1. **Description of required functionalities with screenshots.**

In the screenshots below, each radio button display is different to demonstrate user interaction with the table, histogram, and the map. When a button is selected, it causes dash to display the corresponding table, histogram, and map that reflects the dog category the user selected.

Screenshots of Dash application.

**RESET**

A red line drawing of a person

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Water Rescue**

A red line drawing of a person

Description automatically generated

**`A screenshot of a computer

Description automatically generated**

**A screenshot of a map

Description automatically generated**

**Mountain or Wilderness Rescue**

A red line drawing of a person

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**A screenshot of a map

Description automatically generated**

**Disaster or Individual Tracking**

A red line drawing of a person

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**A screenshot of a map

Description automatically generated**

1. **Describe the tools used to achieve this functionality and a rationale for why these tools were used.**

The project uses three main tools. MongoDB, Python, and the Python framework called Dash. Some other tools used include Panda, which is a library for the Python programming language. Panda uses DataFrames (df) to organize data retrieved from our MongoDB database. As the model component of the development (MVC), MongoDB was used because of its flexibility and compatibility with JSON-like data or objects such as the one stored in our database. MongoDB stores data in the form of binary JSON called BSON. This schema (data organization) is comparatively flexible and less rigid compared to table-based structures used in traditional relational databases. One other reason for using MongoDB for storing our animals’ collection in the AAC MongoDB database is its ability to accept or insert data in the form of dictionaries. MongoDB allows for querying the database or collections and performing CRUD operations using dictionary-like queries. Its ability to accommodate unstructured data more naturally than traditional databases makes it a good choice for this project. Also, PyMongo, the official MongoDB driver for Python, provides a straightforward API for interacting with MongoDB databases. This driver is designed specifically for Python. In our project, we used dictionaries-like queries for interacting with the database.

Python Dash doubles down as both the view and controller of the project. Its vast array of features allows the developer to set up the View interface as well as the control logic of our project. Classes, dependencies, and libraries like dash\_leaflet, dcc, html, dash\_table allow us to set up the view. Dependencies like input, output and state also help us to set up control through callbacks. The framework’s compatibility with the Python language makes it an excellent choice for a project that showed data in the form of graphs and tables. Dash integrates well with Plotly, a Python graphing library. This provides powerful tools for creating interactive visualizations and data dashboards. With dash, developers can generate complex graphs, charts, and interactive plots within their web applications effortlessly. That is what informed the choice of this tool. The Python language was the language of choice in this project because of its flexibility, ease, and compatibility with MongoDB. The language is one of the most widely used in the world of technology having so many libraries and frameworks that allow you to analyze data and graphically represent them.

Below are links to the resources used in this project.

[**https://www.mongodb.com/**](https://www.mongodb.com/) **---** MongoDB and PyMongo

[**https://dash.plotly.com/**](https://dash.plotly.com/) **----** Dash and its resources

[**https://www.python.org/**](https://www.python.org/) **--** Python

[**https://pandas.pydata.org/**](https://pandas.pydata.org/) **--** Panda

<https://numpy.org/> --- Numpy Python library that supports for large, multi-dimensional arrays and matrices.

**Explain the steps that were taken to complete the project.**

The first step in completing the project was the review of the Dashboard Specifications Document. This document provided information on the different dashboard widgets to be created. Also, it provided information on the filtering of the dashboard data based on the selection of a button and the corresponding effect on the dashboard table, chart, and map. This project required the download of a logo for **Grazioso Salvare, the client. It is displayed in the dash.** Next, we created the Python code for database interactions by creating methods for CRUD operation in the Python file. This step had already been done in project one where we created the create method, the read method, the update method and the delete method. The python file has an AnimalShelter class that has properties or attributes that allow us to connect to the database. As stated, the python file, pythonDash.py, contains the create method for inserting data to our database, the read method for reading data from our database, the update method for updating our database and the delete method for deleting document(s) from our database. This file was the link between the database and the Dash file for displaying, manipulating, and controlling data using the CRUD methods as mentioned earlier. The next step was making sure that the dashboard filter options can properly retrieve data from the database and update the table and the histogram. First, we created the radio buttons for categorizing the animal data. We had water rescue button, mountain/wilderness rescue button, Disaster Rescue and Individual Tracking button and Reset button. We used the selection of these buttons to reflect our data display. This was done by **developing database queries that match the required filter functionality.** For instance, when a user selects water rescue radio button, the table, geolocation chart, and the histogram, change to reflect the data that is being retrieved from the database. This can be found in the update\_dashboard method of the callback in the dash ipynb file. As mentioned before, we modified the dashboard widgets (the table and the map) that receive input from the interactive options (radio buttons) and present those dynamic updates to the viewer or client. This step was accomplished by writing code to identify which radio button has been selected based on which to retrieve a particular kind of data from the database or collection. The retrieved data was passed into the table which updates dynamically displaying the new data. Per the instructions in the assignment, I created a histogram which also updates dynamically based on the radio button selected. This was done in the update\_graph method in the callback. No code changes were made to the update\_map method of callback.

* **Identify any challenges that were encountered and explain how those challenges were overcome.**

The project was straightforward to some extent. The only challenge I encountered during this project was the implementation of the update\_dashboard method in our callback. After researching extensively and getting a better understanding of the Dash syntax, it was easy to write the code that filters the data display based on what radio button the user selects. The challenge came when I had to figure out why my table, histogram and geolocation map where not updating when a button is selected. In other words, the dash display was not interactive. The debugging phase was my biggest challenge in spite of all the research I had done to resolve the issue. To overcome this, I employed the assistance of my professor who was kind enough to point me in the right direction. Come to find out, I was wrongly passing data to the table. After the correction, my project worked without any issues surprisingly.