**Requirements**

Grazioso Salvare aims to identify suitable dogs for search-and-rescue training using data from animal shelters in and around Austin, Texas. This project involves the use of MongoDB databases in order to create a client-facing web dashboard for visualizing and interacting with the provided shelter data. The main requirements as specified by Grazioso Salvare are:

1. Branding and Components
   1. Include the logo with a link to snhu.com
   2. A unique identifier that includes my name, as Grazioso Salvare wants to credit me as the creator of the dashboard
2. Interactive Filtering
   1. Provide filter options for the following:
      1. Water Rescue
      2. Mountain or Wilderness Rescue
      3. Disaster Rescue or Individual Tracking
      4. Reset button to clear filters
3. Dashboard Widgets
   1. Data table: Display dynamic data based on selected filter
   2. Geolocation Chart: Map of filtered data
   3. Secondary chart: In this case a histogram

**Technical Details**

Use MongoDB for storing and querying data

Develop an intuitive and user-friendly layout

Ensure the dashboard update dynamically

Optimize the age calculations by using the age\_upon\_outcome\_in\_weeks variable from the database

**Tools & Rationale**

**MongoDB**

* Flexible Schema: MongoDB’s document-based structure allows for a flexible schema, which is helpful when dealing with varying formats from different shelters. This allows the diverse information associated with different dogs and their profiles to be easily integrated.
* Scalability: MongoDB is designed to handle large volumes of data efficiently, meaning that it will function well as the database grows as more dogs and shelters are incorporated.
* JSON-like documents: MongoDB stores data in BSON, which aligns with Python’s native data structures. This compatibility simplifies the process of interchange and manipulation between the database and the Python application

**Dash**

* Pythonic: Dash allows developers to create applications using pure Python, an advantage in this case as it leverages Python’s simplicity and readability
* Component-Based: Dash follows a component-based architecture which encourages modular development and reusable components. This means that creating interactive widgets and visualizations such as the filter options, graph, and map are easier to integrate.
* Plotly: Dash seamlessly integrates Plotly, allowing for easy creation of charts directly within the web-app

**Steps Taken**

The first step I took in order to create this web app was integrating the provided framework for the code into a Jupyter Notebook. From there, I imported the previously completed and documented CRUD module in order to parse the data properly. Once verifying that the CRUD module interacted properly with the dashboard, I began working on the formatting of the page. This included adding the Grazioso Salvare logo along with the redirect to their website onto the top of the page, as well as my identifier. From there, I worked on the bulk of the work that needed to be done, which was creating the filters for the different categories of rescue dogs, and integrating the filters into the dashboard. I created the labels and values, then assigned these filters to certain breeds. When one of these filters is chosen, the query is updated including the breeds, sex, age, and animal type. The query is then passed as a parameter in the .read() function which returns the proper data that I assigned to the records variable. I created a dataframe with this information and printed it to the dashboard. Once this was done, I added the graph to the dashboard by using Plotly create a histogram.

**Challenges**

The biggest challenges that I encountered when creating this dashboard all related to getting the proper breeds to show when executing a search. For whatever reason, the first few times I attempted to get the Mountain filter working, it would return no data. Eventually after I rewrote the blocks dedicated to that filter, I got it working, but I am still not sure exactly what was causing the code. I’m guessing it must have been syntax-related as the other filters worked fine and I quadruple-checked the breeds and their spelling. The other issue I ran into was one that I’m still not quite sure I got fixed. The Water filter is only showing a single breed, even though all breeds are accounted for in the code, and I checked it against the excel spreadsheet for spelling.