**CS340 Project Two: Web Application Dashboard**

Dante Trisciuzzi

6.16.2023

SNHU

Prof. Jeff Sanford

**Project Description**

This project is an interactive web-based dashboard that provides data visualization from a MongoDB database, which contains documents on animals at a shelter. The dashboard allows users to filter the data, view details in a data table, visualize data through a bar chart, and locate the animals on a map.

**Tools and Libraries**

**MongoDB** is used as the model component of the project. MongoDB, being a NoSQL database, offers flexibility and scalability and is really useful when dealing with diverse and loosely structured data, like in our case with the animal data. Its Python driver, PyMongo, provides an intuitive and straightforward API for interfacing with MongoDB, which made it the obvious choice for this project. MongoDB is available here: https://www.mongodb.com/

The **Dash** framework is used to provide the view and controller structure for this web application. Dash makes it easy to create simple interactive components for data visualization, and it works very well with data-centric Python libraries like pandas and Plotly. Dash is available here: https://plotly.com/dash/

We also used **pandas** for data manipulation, **Plotly** for data graphing, and **dash-leaflet** for the map. You can find pandas here at <https://pandas.pydata.org/>, Plotly at <https://plotly.com/python/>, and dash-leaflet at https://dash-leaflet.herokuapp.com/.

**Dashboard Steps**

1. Setup the MongoDB connection and fetch the data using PyMongo, format it with pandas.
2. Initialize the Dash application and layout for the dashboard.
3. Create callback functions in Dash to control interactivity.
4. Run the dashboard app, use debug mode to troubleshoot any issues.

**Notes**

I faced a bit of a challenge when setting up interactive filtering for the data table. I did not quite understand the way data needed to be passed to the callback function initially, and then I had issues getting the data set updated from the dataframe because of JSON formatting issues. This was sorted out by converting the frame to documents first, which I had overlooked.

**Screenshots**

