py from Project One to handle data retrieval and fallback authentication.  
3. Develop Dashboard: Created the Dash layout with filters, DataTable, and geolocation map.  
4. Add Interactivity: Used Dash callbacks to update components when the user selects a rescue type.  
5. Add Breed Chart: Included a Top 10 Breeds bar chart that dynamically updates based on current data.  
6. Test & Debug: Verified callback synchronization, map markers, and authentication fallback.  
7. Deployment: Confirmed local Dash app execution via Codio proxy URL.

**Unfiltered Dashboard View – Shows all components (logo, unique ID, data table, charts).**

A screenshot of a computer

AI-generated content may be incorrect.

**Water Rescue Filter Active**

A screenshot of a computer

AI-generated content may be incorrect.

**Mountain or Wilderness Rescue Filter Active**A screenshot of a computer

AI-generated content may be incorrect.

**Disaster or Individual Tracking Filter Active**

A screenshot of a computer

AI-generated content may be incorrect.

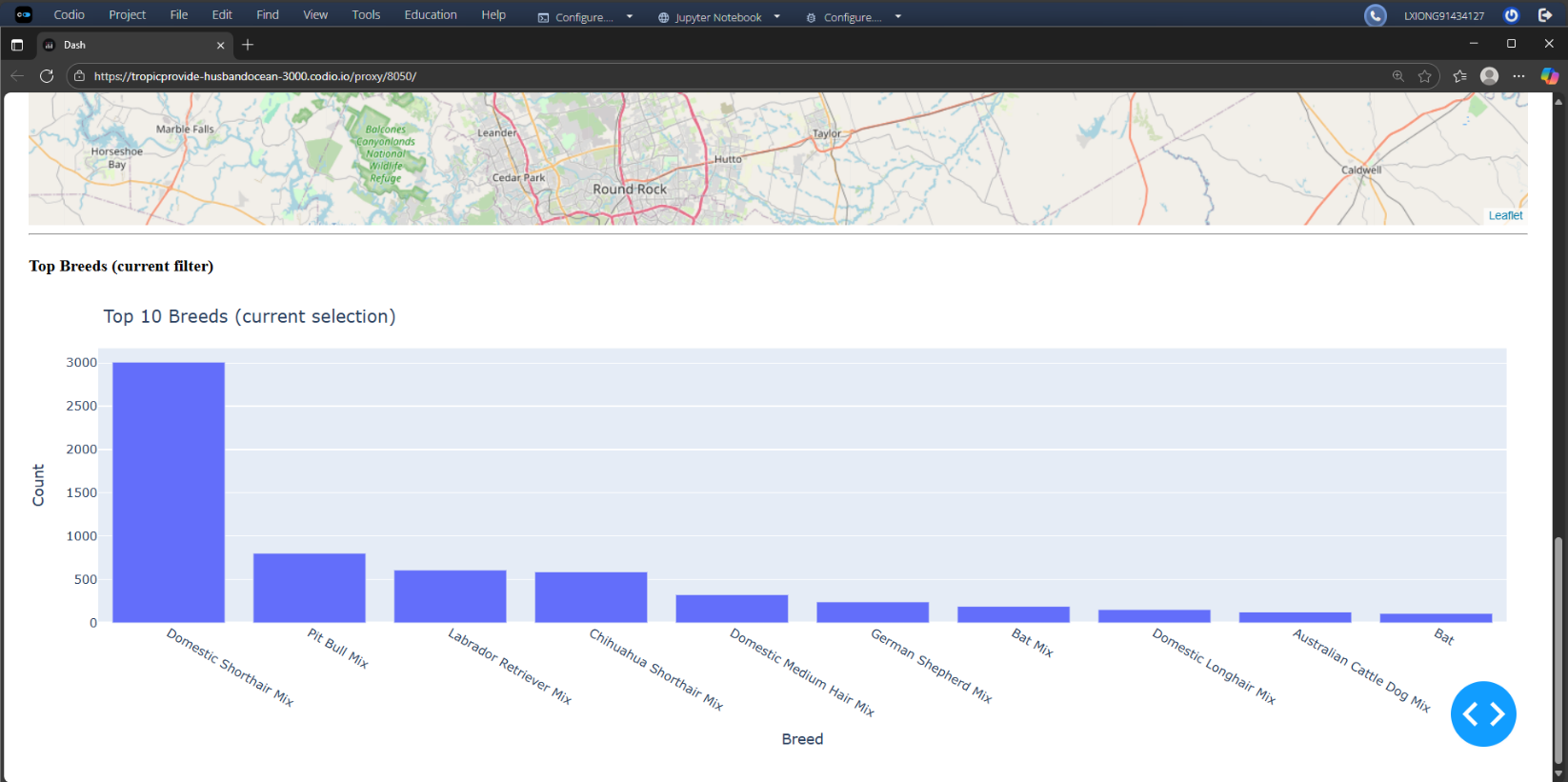
**Reset View – Returns to full dataset.**

A screenshot of a computer

AI-generated content may be incorrect.

A map with a location pin

AI-generated content may be incorrect.



**Challenges and Solutions**

MongoDB authentication error | Implemented fallback no-auth connection when credentials fail  
Callback errors from Plotly chart | Cleaned up column naming to prevent duplicate fields (“count”, “breed”)  
Leaflet map not updating | Added guard conditions to handle missing lat/long fields  
Layout alignment issues | Applied Dash style rules for consistent padding and alignment

**File Description**

ProjectTwoDashboard.ipynb | Main dashboard application with all visualization and callbacks  
crud.py | Handles MongoDB connection and CRUD operations  
aac\_shelter\_outcomes.csv | Dataset imported into MongoDB  
Grazioso Salvare Logo.png | Logo displayed on dashboard  
README.docx | Documentation and proof of execution

**References**

MongoDB Documentation  
MongoDB. (n.d.). *MongoDB manual.* <https://www.mongodb.com/docs/>

**Dash by Plotly Documentation**  
Plotly. (n.d.). Dash user guide & documentation. <https://dash.plotly.com/>