# CS 340 README

# Project Two

Valerie J. Smith

August 12, 2022

**Functionality**

This project is to introduce the CRUD (Create, Read, Update, and Delete) functions for a data set imported to a mongo database*.* The database used contains a collection of animals that was created in a mongo database via an import of a .csv file. The mongo database contains an administrator and a user account. Connections and CRUD functionality to the mongo database are made possible with the use of a python module, further allowing the functionality of the dashboard.

This project contains a dashboard created in Mongo Client with PyMongo. The dashboard represents a request from the fictional client, Grazioso Salvare, who wants a functional dashboard to display numerous aspects of his Animal Shelter data. The dashboard imports a Python module that contains the database connection details as well as the CRUD functionality.

The features of the dashboard include a data table, a location map, and a pie chart that will populate with the data that is chosen by the user using the radio button choices named and selected by the client. The data table also features pagination to provide a cleaner appearance for the client, limiting the rows to 10 at a time. The location map will show an icon for each dog in the visible rows and provide a tooltip with the dog’s name and breed, depending on the selection of the user.

The client specified the ability to show the animals in his database that are Water Rescue Breeds, which include Labrador Retriever Mix, Chesapeake Bay Retriever, and Newfoundland breeds. The preferences are Intact Females that are 26 to 156 weeks in training age. This functionality is represented by the following screenshot:

Chart

Description automatically generated

The client specified the ability to show the animals in his database that are Mountain or Wilderness Rescue, which include German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, and Rottweiler breeds. The preferences are Intact Males that are 26 to 156 weeks in training age. This functionality is represented by the following screenshot:

Graphical user interface, chart, application

Description automatically generated

The client specified the ability to show the animals in his database that are Disaster or Individual Tracking, which include Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, and Rottweiler breeds. The preferences are Intact Males that are 20 to 300 weeks in training age. This functionality is represented by the following screenshot:

Graphical user interface, chart, application

Description automatically generated

The client requested the ability to reset the dashboard to the original unfiltered state. The Reset selection will show the data in the format requested. The following screenshot demonstrates this functionality:

A picture containing application

Description automatically generated

**Tools**

The following tools were used to create the dashboard:

1. Using MongoDB shell version v4.2.6

Go to the download page at https://www.mongodb.com/try/download/community

Choose your OS and your desired MongoDB version.

Click Download.

Go to your ‘Downloads’ folder.

Click on the installer.

Follow the instructions.

1. Python v3.6

<https://www.howtogeek.com/197947/how-to-install-python-on-windows/>

1. Jupyter Notebook v6.0.1 (optional)

<https://jupyter.org/install>

1. Dash, Dash Leaflet, Dash Core Components, Dash HTML Components, Plotly Express, and various Dash dependencies available at: https://plotly.com/
2. PyMongo: <https://pymongo.readthedocs.io/en/stable/index.html>

**Rationale**

The dashboard was created with a Python module that contains the database connection functionality as well as the implementation of the CRUD functions. Python was chosen as the language that will author the script to provide the “glue” to bridge the dashboard to the database. PyMongo was chosen to access the MongoClient, which imports the Python module to allow the Dashboard to interact with the data in the database. The Dash components were chosen to provide the visuals requested by the user and allow interaction between the user and the data. The combination reflects the Model-View-Controller aspect of Object-Oriented Programming.

**Steps for Project Completion**

1. The Data provided by the client was imported to a Mongo Database.
2. An administrator and a user account were created to access the database and to test that the sign on functionality from the python module was successful, with the test user having readWrite access only.
3. The Python module was created to provide the database access functionality as well as the CRUD functions.
4. The dashboard was designed and implemented according to the specifications of the client and features the interactivity with the data that was requested by importing the Python module to assist in the data sorting requested. The client provided a company logo to feature on the dashboard.

**Challenges**

The challenges presented in this project were overcome by utilizing the reference material provided by the tools that are listed previously. Importing the data into the mongo database was straightforward, although creating the Administrator and the user with only read-write access required a good amount of testing to ensure that the correct capabilities were assigned.

The next major challenge was to create the Python script, or driver, which contains the commands to access the database as well as the CRUD functions. It was fruitful to test the functionality of the python driver separately and with test data before attempting to connect the dashboard.

The dashboard components represented here feature a component with an id and various options. The components are then referred to in a callback function, which is coded to communicate with the component. Once an understanding of how to pass the various id’s, inputs, and outputs became apparent, it became clearer of how to implement the dashboard according to the client’s specifications.