Grazioso Salvare Dashboard Project README

Project Overview

This dashboard helps users to interact with and visualize data from the Austin Animal Center Outcomes dataset. The following dashboard helps Grazioso Salvare to identify dogs that are good candidates for search-and-rescue training. By utilizing an interactable dashboard that helps the client to retrieve information based on what type of search-and-rescue animal they are looking for, the dashboard outputs information in the form of an interactable chart, as well as geolocation data for all the animals. By utilizing Python, MongoDB, and the Dash framework, this dashboard aims to help Grazioso Salvare in determining prime animal candidates for specific search-and-rescue operations.

Screenshots

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

Water Rescue:

A screenshot of a computer screen

Description automatically generated

Mountain or Wilderness Rescue:

A screenshot of a computer

Description automatically generated

Disaster Rescue or Individual Tracking:A screenshot of a map

Description automatically generated

Reset:

A screenshot of a computer screen

Description automatically generated

Tools

MongoDB

MongoDB was the ideal database to handle the Austin Animal Center Outcomes dataset due to its ability to handle complex datasets. MongoDB has advanced query capabilities that support powerful and abstract data retrieval. Also, its ability to be accessed via Python through the PyMongo library allows for seamless integration with Python applications. PyMongo provides an intuitive interface for performing CRUD operations and complex queries, making the combination of both Python and MongoDB an ideal choice for this project.

Dash Framework

Dash was chosen as the framework to pair with MongoDB because it is simple to use while also being able to create advanced and interactive web applications. Dash utilizes Flask for the backend, and Plotly for the graphs to develop a complex interactive dashboard that helps users visualize data in various graph styles. Additionally, Dash has a callback mechanism that can enable the creation of responsive web applications, that provide real-time updates and react upon user input.

Jupyter Notebooks

Jupyter Notebooks offers in interactive environment in which to run code segments that can connect to various platforms or databases. Its unique ability to handle various programming languages makes it an asset for constructing applications in an integrated environment. For this project, Jupyter notebooks provided an efficient development environment to write, test, and debug Python code. This notebook helped interface and facilitated the overall construction of the dashboard, providing a robust and interactive platform in which to manage data effectively.

Additional Links

<https://dash.plotly.com/dash-core-components/geolocation>

<https://dash.plotly.com/dash-core-components/dropdown>

<https://plotly.com/python/pie-charts/>

Steps Taken to Complete the Project

1. Install or ensure access to various tools like MongoDB, Jupyter Notebooks, and the necessary Python libraries like Dash, Plotly, and PyMongo.
2. Set up MongoDB: Import the Austin Animal Center Outcomes dataset into the MongoDB database to handle the data storage.
3. Develop a Python CRUD module using PyMongo that interacts with the MongoDB database. Make sure it includes methods for creating, reading, updating, and deleting records.
4. Connect to MongoDB from the Jupyter Notebook and ensure data retrieval is successful.
5. Set up the Dash Framework in Jupyter Notebook.
6. Add the logo and the unique identifier.
7. Add the dropdown menu in which to filter data related to the Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and Reset.
8. Include callback function that updates the data table dynamically based on the filtered criteria within the dropdown menu.
9. Create a geolocation chart using Dash Leaflet that helps visualize location data for the animals.
10. Include a Pie chart that displays the proportion of breeds based on the filtered data.
11. Test each individual component to ensure it is working correctly.
12. Debug any potential issues related to data retrieval and address any errors.
13. Deploy the dashboard in the Jupyter Notebook environment and ensure it is working successfully.

Challenges Encountered

1. Issues related to data import when trying to import the large CSV dataset into the MongoDB database. The solution involved ensuring the csv was successfully downloaded and that the import command was correct.
2. Problems surfaced when ensuring that the ‘aacuser’ had sufficient admin privileges when performing CRUD operations on the dataset. The solution was to include this privilege to the user before queries were performed.
3. Additional problems manifested with getting the callback functions to work properly. The solution was to verify the callbacks were correctly capturing and processing the input values from the dropdown menu.
4. Additional error handling was included with the geolocation callback to help correct errors in the map displaying correctly.
5. Received errors related to port conflicts. By assigning a specific port for Dash to run on, was able to circumvent this error since I was unable to kill the current process.

By addressing each of these challenges systematically, I was able to build a successful project and develop a fully functional and interactive dashboard for Grazioso Salvare.