# CS 340 README Template

## About the Project/Project Title

Second project for CS 340 Client/Server Development. Goal was to create a software application that can work with existing data from animal shelters, contained in a Mongo database, in order to categorize available dogs. This portion of the project focused on development of web application dash board that leverages the python module I developed in the previous project that allows access to CRUD functionality on the Mongo Database.

## Motivation

The theoretical company, Grazioso Salvare, wants software that will ease its search for dogs that can be trained for various rescue and other roles, based on those dogs matching certain profile criteria that they believe make those dogs a good match to those roles. We are leveraging Dash in development of the client facing webpage for the ease with which it incorporates Python, the language that we have chosen to utilize when creating a module with functions that our dashboard will call in order to access and retrieve data from our Mongo database. Mongo itself is a NoSQL database program that has proven to be very user friendly to retrieve data from with our python function calls compared to some of my previous experiences retrieving data from databases utilizing external calls. As a side note, our python module that we utilize to access the Mongo database has been developed with functions for creating, updating and deleting documents in Mongo during our previous project, as well as the read functionality that we are utilizing for this project, although the theoretical company made no requests that would require the additional functionality at this time. In short, the goal here is to provide a web based dashboard that can return a table of information on dogs based on sets of pre-provided search criteria, geolocate selected dogs on a map, and provide a pie chart showing the percentage by breed of dogs shown in the data table.

## Getting Started/Installation

In order to utilize this project, you will first need an instance of Mongo database set up and running. The database will need to have the ‘AAC’ database and have imported the appropriate animal data to the animal shelter collection. Said database will also need to be set up with a user that has access to this collection and the relevant username and password combo will need to be changed in the ProjectTwoDashboard.ipynb file (alternatively additional work can be done to require users to log in with their own user and password on dashboard startup to improve security and I would recommend doing so before using this code in the wild).

You’ll need to then install the Dash framework and Jupyter labs (and of course will need python installed). At this point Jupyter can be used to load and run the ProjectTwoDashboard.ipynb file, which will start the web dashboard. Note…if your set up involves an alternative port number for the Mongo database, or alternative database name, those will need to be changed first in the .ipynb file in the instantiation of the object from the MongoCreateRead.py file which takes both as parameters. If your collection is a different name, you will need to update the parameter in the readQuery() function calls in the .ipynb file.

Note also that both the ProjectTwoDashboard.ipynb file and the MongoCreateRead.py file that was developed earlier in this course are also required in order to utilize this project.

**Tools Documentation:**

Tools used in development, usage, and testing of this project included:

Mongo (Not requiring a schema for its data storage and storing data in collections of documents in a JSON like manner makes data easy for Python to retrieve and store in dictionaries without additional manipulation to get the returned data into a usable format). (https://www.mongodb.com/)

Apporto (https://www.apporto.com/)

Dash (An open\_source Python framework for building web applications. Given that it is a python framework and designed for web applications, it works well with the python code we are utilizing to access the Mongo DB and has the capabilities we need to create a web page application). (https://github.com/plotly/dash)

Jupyter

PyCharm (including the pymongo library) during development of the MongoCreateRead.py module. (https://www.jetbrains.com/pycharm/)

MongoCreateRead.py

## Usage

Once you have started up the ProjectTwoDashboard.ipynb file using Jupyter, it should load a dashboard that begins with the logo and company name for Grazioso Salvare at the top of the dashboard, Scrolling down will bring the user to the data table where animal information pertinent to the currently chosen search criteria is displayed after it has been retrieved by a readQuery() function call to the collection in the Mongo database. Note that the selection criteria can be set by selecting one of the four radio items at the top of the table, and that on start up this selection defaults to the reset option. It is worth noting that when no other criteria is selected, all animals appear in the table, not only dogs. If the client would prefer not to display cats or other animals even when no selection is made, the code could be changed for this setting to make a readQuery() call with {“animal\_type”: “Dog”} as a parameter rathe than the currently empty search criteria that is being passed in. It should also be noted that an instance of our connection to the Mongo database that we are using was instantiated on startup of the dashboard, with parameters for port, database name, user id and password, none of which is displayed to the user. Future development may wish to modify this to require users to provide login information on start up to avoid hard coding this information to improve security.

Once an instance of the AnimalShelter class from MongoCreateRead has been instantiated, you should be able to call the create(), read(), update() and delete() functions (CRUD) on the instance to find, insert, update, or delete from the mongo database collection respectively. In the case of read() it takes parameters for the collection to run find against as well as a key and a value parameter to use in the find query on the mongo database collection. The create() function takes as parameters the collection to insert into as well as the data to be inserted. The update() function takes the collection parameter along with the key and value to use in finding the documents to update and a data parameter for what to update them to be. Finally the delete() function takes a collection parameter, as the others did, along with a key and value to know what documents to delete.

The user may click on any of the other three radio items that are labeled with their criteria type (exact specifications of what breeds, animal type, etc that selection maps to is not displayed to the user but the parameters sent to the readQuery() call in the code are determined by the user selection at this point). Selecting one any of the options causes the table to update after a moments time.

Note that the table displays a limited number of the returned animals from the readQuery() function and buttons at the bottom of the table can be used to progress forward and back through the available selection.

Scrolling further down the user will find a pie chart next to a geolocation map. The pie chart will display the percentage of each type of breed based on the search selection currently active for the data table. The geolocation map will show a marker at the location of the animal currently selected in the table and the user can change that selection in the table. The marker displays the breed of the animal on mouse over and additional information in a pop up if clicked on.

Finally at the very bottom of the dashboard is the unique identifier with the developer’s information (my name.

**Difficulties encountered:**

Largest difficulty faced was in the prior module where I initially learned to utilize Dash along with my prior development python module and connect to the Mongo DB. I was initially attempting to do so with all files on the apporto virtual lab desktop and was not having data returned. I tried troubleshooting with a version of the read query that accepted a key word and value as two separate parameter and then used those to create a query in the python module before calling the find() function for the Mongo database and oddly that returned me a data table that I could view. All other attempts, including creating a read function that took multiple keys and values and then formed them into a larger query in the python module all failed to return from the database call. (I attempted to add in some print statements to determine if it was ever getting back but it did not appear to be). There was a large thread in our general locations on the subject and it seemed like a number of other students were encountering it. Given that another student who had the issue had successfully got the same code to work when using a local environment, I wondered if maybe the permission of the apporto virtual lab desktop might be off somehow. I tried moving my .ipynb and .py file to the local user folder on the apporto virtual lab instance and running it again and it ran fine from there. However I still don’t know the exact issue with running it from the apporto virtual lab desktop.

The other major difficulty I have faced is when I exit out of apporto virtual lab without shutting down my Mongo database and return later I often log back in only to find that it still has my database claiming to be running but refusing all form of connection. My attempts to shut it down, connect to it or kill it at that point have all met with failure and my lack of familiarity with apporto has made it difficult to figure out how to fully shut down my virtual lab instance to the point where the Mongo database will also be shut down when I log back in. This forced me to get in the habit of making sure I always shut down my Mongo database before exiting apporto, but sometimes when I’m looking away for a bit and the virtual lab times out on its own I’d still come back to this situation.

It has been a while since I’ve done anything with a webpage and it took me a bit to remember how the various html tags and style options and the like work. It’s something I used to know at least competently and enough of it came back while I needed it for this project but it was a little frustrating to know at one point it would have been easy for me to really add a lot of extras to this project. We didn’t spend enough time on it for me to regain my fluency during the course, but I suspect the web development formatting was not the most central goal of the course.

### Screenshots

Screenshots of the project two dashboard and components. NOTE: my unique identifier is at the end and the logo is at the start, plus a data table, plus a side by side pie chart and geolocation map so all those things do not fit on screen at the same time.

First screenshot is the logo at the top, which takes up most of the screen:

Graphical user interface, text, application

Description automatically generated

Next up we have the initial state of the data table and filter selection, followed by the initial pie chart and the initial geo map, which are side by side but are too large to fit in the output area visible at the same time (Unique identifier appears under the pie chart):

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Excel

Description automatically generated

Graphical user interface, application, map

Description automatically generated

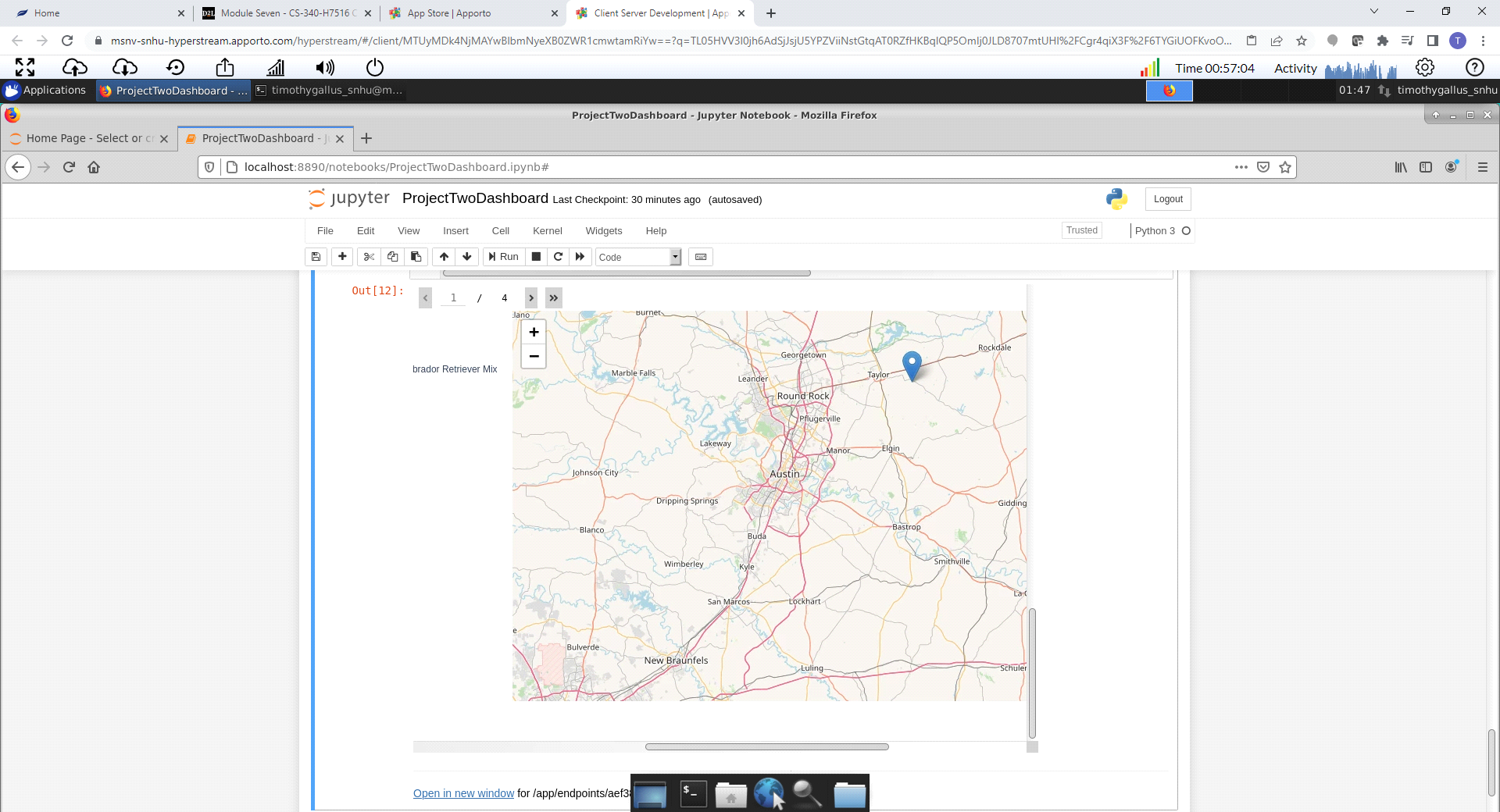
Next Screenshots are Similar to the above 3, but with Water Rescue filter selected:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated



Next up are the 3 screenshots for when Disaster or Individual Training selector is selected:

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application, map

Description automatically generated

The next 3 are for the Disaster or Individual training filter selection:

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Excel

Description automatically generated

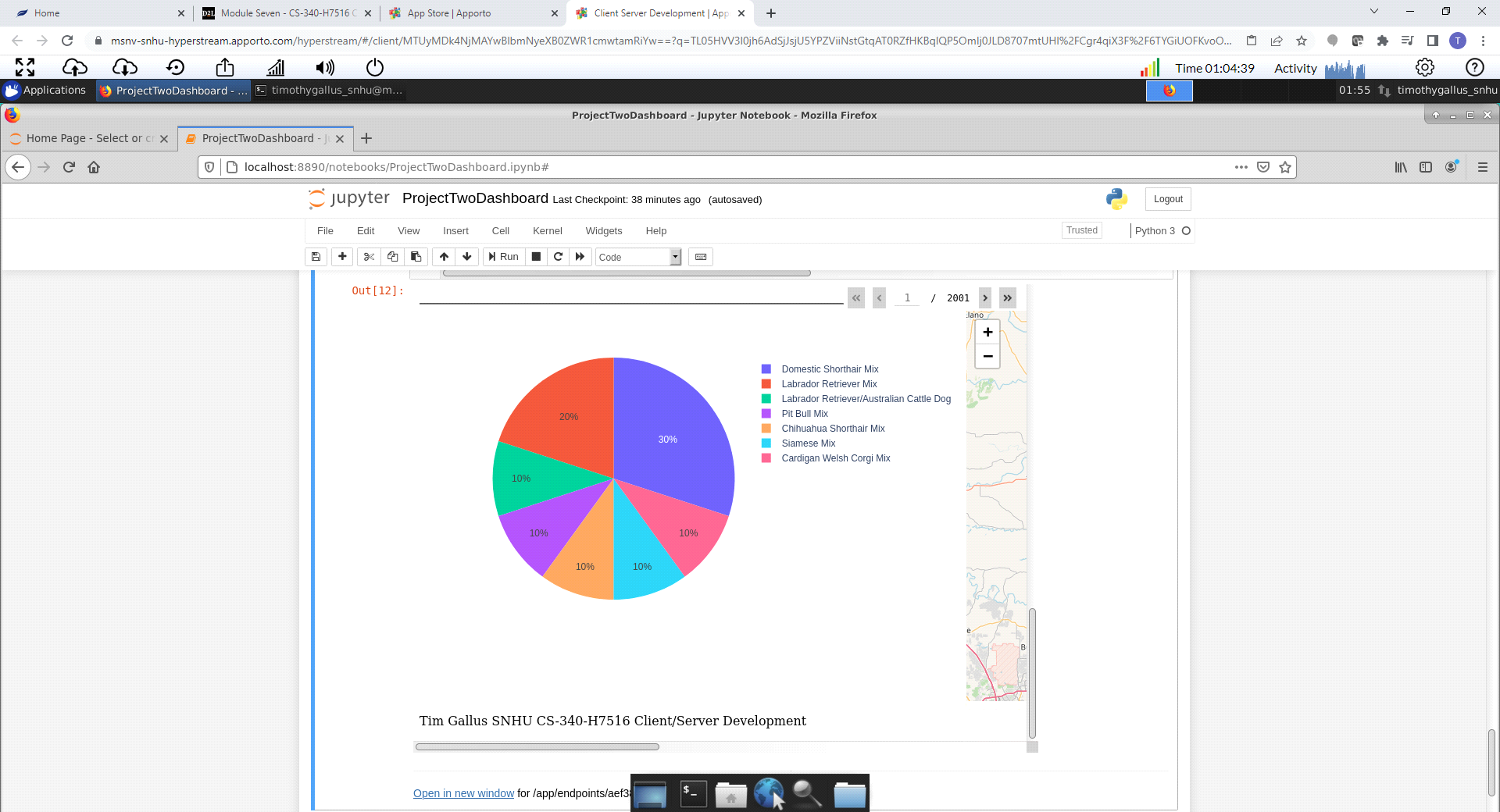
Graphical user interface, application

Description automatically generated

Final 3 are with the reset filter option selected and roughly match the initial state. Note that I moused over the marker in this one and it is the location of a cat. It may be worth bringing up to those interested in this product and seeing if we want to limit this to an {"animal\_type": "Dog"} query to avoid other animals appearing, although maybe there are future searches where non-dogs are desired:

Graphical user interface, text, application

Description automatically generated



Graphical user interface, application

Description automatically generated

## Contact

Tim Gallus