Jeremia Faust

Doctor Scott P Overmyer

CS-340 Client/Server Development 21EW1

October 17, 2021

# CS 340 README

## About the Project/Project Title

As an employee of Global Rain, I have been assigned to work on a project for Grazioso Salvare. I am tasked with creating a software application that will use existing data that will be used in creating a database and client web application. We are using a full stack code model with MongoDB as the base layer, python for the middle-layer and finally python for the web application which will be the dashboard to interface with the database.

## Motivation

Grazioso Salvare searches for dog candidates for search and rescue training. He has an agreement with five animal shelters and needs an application to help search through the data he receives. This will allow him to focus more on his job than sifting through the raw data that he receives.

## Getting Started

To get a local copy up and running you need to

* start MongoDB with the use of /usr/local/bin/mongod\_ctl start.
* run the python code file project2\_1 to start the program.

## Installation

The tool you will need for this application:

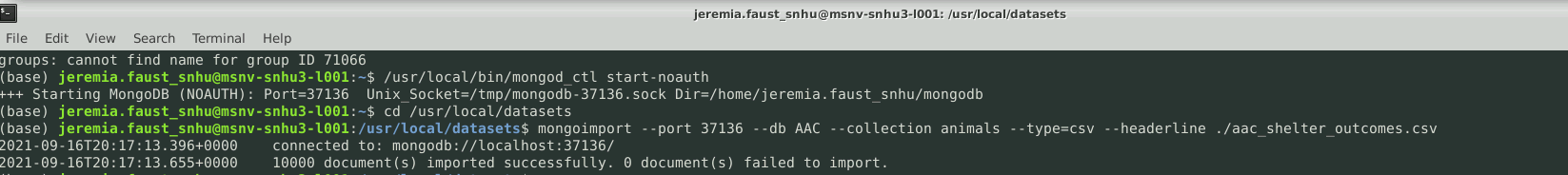
* MongoDB : <https://www.mongodb.com/>

MongoDB is an application to manage databases and has many functions.

To start the server,

* we use the command /usr/local/bin/mongod\_ctl start – noauth
* to use mongoDB after the server is started, we just type mongo
* show db command -will show us all the databases that are available
* use AAC command - loads us into the database
* show collections- shows us the collections within the database AAC
* a sample command that can be used is db.animals.find\_one(). This is used to find the first entry and displays it.

The command that are important for this project are insert\_one(), find(), update(), and delete\_one(). This is the bases of the Crud level.



**How to install database**

* Open terminal
* Start mongodb server
* Import database
* Python3 : https://www.python.org/
* Jupyter notebook : <https://jupyter.org/install>

Jupyter notebook is an IDE that can use python to create programs. One thing to note is do not create a notebook and try to save it into a .py file. It will think it is still a notebook. What you need to do is open a .txt file and save it as a .py file. this is important when you are creating object-oriented coding. If you make change remember to reset kernel otherwise you will continue using old variables.

* Database AAC: this is the current database that we will be using within mongoDB. We could add to the database by adding to the collections. But this will need some additions to the base code because the code is directly hard coded to only use the one collection.
* Dash Framework: <https://plotly.com/dash/>

To use the dash framework, you have to import the libraries to be able to use the dash framework

Text

Description automatically generated

## Usage of the base level

The base level of our stacked coding is the database in which we use mongoDB. MongoDB is used to maintain the functionality of our database with authentication. It was created in the terminal using mongoDB. Two accounts we created an admin and aacuser account. The admin has full administrative privileges. The aacuser account only has privileges with the AAC database. The database that we are using is the AAC database with the animals’ collection. This AAC database contains the raw data that was received from the animal shelters. We also implemented an index to provide quicker access to the data.

Text

Description automatically generatedText

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

**Usage of the middle layer**

At this point, we switch over to the middleware or the glue level of our stacked coding. This is done by using Python to create the CRUD functionality that interacts with the base level. To bridge the gap between mongoDB and python, we use the python driver called PyMongo. The PyMongo driver allows python to interact with MongoDB with the need to use the terminal. Within python, CRUD functionality was created using object-oriented programing that controls MongoDB. There is a function for each of the CRUD and they are create, read, update, and delete. When the create function is called it inserts the data using the insert\_one() command for mongoDB. When the read function is called, it uses the find() command for mongoDB to search the data with the parameters inputted then returns results. When the update function is called the update(data,{“$set”:update}) is used to find and update one entry. So, if there are multiple entries with same search parameters, only the first one found will change. When the delete function is called, delete\_one() is used to delete the first entry that was indicated. The CrudTest notebook tests the four functions create, read, update, and delete. We can call them by importing the functions from the .py file. This makes the code reusable and keeps the code clean. With the tests, an array of entries was created. the create function uses a for loop to call and insert an element of the array into the database. For the read test, the read function called to search for every instance of the {“type”:”dog”} and print it out to console. For the update function, it updates the name of the dog from {“name”:”test1”} to {“name”: “Bob”}. For the delete function we remove the entry with the {“type”:”cat”}. For the test we reuse the read function multiple times to show that the other functions are working as intended

## 

## Code Example for CRUD

Graphical user interface, text

Description automatically generated with medium confidence

This show the class being set up and it also finds the database from mongodb

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

This is the code for the c and r in crud.

Graphical user interface, text, application

Description automatically generated

This is the code for U and D in CRUD

### Tests

To run the test run the notebook file CrudTest

*Text

Description automatically generated*

*Text

Description automatically generated*

This is the expected output. As you can see the tests have been run a few times and it shows each entry.

Table

Description automatically generated

**Usage of Client level**

The client level is the final level of our full-stacked development. This is the level where we create a dashboard for the client to interact with the database. We use the dash framework within python to create a visual pleasing and easy to use dashboard for the users to interact with. Dash is a python framework for creating web applications that require an analytic format. It consists of two parts to function the layout and the interactively. The Dash framework uses two libraries to create the layout of the web page. These are the dash\_html\_components as dcc and the dash\_core\_components as html. The interactive section creates the functionality of the website such as making the buttons work.

For the client level using dash python framework within the Jupyter notebook we start by importing out CRUD file that will provide the functionality to our database within mongoDB. We then hardcode the authentication to our database. Finally, we set up our class read method so that we can use the Crud functionality within Dash. Now our project is all set up to create the layout of our web page. Our layout uses a tree format starting with the html.Div(). To import out jpeg of the Grazioso logo we first put the jpeg in the same folder as our files. Then we import the exact name to a variable, and we encode the image using the base64. The client wants the image once clicked to go to the website <https://www.snhu.edu>. To do this, we use the dcc.link() component. The logo jpeg is use as part of the title page and we finish the title with the html.Center(html.B(html.H1('Animal Shelter Web Dashboard', style={'height': '40px', 'padding':'10px','color':'red'}))),. The next part of our layout is the data table which uses DataTable() as the syntax create a table. The data table is used to put the raw data into an interactive table that is visually pleasing and easy to read. We also created four buttons using the html.button(). The rest of the lay out set up the locations for the graph of the breed available within search parameters and the geolocation map. The final portion of the lay out is the creator’s signature.

The final step of the dash framework we create the interactivity portion. This section has two parts the callback and the method. The call back links the layout to the methods that tells the layout to do things. This provides the website with the ability to use the buttons and creates the graph and map that is required for this project. The callback for the buttons calls for the data table and the buttons that were created in the layout thus linking to the method on\_click. The on\_click method has five if statements with a return back to the datatable. The first statement is the default that is loaded on start. The next three statements are used to make calls to the database using the code

df = pd.DataFrame.from\_records(shelter.read({'$and':[{'$or':

[{'breed':'Labrador Retriever Mix'},

{'breed' : "Chesapeake Bay Retriever"},

{'breed' : 'Newfoundland'}]},

{'sex\_upon\_outcome' : 'Intact Female'},

{'age\_upon\_outcome\_in\_weeks' : {'$gte':26,'$lte':156}}]})).

The final button is used to the reset back to the raw data. The next callback is for the graph. We set df to the viewData of the current setting of the Data table. We return the settings for our graph. X is our breed and the type of graph which is a bar graph and then we set the color of the bars to purple. We then setup the layout with the title the height and the margins. To gather the data, we finish off with a for column in breed and an if column in df. Our final portion of code is the callback for the geolocation chart. It shows the location of the first animal in the data table. There are two sections to the return. The first part sets up the size of the map and where the start location is. The second portion is the marker position and the tool tip popup.

**Web Dashboard code example**

****

This is how you call our crud module.

Text

Description automatically generated with low confidence

Hard coded user authentication



Class read method.

**Dashboard layout**

**Logo, company name

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**



Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

**Dashboard interactivity**

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

**Completed Dashboard**

**A picture containing text

Description automatically generated**

Start state

*Text

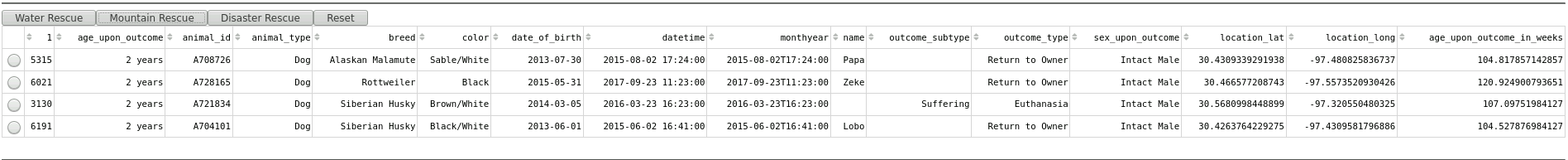
Description automatically generated*

Water rescue

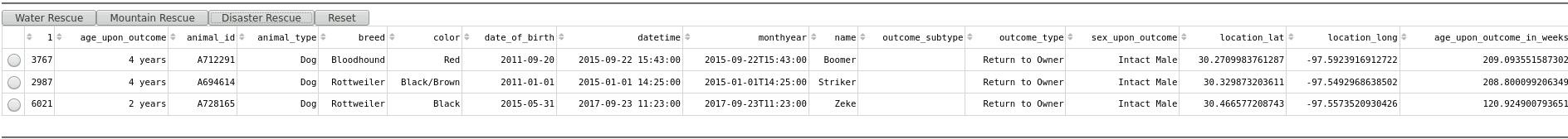
A picture containing text

Description automatically generated

Mountain rescue



Disaster rescue



Reset

Text

Description automatically generated with medium confidence

Bar chart

Chart, bar chart

Description automatically generated

Map

Map

Description automatically generated

**Encountered problems**

Most of my minor issues we quickly resolved with trial and error until I got it right. The most evident was with the graph and the geolocation map. The issues that took some time with both the graph and the geolocation map were with how to connect the data to the callback, but these were eventually figured out. There were three major problems the required that I get help from my professor. The first one was with the creation of the .py for the CRUD. I was unfamiliar with Jupyter notebook, so I took a notebook and tried changing a notebook in a .py by changing the name to do so. To fix this problem we needed to open a notepad and create the code there. The second problem I ran into was authentication, I needed to create the user account while logged into the admin account. I created the user account while the server was running in no auth mode. The last major issue was with the buttons. I originally was trying to use the buttons with the context\_callback. It in theory should have allowed the buttons to only to be used with one click not with the number of clicks. When I could not make this work, I tried to make it work using the number of clicks that is stored in n\_clicks but I ran into problems when I tried to add more than two buttons. My professor indicated that I needed to add more to the if statement. int(bt1) > int(bt2) to (int(bt1) > int(bt2) and int(bt1) > int(bt3) and int(bt1) > int(bt4)). This made all the buttons work with the number of clicks to indicate the filter type.

**Bibliography**

*Bar charts*. Plotly. (n.d.). Retrieved October 17, 2021, from https://plotly.com/python/bar-charts/.

*Dash core components*. Plotly. (n.d.). Retrieved October 17, 2021, from https://dash.plotly.com/dash-core-components.

Dash. (n.d.). Retrieved October 17, 2021, from https://dash-leaflet.herokuapp.com/.

*Learn.snhu.edu*. (n.d.). Retrieved October 17, 2021, from https://learn.snhu.edu/content/enforced/749729-CS-340-Q5031-OL-TRAD-UG.21EW5/course\_documents/aac\_shelter\_outcomes.csv?\_&d2lSessionVal=zKtLcKzjLbqzhi0hP0cnRSl9R&ou=749729.

*Python dash\_core\_components.link() examples*. Python Examples of dash\_core\_components.Link. (n.d.). Retrieved October 17, 2021, from https://www.programcreek.com/python/example/120748/dash\_core\_components.Link.

## Contact

Your name:

Jeremia Faust