# CS 340 README Template

**About the Project/Project Title**

***Grazioso Salvare Animal Rescue Dashboard***

*This project was developed for Grazioso Salvare, a company that specializes in training rescue animals. With this application Grazioso Salvare will be able to identify and categorize dogs that are suitable for search and rescue training in the Austin, TX area from their animal shelter data files.*

*This dashboard application has an easy to use user interface which will help Grazioso Salvare filter and visualize his animal shelter data and to easily identify potential rescue dogs based on dog breeds and different types of rescue operations.*

## Getting Started

*To get this project working on a locally follow these steps:*

1. *Set Up MongoDB*
   * *Download and Install MongoDB*
   * *Start the MongoDB server by running ‘Mongosh’*
2. *Set Up Python and PyMongo*
   * *Download and Install Python*
   * *Install PyMongo using pip*
     1. *pip install pymongo*
3. *Set Up User Authentication* 
   * *Create an ‘aacuser’ account with readWrite permissions*
   * *Follow steps 6-7 of the* [*MongoDB Manual Enable Access Control*](https://www.mongodb.com/docs/v6.0/tutorial/enable-authentication/) *tutorial*
4. *Import Dataset*
   * *Import the ‘aac\_shelter\_outcomes.csv’ dataset into MongoDB using mongoimport*
5. *Load the ipynb file into Juypter Notebook and click run. This should give you the link to the Dash page where the project UI is active.*

## Installation

*The following are tools that need to be downloaded to use the software:*

*Python: This is the programming language for creating the CRUD module. It was chosen due to how simplistic it is to use as well as its libraries that support database operations. It also has seamless integration with MongoDB and is compatible with web frameworks like Dash.*

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

*PyMongo: PyMongo is Pythons driver for MongoDB and allows Python applications to connect with MongoDB and do CRUD operations.*

*Installation --* [*pip install pymongo*](https://pypi.org/project/pymongo/)

*Jupyter Notebook: Jupyter Notebook is used for writing and testing Python code. This was used to test if the CRUD operations were successful which was excellent for debugging.*

*Installation –* [*pip install jupyterlab*](https://jupyter.org/install)

*MongoDB: MongoDB is the backend database that is used to store and manage the data from the CSV animal shelters file. This database was chosen due to its integration with Python through pymongo as well as its JSON document structure making it align with the web.*

*Installation --* [*https://www.mongodb.com/docs/manual/installation/*](https://www.mongodb.com/docs/manual/installation/)

*Dash Framework: is a python web application framework that provides a UI and controller components for web apps. It also allows the developer to create interactive data dashboards.*

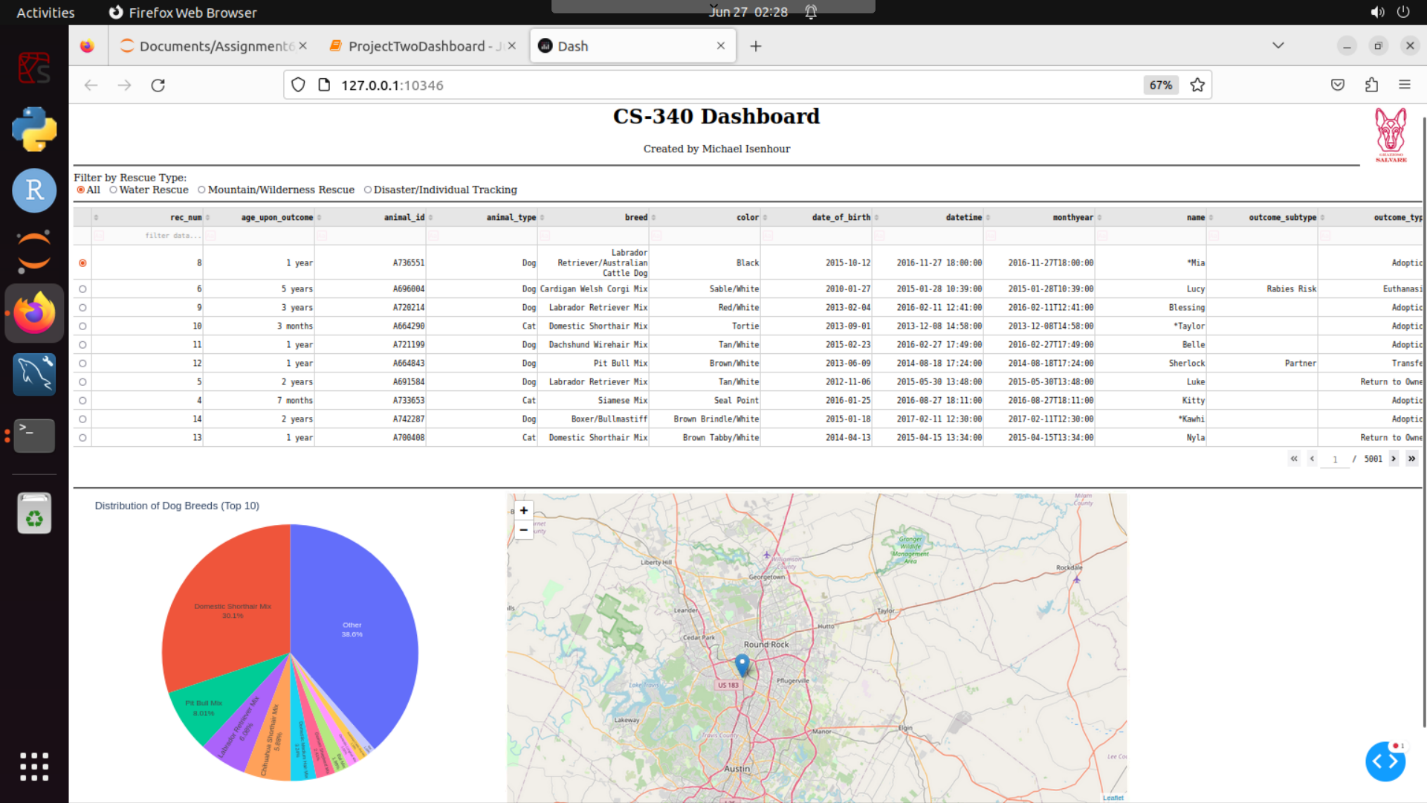
*Installation --* [*pip install dash*](https://dash.plotly.com/installation)

***UI Functionality***

*The following screenshots will show the functionality of this web application. In the web application we have the companies logo as well as my name since I am the developer. We also have interactive radio buttons that will filter the results of the database chart and graphs.*

*Figure 1 shows the Starting State of the Web Application. In this state the data is unfiltered as the radio button is in the All position allowing all data to be seen. The graph to the left shows the top 10 breeds and the rest are put in the category Other so the graph could be read easily. The geo location graph to the right is showing a marker where the first dog is located.*

Figure 1 Starting State of Web Application

**

*The figure 2 screenshot shows the filtered data when the Water Rescue button is clicked. This then updates the data allowing only the dogs who could go on Water Rescue missions to be seen. This also updates the chart on the left as well as the geo location chart showing now where the first dog of the filtered results is located.*

Figure 2 Filtering the Data by Water Rescue Type**A screenshot of a computer

Description automatically generated**

The figure 3 *screenshot shows the filtered data when the Mountain/Wilderness Rescue button is clicked. This, just like the Water Rescue buttion, updates the data allowing only the dogs who could go on the filtered mission type( in this case being Mountain/Wilderness Rescue missions) to be seen. This also updates the chart on the left as well as the geo location chart showing now where the first dog of the filtered results is located.*

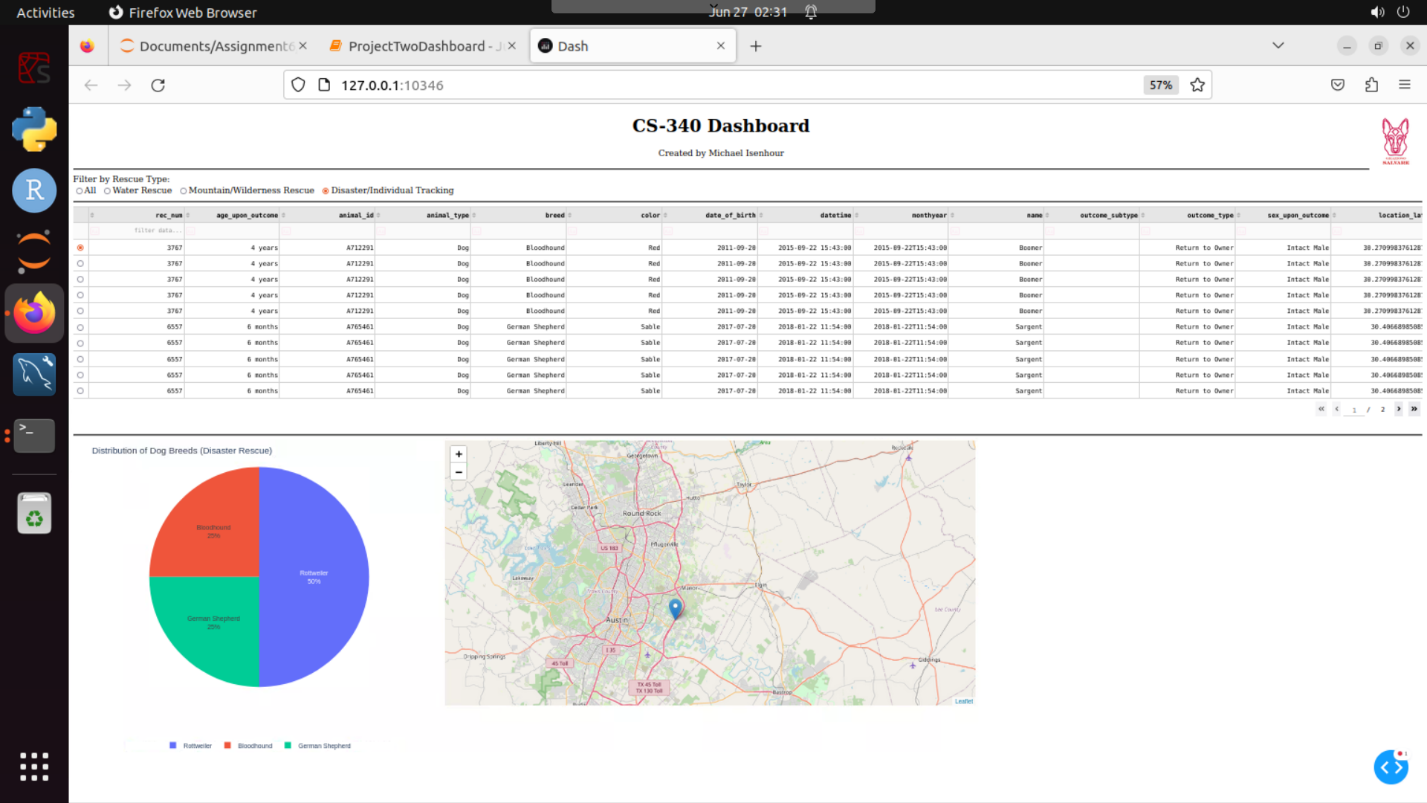
Figure 3 Filters data by Mountain/Wilderness Rescue type

*A screenshot of a computer

Description automatically generated*

The figure 4 *screenshot shows the filtered data when the Disaster/Individual Tracker button is clicked. This filters the data allowing only the dogs who could go on the Disaster missions to be seen. This also updates the chart on the left as well as the geo location chart showing now where the first dog of the filtered results is located.*

Figure 4 Filters data by Disaster/Individual Tracker Rescue type

****

**Steps Taken to Comple Project and Challenges**

When it comes to completing this project the main step was reading the documentation and really understanding what our client wanted. After loading in the clients CSV file and getting the CRUD setup complete doing the User Interface was actually somewhat easy. I added the clients logo as requested as well as made sure it had a link back to the schools website like it was written in the docs. I also implemented the data table and the two graphs.

When it comes to challenges I faced the main one is the fact that the CSV file my client gave me wasn’t set up properly by the client therefore different breeds that should be showing up under Water Rescue and Individual Tracker rescue aren’t showing up. This is because the breed wasn’t named properly so its not pulling that info when searching for it in the database. Another issue I ran into was trying to get my graph to be larger initially but as you can see from the screenshots I was able to fix that issue. Other than that that was the only real issues I ran into for this project.

## Contact

Your name: Michael Isenhour