Sabin Neupane

Jeff Sanford

Project Two

April 19, 2025

The project offers data visualization for Grazioso Salvare, an animal rescue organization. It combines the data with MongoDB and the Dash library to offer an interactive visualization to display, filter, and view rescue animal data by the type of mission.

The product resulting from

• A filterable data table that is sortable.

An interactive world map with markers.

• A pie chart of the distribution of the sex of the selected animals.

• A breed distribution bar graph.

• Radio buttons to filter by mission requirements.

• Interactive, integrated Dash component layout.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Disaster or Individual Tracking

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Mountain or wilderness rescue:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Water Rescue:

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

Tools Used:

Dash (by Plotly): Used to develop the interactive dashboard as an application within the Jupyter framework.

dash-leaflet: Is utilized for displaying interactive maps with location markers.

plotly.express: Used to produce plots like pie charts and bar charts.

dash-table: To create dynamic, filterable tables

Pandas: for dealing with MongoDB data structures

base64: To embed images into the dashboard (logo).

Mongo DB

MongoDB is utilized to store animal shelter data and is used as the application template. The database is accessed via an AnimalShelter custom class within the crud.py module that encapsulates CRUD operations. MongoDB is particularly suited to the project because it is lenient about semi-structured JSON-like documents and is easy to interface with Python via PyMongo. The project utilizes JupyterDash to embed Dash into Jupyter Notebooks so that the dashboard is run inline to make development and testing easier.

Dash Framework:

Dash is both the view, and the controller of this web app. Dash provides one with the ability to create an interactive and clean interface using only Python without requiring frontend frameworks. Callback functions govern the relationship between user input (i.e., radio buttons) and dashboard charts and tables, with dynamic updates and responsiveness.

Development and testing were performed in JupyterDash, which combines Dash functionality along with the convenience of inline execution of Jupyter Notebooks.

Steps to complete:

For the project, I first created an interface to MongoDB via a custom AnimalShelter class to carry out CRUD operations: create, read, update, and delete, such that the application could communicate successfully with the rescue data. Having finished the backend, I next went about building the dashboard via Dash, where I added features like data table, an interactive map, a pie chart, and a bar chart to plot the data. I added radio buttons to filter by animal types of missions, as well as used Dash callback functions to dynamically refresh the charts as well as the table on user input. The map was generated through dash-leaflet to be able to plot animal positions as markers, while using plotly. Express was used for the pie chart as well as the bar chart.

Some of the challenges I overcame included handling invalid ObjectId input to MongoDB and debugging layout arrangement issues, but I overcame these by entering validated IDs and using an adaptive layout that changes with different screen sizes. Having tested it within the Jupyter setting, the dashboard was finished to be able to enable Grazioso Salvare’s mission monitoring as well as data analysis.

Challenges & Solutions:

Some debugging was required during development. The issue was one of MongoDB ObjectId exceptions, where invalid ObjectIds such as "aacuser" caused exceptions. The issue was corrected by altering the read () method in crud.py to validate ObjectIds before querying. The second issue was getting the map and charts to properly align, as elements of the Dash layout initially misaligned or overlapped. The problem was corrected by using a flex layout for both the side-by-side map and pie chart, making it more responsive and readable.