# CS 340 README

**CJ Busca**

## About this Project

This web application's goal is to give users a simple way to search through databases of data and return optimal combinations of breed, sex, and age for training rescue dogs.

A database, an API, and a dynamic dashboard make up the project.

**Using MongoDB**

Because Mongo offers a simple setup of the database from a .csv and a Python compatible interface, it was chosen specifically for this program. Although Python may use SQL-like database tools, the syntax is so dissimilar that switching between them can be difficult. When performing the fundamental CRUD operations of a database using Python, the selection tools in a SQL database are substantially more complicated than those in a MongoDB.

**Using Dash**

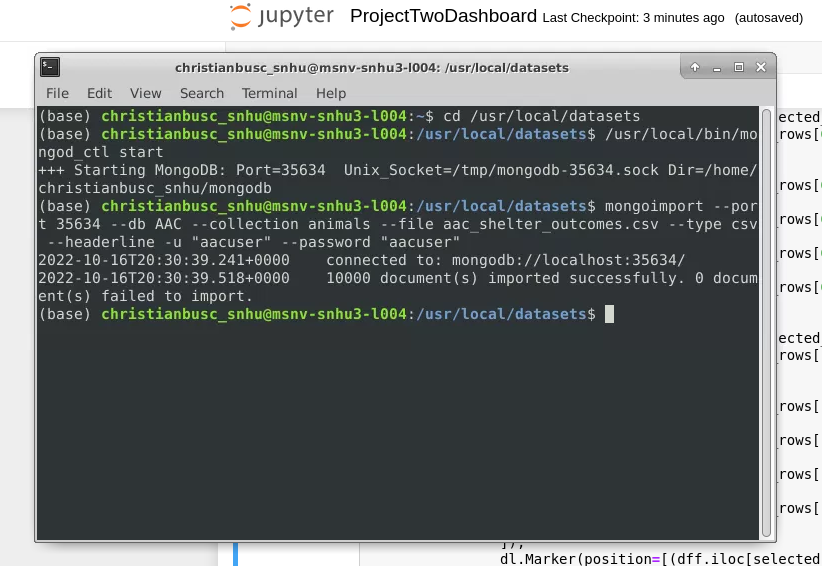
Dash was chosen as the dashboard construction tool because of how dynamic it’s features are. Dash is a program that uses react JavaScript and offers a remarkably adaptive framework. HTML Dash tags are used in Dash to manage segment outputs. Updates are then made to any target inputs defined in the app callbacks process in accordance with the Python module's program instructions.

**Getting Started**

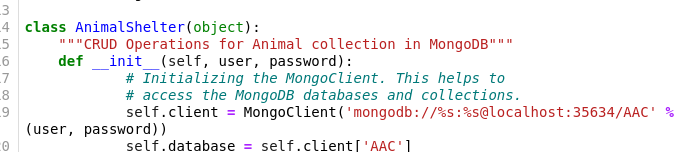
You must follow a number of procedures in order to make a local copy of this program.

Making a Mongo database is the first step. To access that database, the second step is to program a Python CRUD module. Making a Dash web application that uses the Python CRUD module is the last step. This multi-layered program responds instantly to changes in the table.

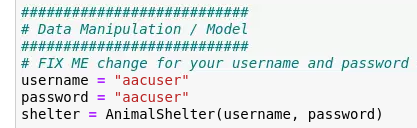
1. Establish the AAC database in a Mongo database.
2. Register a user with the priviledges to read and write to that AAC database.
3. Import the dataset from the collection animal\_shelter\_outcomes.csv.
4. Example of logging in as “aacuser” below.



1. Within Animal\_App.py, the user must change the localhost’s port to match the port MongoDB is currently serving.



1. Within the dashboard username and password must also reflect the username and password of the user who will be using the database. In this example, the username and password is “aacuser”.



1. Make sure the test data for the create function is different every time, or remove the created record in between runs, before starting the tests.
2. The desired HTML/CSS layout and the correct ids for the data frame, map, and chart should be configured in a new Dash online application dashboard.
3. To add all the data to the initial data frame, create an app callback.
4. Generate radial variations and implement database searches according to the client's preferred breed characteristics.
   1. Complex queries are required at this point in order to group numerous searches into a parent search and provide the outcomes. If you're not cautious, this can lead to a lot of syntactic issues.
5. To update the map with the initial item of a category up until the user selects an item, create an application callback. Then design a feature that ascertains the user's selection and shows that on the map.
   1. The suggested beginning point made advantage of the difficult-to-use "supplied virtual row ids" variable that comes with Dash.
   2. You will have a much simpler time integrating if you skip it and go straight to "selected rows."
6. The data that is now being presented on the screen can be used to create a pie chart.
   1. Make an application callback with the pie chart in hand that refreshes the pie chart with the data filtered from the displayed criteria.
   2. Only after the pie chart is populated, create methods that implement the functional requirement and target data.

## Installation

This will require Jupyter Notebooks, Python, and MongoDB. Installation instruction are as followed:

**Jupyter Notebooks**: Most operating systems allow Jupyter to be installed from the command line. Follow these instructions for more information on thorough instructions, for Windows, Mac, and Linux:

* <https://jupyter.org/install>
* <https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html>

**Python**: Once Python is installed, you should be able to use this software from a Mac, Linux, or Windows Command Prompt via the Terminal. Installation instructions for Python:

* https://wiki.python.org/moin/BeginnersGuide/Download

**MongoDB**: There are two editions of MongoDB: Community and Enterprise. You may find thorough directions for installing MongoDB here:

* <https://docs.mongodb.com/manual/installation/>

**Plotly**

To generate the right charts, Plotly must be installed. Plotly is a graphing tool for the Python language that can be loaded straight from your Jupyter notebook into your Python module. For more information on installing Plotly, view the following information:

* <https://plotly.com/python/getting-started/>

**Dash**

Dash is a framework for creating web applications. Using the following details, you may import the Dash Core Components into your Jupyter notebook:

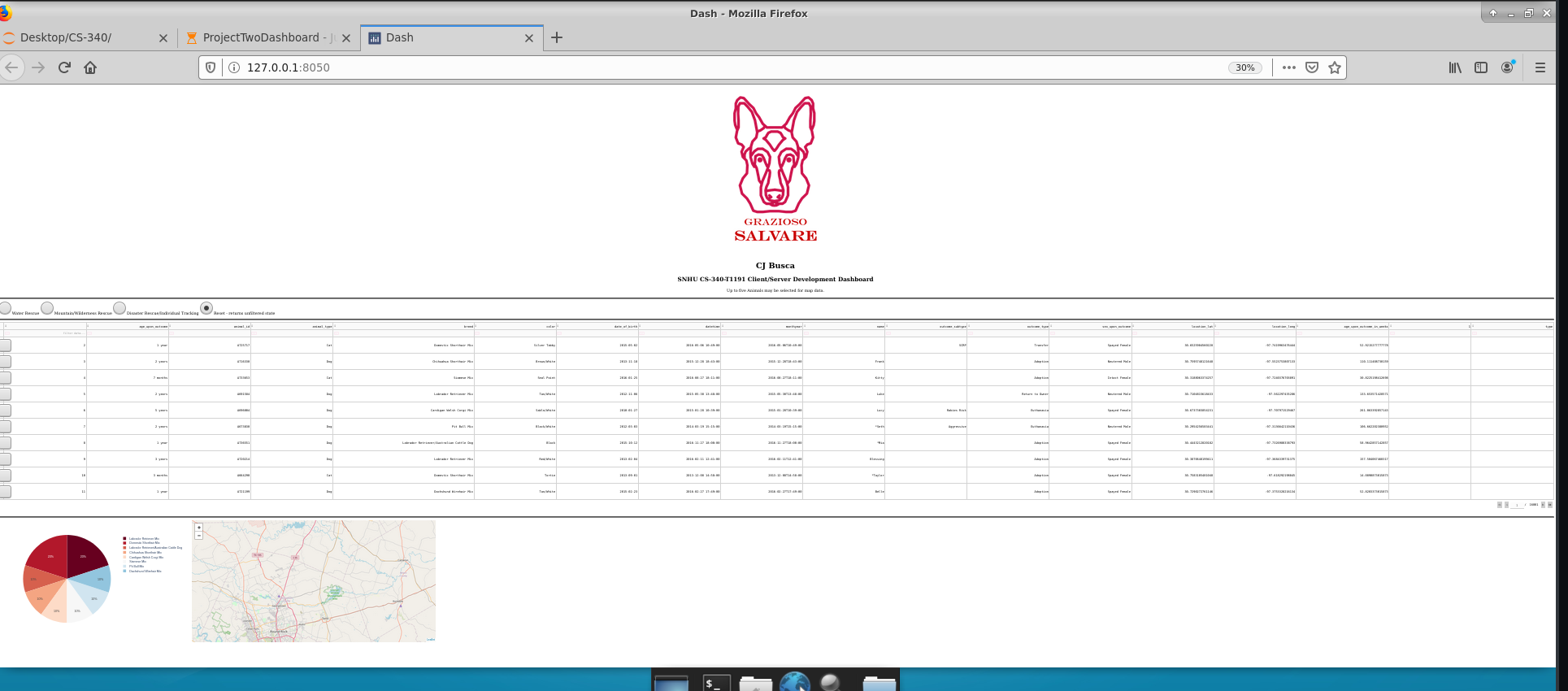
* <https://pypi.org/project/dash/>

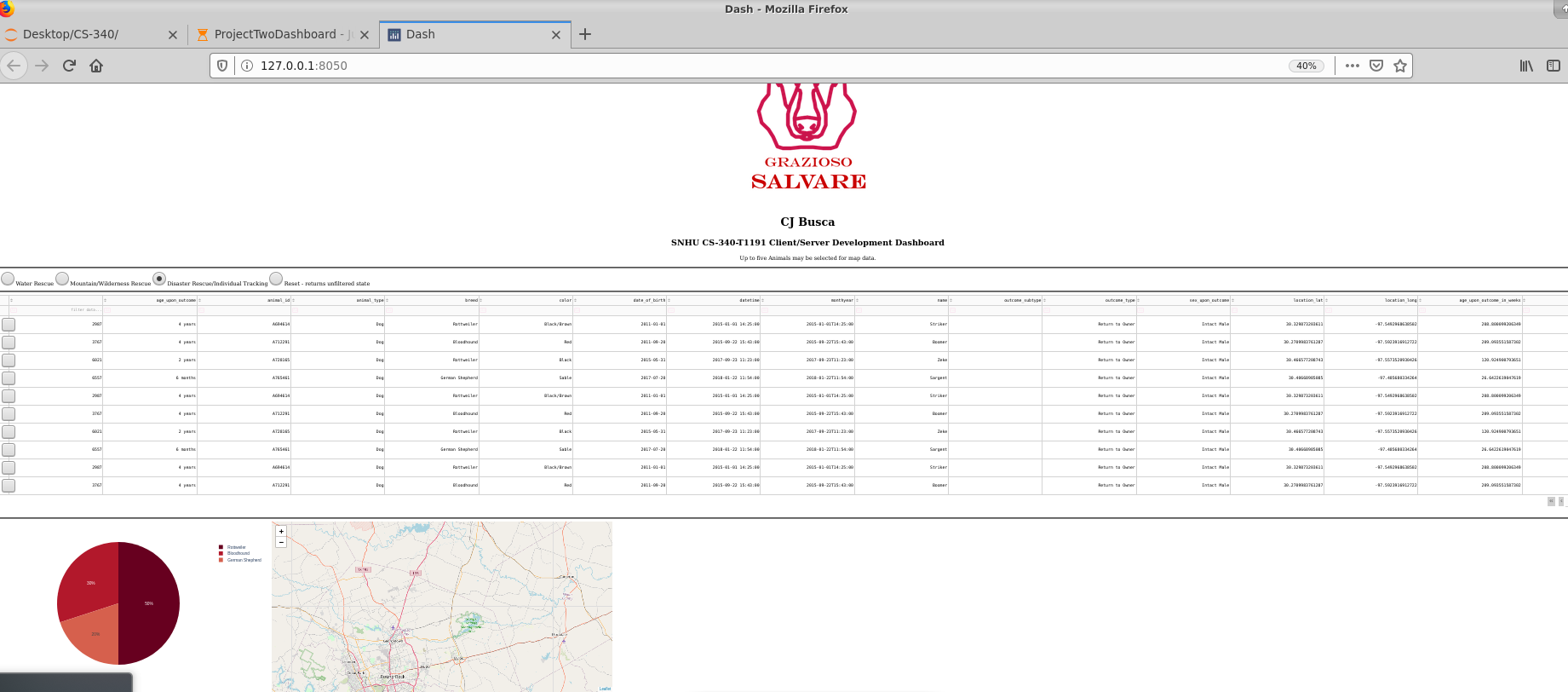
**Pandas**

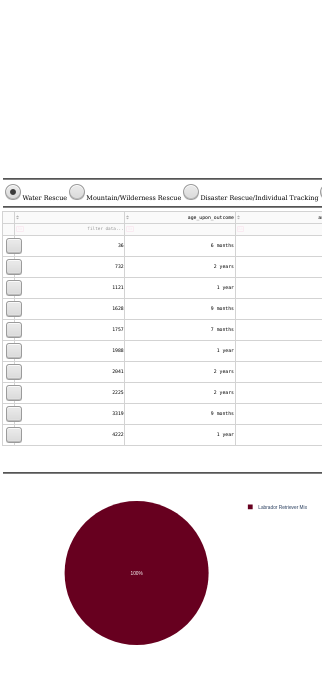
Additionally, this web application makes advantage of Pandas. The Python tool used to construct the data frames is called Pandas. Before using Pandas here, you should review any additional dependencies and information:

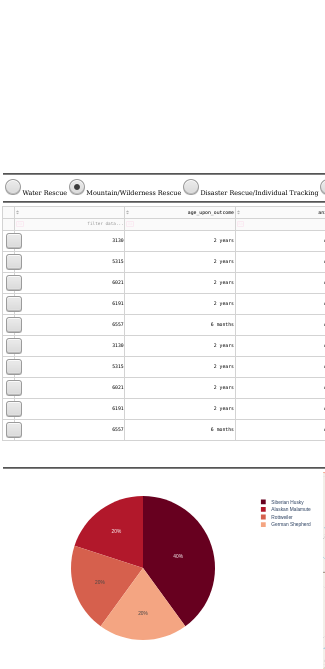
* <https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html>

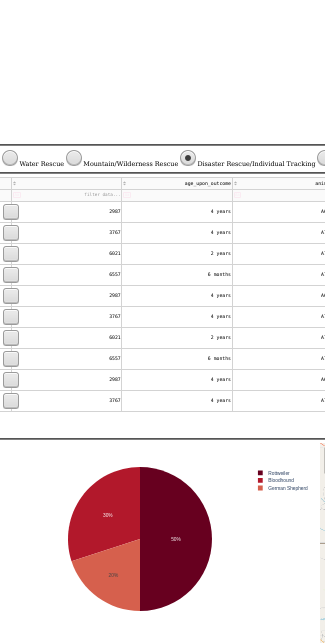
**Demonstration**











**General Challenges**

During the completion of this project, I had experienced difficulty due to time constraints and the skeleton code for this project not working initially. I was having trouble with my HTML outputs not displaying within the notebook, but after some research and some help from the discussion posts, I found a new method by utilizing the application’s Dash port and displaying the project through there. This however, caused some integration issues. Some functions are slow to process or do not load at all unless they are refreshed multiple times. I ended up developing my applications using my own CentOS server as a reference.

For the majority of this course, I was blind not being able to see the visuals from the functions I would implement within my CRUD and my ipynb. I rectified this situation by following the discussion post comments where other users were facing the same issue. I had also experienced weird bugs within my environment where my packages for Dash were deprecated, which led me down another rabbithole of troubleshooting.