# CS 340 README Template

## About the Project/Project Title

This project developed an interactive dashboard application for Grazioso Salvare, allowing their team to visualize and filter rescue animal data based on specific criteria for Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking scenarios. The app connects to a MongoDB database and provides a user-friendly interface for filtering data, viewing breed distributions, and mapping animal locations.

## Motivation

Grazioso Salvare needed a tool that enables quick identification of suitable animals for specialized rescue tasks. This dashboard streamlines that task and supports their goal of optimizing rescue training selection.

## Getting Started

* Enter the MongoDB shell using a command terminal.
* Import the csv file (aac\_shelter\_outcomes.csv)
* Provide username and password for authentication.
* User needs access to Python to run the program

## Installation

* MongoDB Access
* Python IDE to run python code
* Jupyter Notebook Access to run ipynb file
* Dash Framework to utilize features for web-based functions

## Usage

### Code

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

### Tests

Ensuring the reliability and functionality of the Grazioso Salvare Dashboard involved a comprehensive testing strategy. Below are the steps and outcomes for the test:

Filter Functionality:

* Start the dashboard by running `ProjectTwoDashboard.ipynb` in Jupyter Notebook.
* Use the radio buttons to switch between 'Water Rescue', 'Mountain or Wilderness Rescue', 'Disaster or Individual Tracking', and 'Reset'.
* Expected Outcome:
* The DataTable should reflect the correct subset of animals based on the selected rescue type. For instance, selecting 'Water Rescue' should display only Intact Female dogs of specific breeds aged between 26 and 156 weeks.
* Confirm by checking the number of entries and the data displayed in the table.

Data Visualization Testing:

Pie Chart:

* After filtering, look at the pie chart.
* Expected Outcome: The chart should update to show the breed distribution of the filtered animals, ensuring dynamic updates with each filter change.

Map:

* Steps: Examine the map for markers indicating animal locations.
* Expected Outcome: Each animal in the DataTable should have a corresponding marker on the map. Selecting an animal should highlight or provide additional info on the map.

Interactive DataTable Testing:

* Sort the table by various columns (e.g., 'name', 'breed', 'age\_upon\_outcome\_in\_weeks').
* Use the native filtering to narrow down the data.
* Click on a row to select an animal.

Expected Outcome:

Sorting should order the rows as expected.

* Filtering should reduce the visible dataset appropriately.
* Selecting a row should update the map to highlight the chosen animal.

### Screenshots

Water Rescue Filter:

A screenshot of a computer

AI-generated content may be incorrect.

Mountain Rescue Filter:

A screenshot of a map

AI-generated content may be incorrect.

Disaster Tracking Filter:

A screenshot of a map

AI-generated content may be incorrect.

Reset All Data:

A screenshot of a computer

AI-generated content may be incorrect.

**Tools Used**

* MongoDB for flexible and scalable data storage
* PyMongo for Python-MongoDB interface
* Dash (with JupyterDash) for reactive dashboard UI
* Plotly Express for pie charts
* Dash Leaflet for interactive mapping
* Pandas for data manipulation in Python

## Challenges

## Mapping markers required attention to correct structure of child components to prevent errors.

## Handling dynamic filtering while ensuring performance with large datasets.

## Learned intricacies of MongoDB querying, especially with complex breed and age filters.

## Contact

Nathan Sakhichand

Nathan.Sakhichand@snhu.edu