CS 340 README – Alis Gurung

About the Project/Project Title

The purpose of this web application is to provide a user-friendly interface to sift through data stored in a database, powered by Mongo DB, to return desirable combinations of breed, sex, and age to training rescue dogs.

**Motivation for Using Mongo DB**

MongoDB was used with this program because of its quick setup and the dynamic nature of MongoDB schema is useful because most of the data that is being generated by internet applications and IoT devices are non-structured which cannot be saved in a traditional SQL database. Additionally, many companies will store data before they know how it will be used later

**Motivation for Using Dash**

Dash allows to build the dashboard and use its dynamic nature which is build on ReactJS. Dash involves html Dash tags that control outputs to segments. Then, updates to any of the target inputs specified in the app callbacks process based on instructions programmed in the Python module.

**Getting Started**

To create a local copy of this program, you will need several steps. The first is to create a Mongo Database. The second is to program a Python CRUD module to access that database. The final step is to create a Dash web application that implements the Python CRUD module. This is a multi-layered application that reacts to changes in the table in real time.

1. Create a Mongo Database and create a database called AAC.

2. Create a user with read/write privileges to that AAC database.

3. Import the data from aac\_shelter\_outcomes.csv file

4. Develop the CRUD module to use python module to display data

4. To run the tests, add the test code to a Jupyter notebook and ensure that the test data for the create function is different each time or delete the added record between tests.

5. Create a new Dash web application dashboard and configure with the desired HTML/CSS layout and appropriate ids for the data frame, map, and chart.

## Installation

The tools you will need to run this include Jupyter Notebooks, Python for command line, and MongoDB. The installation of each is detailed in labeled sections right below this line.

**Jupyter Notebooks**: Jupyter can be installed from the command line in any major operating system using the simple instructions here: <https://jupyter.org/install>. For detailed instructions like Proxy servers for Windows, Mac, and Linux, follow the instructions here: <https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html>

**Python**: Detailed installation instructions for Python are available here: <https://realpython.com/installing-python/>. Once you have Python installed, you should be able to use this program from the Terminal on Mac or Linux or from the Command Prompt for Windows.

**MongoDB**: MongoDB comes in Community or Enterprise editions. Detailed instructions for the installation and downloading of MongoDB are available here: <https://docs.mongodb.com/manual/installation/>.

**Plotly**

Plotly must be imported in order to generate the proper charts. Plotly is a charting tool for Python applications and can be imported directly into your Python module from your Jupyter notebook.

**Dash**

Dash is a framework used to build web applications. You can import the Dash Core Components into your Jupyter notebook and you can install Dash using the following information: <https://pypi.org/project/dash/>

**Pandas**

Pandas is used in this web application as well. Pandas is a tool for Python that creates the data frames. Pandas has other dependencies and information that should be reviewed before use here: <https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html>

**Usage**

This application currently has 3 main functions. The first uses radial buttons to sort data based on breed, age, and sex characteristics for the different types of rescue dogs desired by Grazioso Salvare. Clicking any of the radial buttons runs the database queries and returns an updated data frame with the desired results. The final radial button titled “Reset” reverts the table back to the un-filtered state. The second function is the map and dynamic map updates. This starts the map with a marker at the position of the first item listed in the data frame. Once the user selects a row (or, up to five at a time), the map changes the marker (or adds markers for multiple selections) to the selected row(s). The third function is the dynamic pie chart. This simply sorts the data by breed and outputs a pie chart of the current data frame (not the full database) based on breed of animals shown.

This application currently has four functions. The first two obtain data from the user on what to enter into the database (create) or what to search for in the database (read). The second two functions actually run the creation and the search. The obtain functions ask the user for specific input (for the read data a key is requested and a value to run the search, while in the create function the keys are provided and only the values are requested). The create function works by passing a Python dictionary into the MongoDB as a new item and returning Boolean True if it is successful or False if it is not. The read function returns False if nothing is available or it returns the data if matching data is found.

**General View**

A picture containing text

Description automatically generated

**Water Rescue**

**Graphical user interface, chart

Description automatically generated**

**Mountain Rescue**Graphical user interface, chart, application

Description automatically generated

**Disaster Rescue**

Graphical user interface

Description automatically generated

**Known issues (CRUD and Tests):**

The big known issue at this time is that when you run the unit tests, they can only be run once without getting an error on the create function. I have no idea why that is at this time because I can manually run the create function all day with the exact same data, but only once via the test. If I run the test a second time *after* accessing the database via the terminal and deleting the item I have added, then it works again. I think it is trying to verify the new addition, but I don’t see how that matters to the test since the create function itself is capable of adding the same item again and again and I have tested this. The update tests, as a note, will also only work one time because once names are updated, if you do not provide unique data, then they will not be updated again.

**Known issues (Dashboard):**

I had a hard time making the Dashboard able to show multiple selected rows at once. I rectified this in a clunky way by having several, manually-coded, output statements for options of 1-5 for brevity. I did not want to keep going to 10 or higher because of the huge number of lines that would make in the code. I would like to find a cleaner and more concise way of coding this. The app also seems to slow down if you do too much. Lastly, if you do not de-select an item when you switch filters, then the selection of items does not work again to update the map. I looked for ways to fix this, but it is a bug persistent in the final version.

Contact

Alis Gurung ([hari.gurung@snhu.edu](mailto:hari.gurung@snhu.edu))