Southern New Hampshire University

CS 340 Client/Server Development

Professor Tad Kellogg

Student: Tyanna Prince

10/21/2022

# CS 340 README

## About the Project/Project Title

This project is a web application that provides a user-friendly interface that allows the user to browse through data that is stored inside a database. For this project we used an animal shelter search and rescue database with a list of dogs and cats with information that can return a user’s combinations of searches between breed, age, and sex of the animal. The web application is powered by MongoDB and consists of a dynamic dashboard, API, and a database.

**Motivation for Using Mongo DB**

Mongo provides a simple and quick setup of the database used for this project which was in csv format, and it also has a Python friendly interface. Python has the ability to use database tools however, the syntax behind it differs greatly and it can be a hassle switching from one to the other.

**Motivation for Using Dash**

Dash is a tool that is used when building dashboards and is very dynamic. Dash has a quick responsive framework and is a react JavaScript based tool which involves html “Dash Tags”, that control outputs and updates to any target input that has been specified within the app callbacks.

**Getting Started**

There are several steps needed for this project:

1. Create a Mongo Database and create a database called AAC.
2. Create a user with read/write privileges to that AAC database.

Text

Description automatically generated

1. Import the data from aac\_shelter\_outcomes.csv file
2. I imported the AAC csv file by running Mongo via -noauth
3. Update the port number (Mongo gives you the port number when you start the service) on the file import.
4. Example import:

Text

Description automatically generated

1. Login as “admin”

Text

Description automatically generated

1. Update the port number on localhost of the AnimalShelter.py Python code.
2. Update the “aacuser” and “Password” to the username and password you created.
3. Sample Initialization:

Text

Description automatically generated with medium confidence

1. add test code to a Jupyter Notebook and ensure that the test data for the create function is updated each time.
2. Create a new Dash dashboard and configure with the desired layout and appropriate ids for the data frame, map, and chart.
3. Create an app callback to populate the initial data frame with all of the data.
4. Create radial options and program the database queries based on client specifications.

i. complex queries are needed to combine many searches into a parent search and return the results.

1. Create an application callback to update the map with the first item of a given category until the user selects an item. Then, create functionality that determines the user selection and displays it on the map instead.
2. Create a chart of your choice from the displayed data on the screen, I chose a pie chart. Then create an application callback that updates the pie chart with the data that is filtered from the display filters.
3. Once you have the pie chart populating, develop methods that implement the functionality that you are looking for and the correct target data (I used ‘breed’).

## Installation

The tools you will need to run this include Jupyter Notebooks, Python for command line, and MongoDB.

**Jupyter Notebooks**: Jupyter can be installed from the command line in any major operating system using the simple instructions below:

<https://jupyter.org/install>.

**Python**: Detailed installation instructions for Python are available below: <https://realpython.com/installing-python/>.

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

**Plotly:**  Plotly is a charting tool for Python applications and can be imported directly into your Python module from your Jupyter notebook. see the documentation below: <https://www.journaldev.com/19692/python-plotly-tutorial#:~:text=Installation.%20To%20install%20plotly%2C%20open%20a%20terminal%20window,to%20install%20to%20collect%20dependencies%20and%20download%20them%3A>

**Dash**

Dash is a framework used to build web applications. You can import the Dash Core Components into your Jupyter notebook by using the following information below:

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

**Pandas:**. Pandas is a tool for Python that creates the data frames. Pandas has other dependencies and information that should be reviewed. Use the link below to review:

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

## Usage

There are three main functions in this application. The first function makes use of radical buttons in order to sort through data based on sex, breed, and age for the specific types of rescue animals specified by Grazioso Salvare. By clicking any of these buttons you will run the database queries and will be returned updated data based on your query. The last radical button or Reset button, reverts the table back to its original state. The second function would be the map which includes any dynamic map updates. This function will start the map with a marker placed at the position of the first item (animal) in the data frame. After the user selects any row, the map will change the marker location to that of the selected row. The third and final function is the pie chart which sorts the data by breed and shows the data in a pie chart of the current data frame. The pie chard does not show the full database but rather the breed of animals shown. You can sift through from the drop-down box to select between All (all items in database), water rescue, mountain or wilderness rescue, and disaster/individual tracking. See screen shots below for examples:

Graphical user interface, text, application, email

Description automatically generated

ALL

A picture containing diagram

Description automatically generated

Water Rescue

Text

Description automatically generated

Mountain or Wilderness Rescue

Graphical user interface, text, application

Description automatically generated

Disaster/Individual Tracking

Graphical user interface, text, application

Description automatically generated

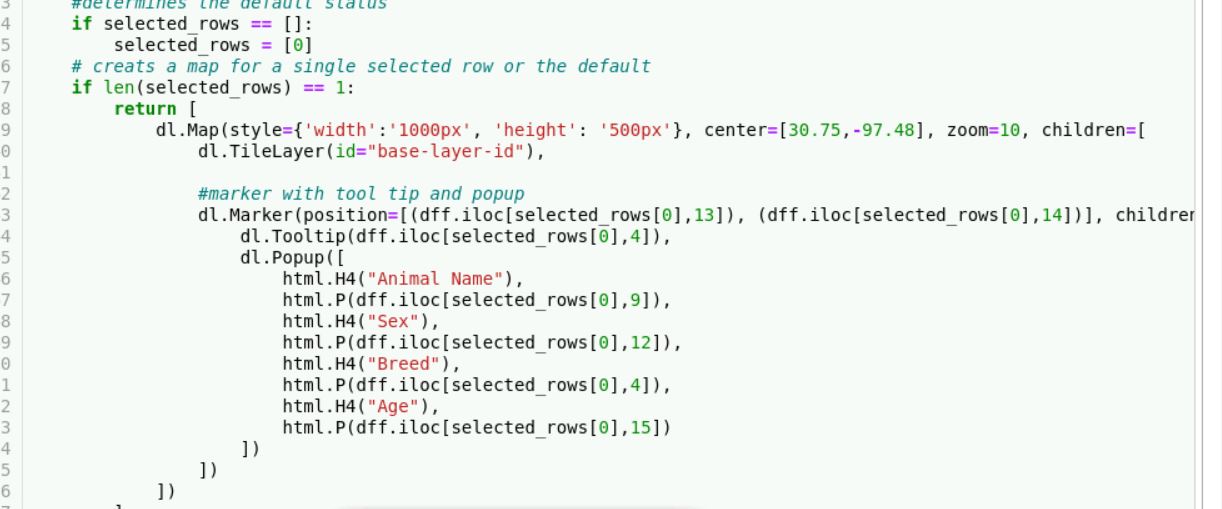
**Code Samples from Dashboard**

Complex Query

A picture containing table

Description automatically generated

Map Markers



Pie Chart

Graphical user interface, text, application

Description automatically generated

### Code Example (CRUD Operations)

Create:

def create(self, data):

if data is not None:

data\_create = self.database.animals.insert(data) # data should be dictionary

return data\_create

else:

raise Exception("Nothing to save, because data parameter is empty/incorrectly formatted")

Read:

Read(target):

def read(self,data):

if data is not None:

data\_read = self.database.animals.find\_one(data,)

return data\_read

else:

raise Exception("Nothing to read because data parameter is empty/incorrectly formatted")

def readAll(self, data):

data\_read = self.database.animals.find(data,{"\_id":False})

return data\_read

Update:

def update(self, query, data):

if data is not None:

data\_update = self.database.animals.update\_one(query, data)

return data\_update

else:

raise Exception("Nothing to update because data parameter is empty/incorrectly formatted")

Delete:

def delete(self,data):

if data is not None:

data\_delete = self.database.animals.delete\_one(data)

return data\_delete

else:

raise Exception("Nothing to delete because data parameter is empty/incorrectly formatted")

## Contact

Tyanna Prince

[Tyanna.prince@snhu.edu](mailto:Tyanna.prince@snhu.edu)