**Project Name: Animal Shelter Dashboard**

Functionality

This dashboard was created to allow users to interactively explore data from an animal shelter. Key functionalities include:

* Interactive table display of animal data.
* Filtering options to view animals based on rescue type.
* Map visualization showing the location of the selected animal.
* A pie chart that dynamically updates based on the data shown in the table.

Tools Used

* Python: The core programming language used.
* Dash: A Python framework for building web applications. Chosen for its ability to create interactive, web-based dashboards with minimal setup.
* Pandas: Used for data manipulation and analysis.
* MongoDB: Served as the database to store and query data. It was chosen for its flexibility with unstructured data and ease of integration with Python.
* Plotly: For creating the interactive data visualizations.
* JupyterDash: To run the Dash app within a Jupyter environment for development and testing.

MongoDB as the Model Component

MongoDB is a NoSQL database that provides high performance, high availability, and easy scalability. It works well with Python, allowing for complex queries and data aggregation. For this project, MongoDB was used due to:

* Its schema-less nature, which is well-suited to the varied data from animal rescues.
* The ability to handle large volumes of data efficiently.
* Robust querying capabilities that are essential for filtering the dashboard data.

Dash Framework for View and Controller

Dash provides the front-end and controls the user interactions. It was chosen because:

* It is written on top of Flask, Plotly.js, and React.js.
* Dash is reactive which means the dashboard is automatically updated with new data.
* It supports interactive components that are essential for a dynamic user experience.

Resources

* Dash User Guide
* MongoDB Python Driver
* Pandas Documentation

Reproduction Steps

* Set up a Python environment and install required packages: Dash, Pandas, PyMongo, etc.
* Ensure MongoDB is installed and running. Import the initial dataset into the MongoDB database.
* Run the Python script to launch the Dash app.
* Access the web application through the provided local URL.

Challenges and Resolutions

* Challenge: Interfacing Python with MongoDB.
* Resolution: Utilized PyMongo and established a clear schema for data insertion and querying.
* Challenge: Real-time updating of data visualizations based on user input.
* Resolution: Leveraged Dash callbacks to refresh components upon user interaction.

A screenshot of a computer

Description automatically generated

A pie chart with numbers and a number of different colors

Description automatically generated

A screenshot of a computer

Description automatically generated

A pie chart with a pie chart in the middle

Description automatically generated

A screenshot of a computer

Description automatically generated

A map with a location pin

Description automatically generated