# CS 340 README Template

*Use this template to complete your README file. When completing the template, keep the headings as they are so that your document has a clear organization. Remove the italicized prompt text after you have completed each section for a polished final document.*

## About the Project/Project Title

This application imports a cvs file into a mongo database that allows the user to view a list of various animals located at an animal shelter. The application is used to help the user filter the various animals into categories that will help them find dogs for working purposes. The use will be able to filter the dogs based on what type of rescue mission they need.

## Motivation

Animals are an important part of the search and rescue divisions. Dogs in particular are used in many different areas. From police to fire and even disaster relief, dogs are one of the primary resources for a rescue team. Finding the right dogs for the job is very important for these different teams. Shelter dogs could be given a new home, and a new job with this application.

## Getting Started

A user will need to be able to run the Mongo Shell in a Linux Terminal. In addition to this they will need to import the “aac\_shelter\_outcomes.csv” into the Mongo Shell as well. The user must also have the proper credentials to modify the mongo database associated with AAC. Jypeter Notebook is required as well as

**Why We Used MongoDB, Python and Dash**

Python is a widely used and trusted programing language. It is simple to implement, and works will with Mongo. Additionally, pythons object-oriented programing makes this program modular. This means that this can later be easily modified to accommodate different search criteria. The application uses the Model-View-Controller (MVC) architectural pattern. This allows us to use python to regulate the controller that will display the Dash application. Modifications can easily be made within the Python code. The data model exists in mongoDB. MongoDB allows the document to structure to be less restrictive. This will allow easier modification for the end use without having to jump through too many hoops. With all of these together, making a scalable and modular application becomes much easier for the user.

## STEPS FOR COMPLETION

The first step was to create login credentials within mongo to access and modify the AAC database. Additionally, importing the CSV file into the Mongo Database will store the data we will need for later. From here we then moved over to Jupyter Notebook to start the creation of the CRUD file that will be used to make the modifications through PyMongo. Development of this code is important to the functionality of the Dash app. Once working code is developed, we them start working on the Dash application. This is where the application comes to life. Importing the CRUD file into this code allows for the program to run with the proper authentications. We need to also make sure to code in the username and password for the AAC database. The we work on coding in the various widgets that will be used in the final Dash application. A drop-down menu was used to filter the different rescue missions. A map and location information is stored to show the location of the various animals. In addition, a pie chart is used to show the percentage of different breeds of dogs is used for quick reference.

## Installation

The user will need a Linux devise.

Current version of MongoDB - <https://www.mongodb.com/docs/manual/administration/install-on-linux/>

Files were created and modified in Jupyter Notebook - <https://jupyter.org/install>

**A picture containing application

Description automatically generated**

## Dash application Overview / Reset

Table

Description automatically generated

**Drop-Down Menu**

**Graphical user interface, application

Description automatically generated**

**Mountain or Wilderness**

**Graphical user interface, application

Description automatically generated**

**Water**

**Graphical user interface, application

Description automatically generated**

**Disaster or Individual tracking**

**Graphical user interface, application

Description automatically generated**

### Code Example

**Dashboard Layout with Logo and Dropdown**

Text

Description automatically generated

**Water Filter**

Text

Description automatically generated

**Pie Chart**

Text

Description automatically generated with low confidence

**Map**

Graphical user interface, text

Description automatically generated

**Dash Framework -** <https://www.tutorialspoint.com/python_web_development_libraries/python_web_development_libraries_dash_framework.htm>

<https://dash.plotly.com/datatable>

<https://dash.plotly.com/dash-core-components>

<https://plotly.com/python/pie-charts/>

<https://dash-leaflet.herokuapp.com/>

**MVC Framework – Introduction -** <https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>

<https://www.geeksforgeeks.org/mvc-design-pattern/>

## PyMongo –

<https://pymongo.readthedocs.io/en/stable/>

## Contact

Dane.Clark1@snhu.edu