CS-499-T4251 Computer Science Capstone

Professor Goggin

Carlos Aguirre

Southern New Hampshire University

March 31, 2023

Milestone Three: Milestone Four: Enhancement Three: Databases

- **Databases**

For this third category as part of my final project, I decided to work with an artifact that was completed during **CS-340 Client/Server Development.**

Artifact: Salvare Search for Rescue App.

This project required creating an app for the company Grazioso Salvare, which was looking for a program that could discover and classify available dogs using data already accessible from animal shelters. A database and a client-facing web application dashboard are two components of the full stack development of this application that Global Rain has contracted for. Grazioso Salvare will interact with and display data from a MongoDB database using this dashboard. The dashboard must have a simple, user-friendly layout that will cut down on training time and user errors.

I choose to add this project to my portfolio in order to highlight my proficiency with Python and database concepts.

I ended up with a partially completed application when I was finishing up CS-340. I'll start afresh with the application as I had some issues turning in the course's first milestones back in CS-349. In essence, I'll create a fully functional application based on the client's requirements and demonstrate my proficiency with Python and how it works with MongoDB.

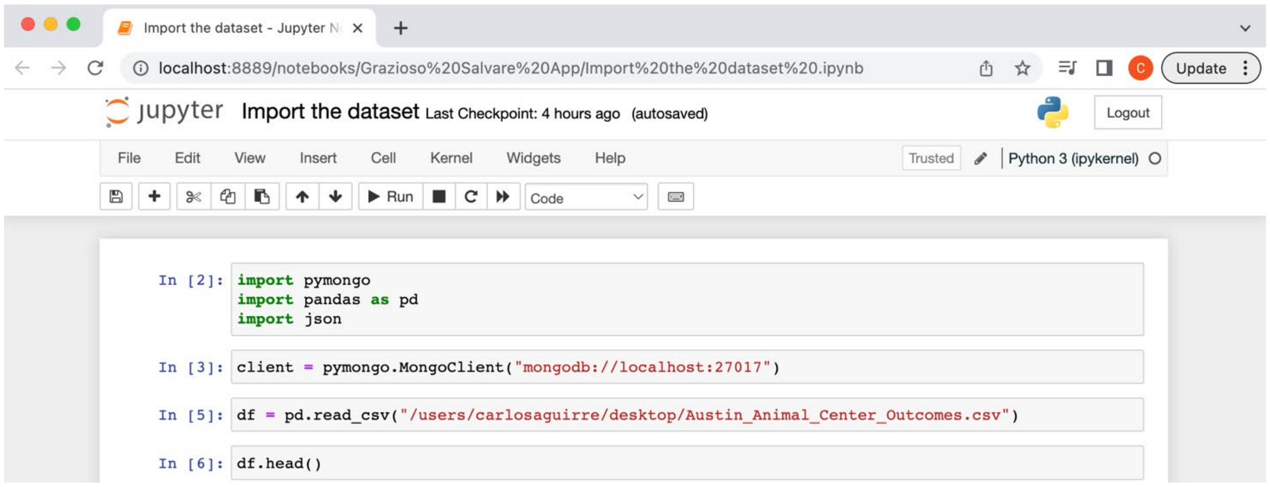
A SQL database performs SQL queries, whereas MongoDB allows JSON querying. There are other sorts of databases. In contrast to a SQL Database, which is still more predetermined and good for various forms of data storage, MongoDB is a more dynamic and complex option that is suitable for hierarchical data because of its core qualities. There are a lot of options to pick from however for this final milestone I decided to build the project using MongoDB.

Back in module one, I said "I will develop a completely functional application based on the demands of the client and show that I am familiar with Python and how it interacts with

MongoDB." In CS-340 we had the great resources of using Aporto Computer Virtual lab which had already install a lot of key components to complete this project, however as I mentioned previously, I followed a similar procedure to the other item and initially focused on making sure the database was operating properly. I utilized the Austin Animal Center data-frame and Mongo Db to build the database. The data was then loaded into the database. I created the database using Jupyter notebooks and was able to finish the Python CRUD operations for MongoDB.

To build the database using Jupiter notebooks first, I installed the library Pymongo and Pandas which would be used the read the CVS file from AAC and JSON since we must convert the CVS file to JSON.

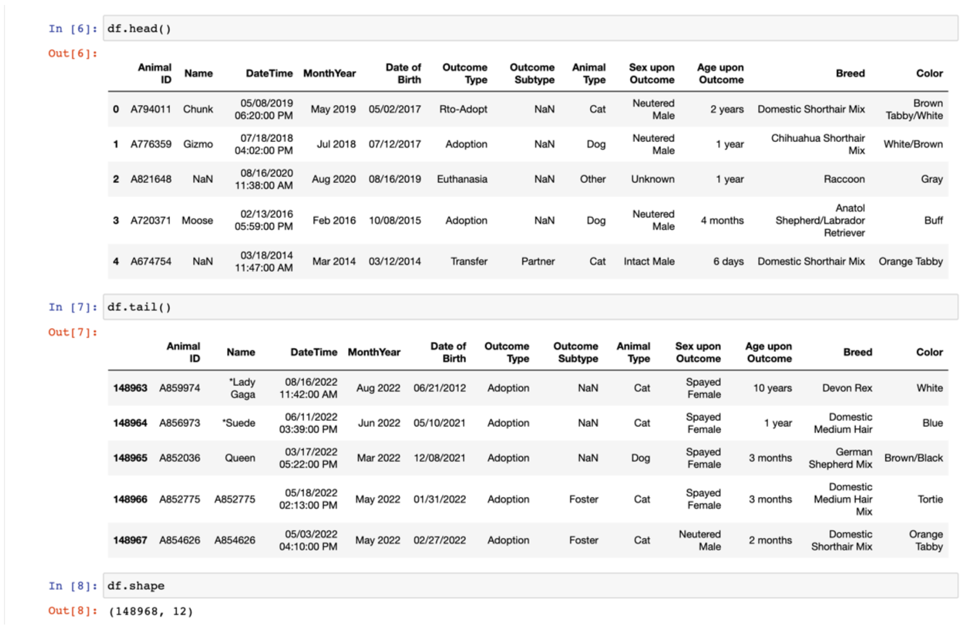
Then I move to create the python and mongo db connection by using the method Mongoclient. I created a variable client and then called the method within the variable to create the connection.


Import the dataset - Jupyter N
localhost:8889/notebooks/Grazioso%20Salvare%20App/Import%20the%20dataset%20.ipynb
Update
jupyter
Import the dataset Last Checkpoint: 4 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [2]: import pymongo
        import pandas as pd
        import json

In [3]: client = pymongo.MongoClient("mongodb://localhost:27017")

In [5]: df = pd.read_csv("/users/carlosaguirre/desktop/Austin_Animal_Center_Outcomes.csv")

In [6]: df.head()
```

The last to lines of code listed in the picture above were written to import the dataset from Austin Animal Center into a collection within the AAC database. I wanted to make sure that the database set was successfully imported so I ran the method df.head() to see the top files of the data frame and df.tail() to see the bottom files of the data frame.

Milestone Three: Milestone Four: Enhancement Three: Databases



```
In [6]: df.head()
Out[6]:
```

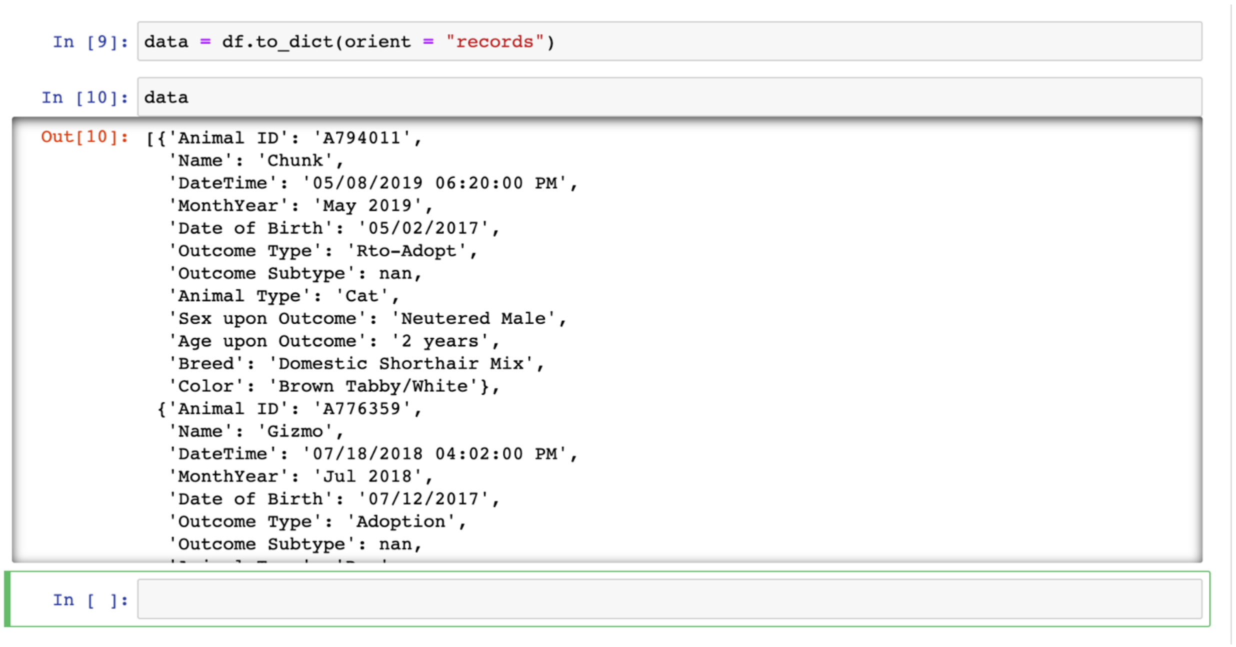
	Animal ID	Name	DateTime	MonthYear	Date of Birth	Outcome Type	Outcome Subtype	Animal Type	Sex upon Outcome	Age upon Outcome	Breed	Color
0	A794011	Chunk	05/08/2019 06:20:00 PM	May 2019	05/02/2017	Rto-Adopt	NaN	Cat	Neutered Male	2 years	Domestic Shorthair Mix	Brown Tabby/White
1	A776359	Gizmo	07/18/2018 04:02:00 PM	Jul 2018	07/12/2017	Adoption	NaN	Dog	Neutered Male	1 year	Chihuahua Shorthair Mix	White/Brown
2	A821648	NaN	08/16/2020 11:38:00 AM	Aug 2020	08/16/2019	Euthanasia	NaN	Other	Unknown	1 year	Raccoon	Gray
3	A720371	Moose	02/13/2016 05:59:00 PM	Feb 2016	10/08/2015	Adoption	NaN	Dog	Neutered Male	4 months	Anatol Shepherd/Labrador Retriever	Buff
4	A674754	NaN	03/18/2014 11:47:00 AM	Mar 2014	03/12/2014	Transfer	Partner	Cat	Intact Male	6 days	Domestic Shorthair Mix	Orange Tabby

```
In [7]: df.tail()
Out[7]:
```

	Animal ID	Name	DateTime	MonthYear	Date of Birth	Outcome Type	Outcome Subtype	Animal Type	Sex upon Outcome	Age upon Outcome	Breed	Color
148963	A859974	*Lady Gaga	08/16/2022 11:42:00 AM	Aug 2022	06/21/2012	Adoption	NaN	Cat	Spayed Female	10 years	Devon Rex	White
148964	A856973	*Suede	06/11/2022 03:39:00 PM	Jun 2022	05/10/2021	Adoption	NaN	Cat	Spayed Female	1 year	Domestic Medium Hair	Blue
148965	A852036	Queen	03/17/2022 05:22:00 PM	Mar 2022	12/08/2021	Adoption	NaN	Dog	Spayed Female	3 months	German Shepherd Mix	Brown/Black
148966	A852775	A852775	05/18/2022 02:13:00 PM	May 2022	01/31/2022	Adoption	Foster	Cat	Spayed Female	3 months	Domestic Medium Hair Mix	Tortie
148967	A854626	A854626	05/03/2022 04:10:00 PM	May 2022	02/27/2022	Adoption	Foster	Cat	Neutered Male	2 months	Domestic Shorthair Mix	Orange Tabby

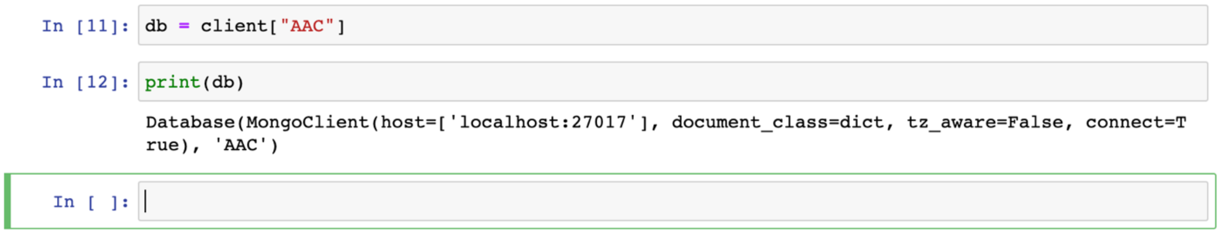
```
In [8]: df.shape
Out[8]: (148968, 12)
```

I used df.shape to display the form of our data frame at the bottom of the image above. I had to convert the data into a JSON file since, as we all know, Mongo DB stores data as keys and values.



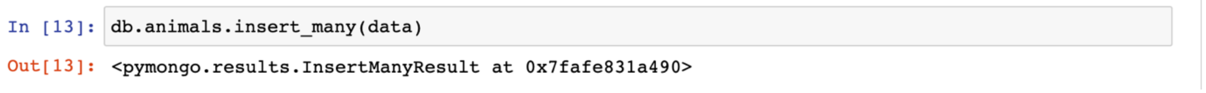
```
In [9]: data = df.to_dict(orient = "records")

In [10]: data

Out[10]: [{'Animal ID': 'A794011',
           'Name': 'Chunk',
           'DateTime': '05/08/2019 06:20:00 PM',
           'MonthYear': 'May 2019',
           'Date of Birth': '05/02/2017',
           'Outcome Type': 'Rto-Adopt',
           'Outcome Subtype': nan,
           'Animal Type': 'Cat',
           'Sex upon Outcome': 'Neutered Male',
           'Age upon Outcome': '2 years',
           'Breed': 'Domestic Shorthair Mix',
           'Color': 'Brown Tabby/White'},
          {'Animal ID': 'A776359',
           'Name': 'Gizmo',
           'DateTime': '07/18/2018 04:02:00 PM',
           'MonthYear': 'Jul 2018',
           'Date of Birth': '07/12/2017',
           'Outcome Type': 'Adoption',
           'Outcome Subtype': nan,

In [ ]:
```

I named the database in the image below: I chose to call it AAC, which stands for Austin Animal Center, just like I did in CS-34O.



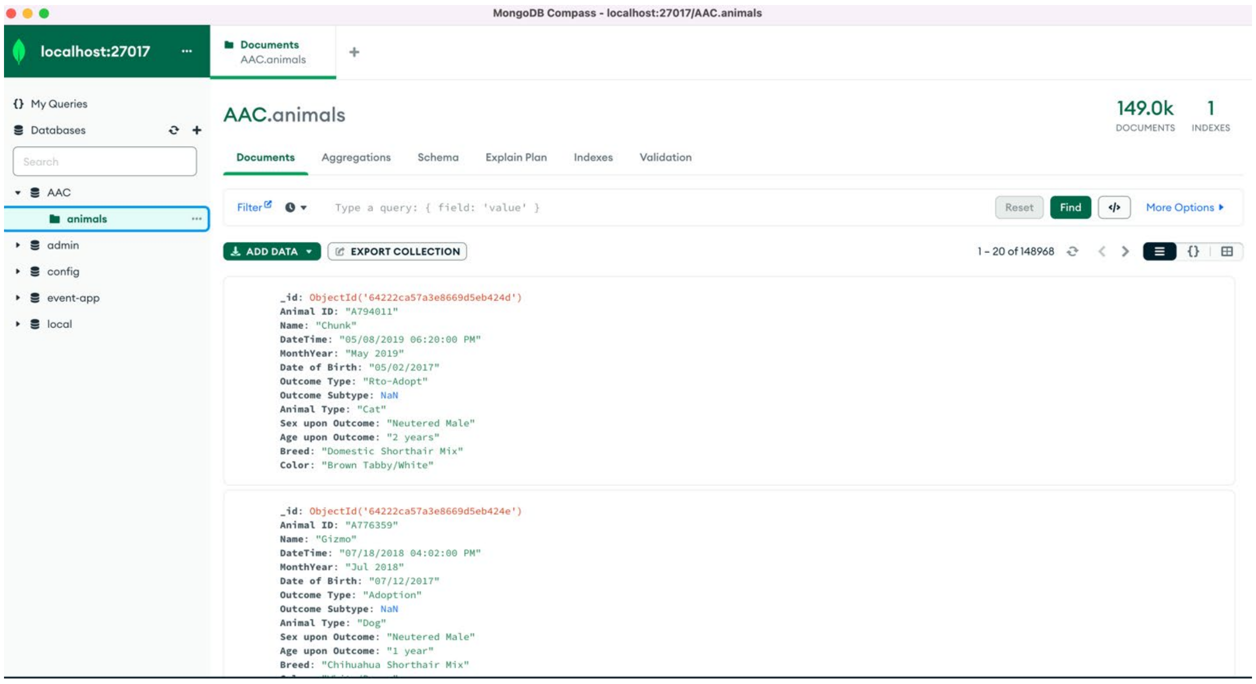
```
In [11]: db = client["AAC"]

In [12]: print(db)

         Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=T
         rue), 'AAC')

In [ ]:
```

Once created the database, I inserted the data into the database collection called animals. Data records are kept in MongoDB as documents, specifically BSON documents, which are organized into collections. One or more document collections are kept in a database. As we can see in the result, the data frame was successfully inserted.



```
In [13]: db.animals.insert_many(data)
Out[13]: <pymongo.results.InsertManyResult at 0x7fafe831a490>
```

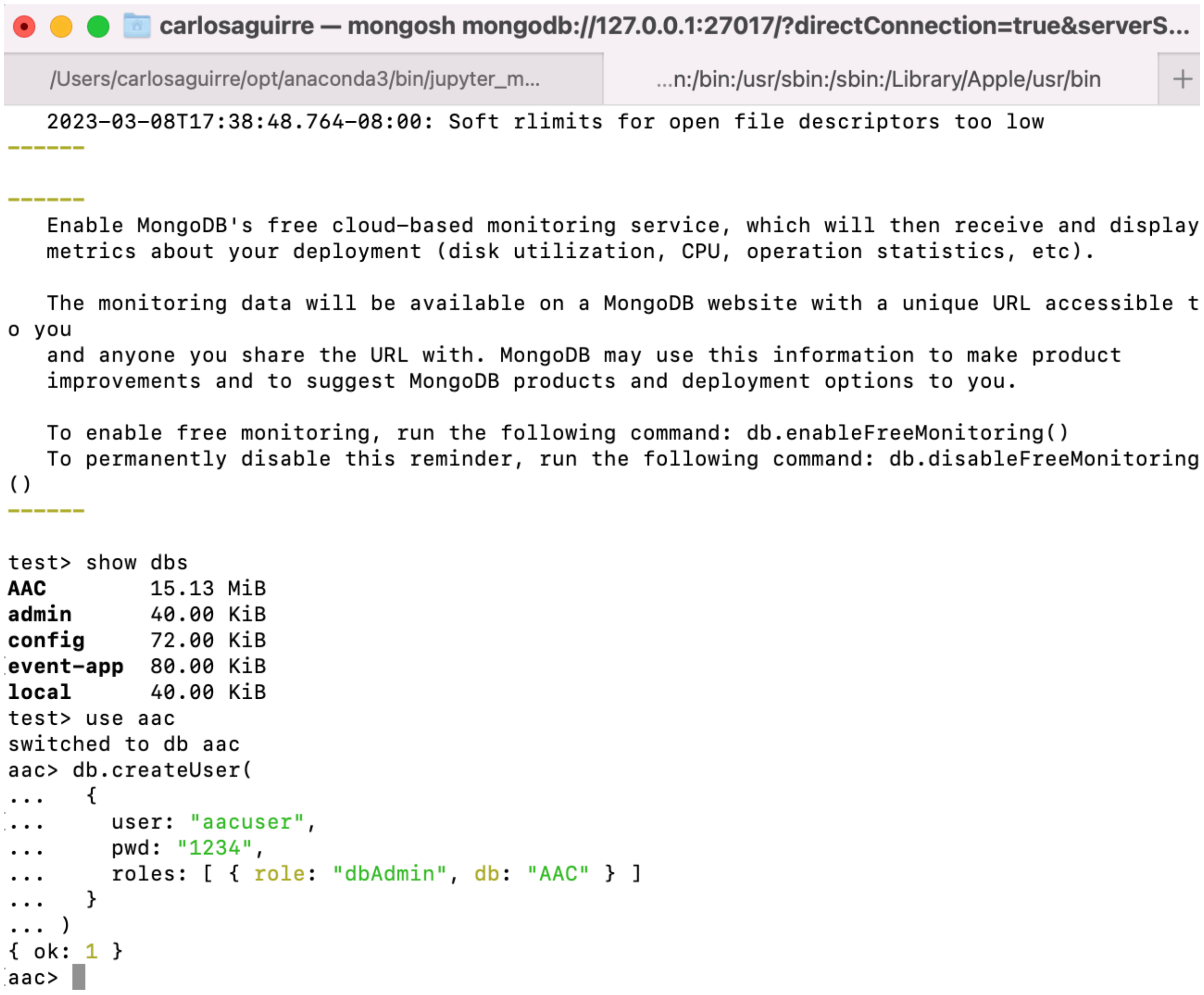
To confirm that the data was successfully imported I used Mongo DB Atlas. I decided to use Atlas because it offers the flexibility you need to create robust and effective global applications on the cloud providers of your choice while making the process of installing and managing your databases simpler. As we can see in the image below, the database and collection were both created.



```
_id: ObjectId('64222ca57a3e8669d5eb424d')
Animal ID: "A794011"
Name: "Chunk"
DateTime: "05/08/2019 06:20:00 PM"
MonthYear: "May 2019"
Date of Birth: "05/02/2017"
Outcome Type: "Rto-Adopt"
Outcome Subtype: NaN
Animal Type: "Cat"
Sex upon Outcome: "Neutered Male"
Age upon Outcome: "2 years"
Breed: "Domestic Shorthair Mix"
Color: "Brown Tabby/White"

_id: ObjectId('64222ca57a3e8669d5eb424e')
Animal ID: "A776359"
Name: "Gizmo"
DateTime: "07/18/2018 04:02:00 PM"
MonthYear: "Jul 2018"
Date of Birth: "07/12/2017"
Outcome Type: "Adoption"
Outcome Subtype: NaN
Animal Type: "Dog"
Sex upon Outcome: "Neutered Male"
Age upon Outcome: "1 year"
Breed: "Chihuahua Shorthair Mix"
```

Milestone Three: Milestone Four: Enhancement Three: Databases

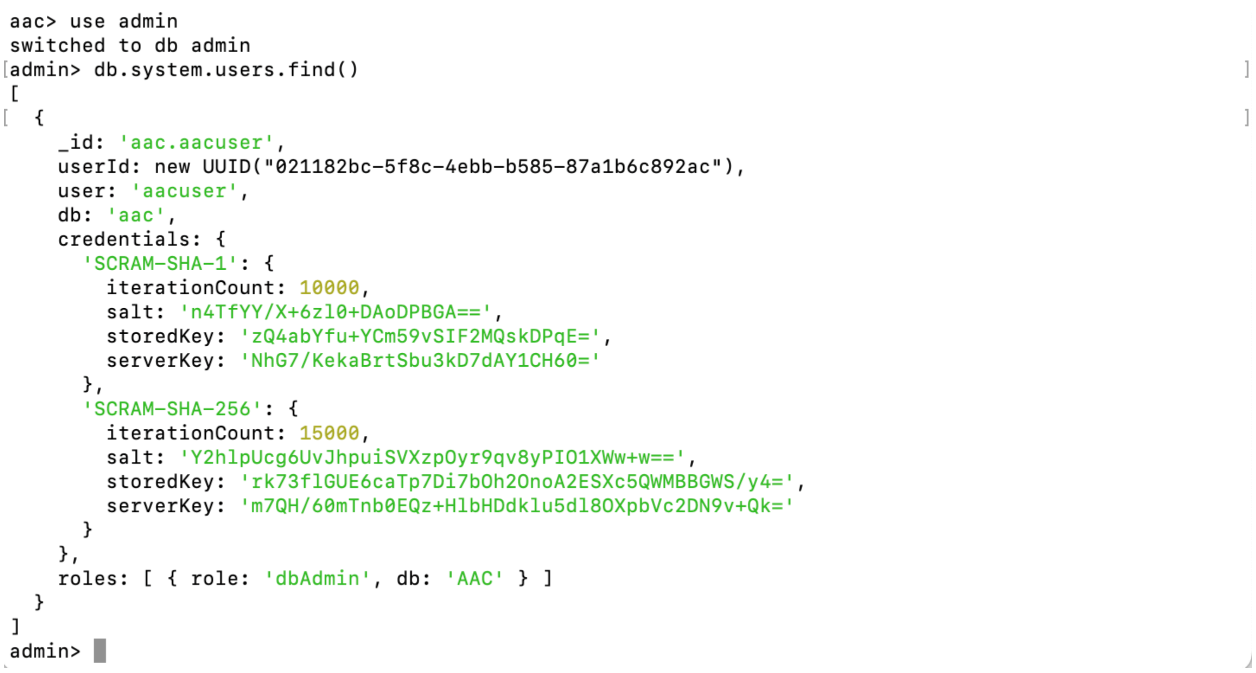
I was asked to set up an administrator account using MongoShell in order to assure user authentication to the database in order to meet the security requirements of the client.


carlosaguirre — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverS...

/Users/carlosaguirre/opt/anaconda3/bin/jupyter_m... | ...n:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin

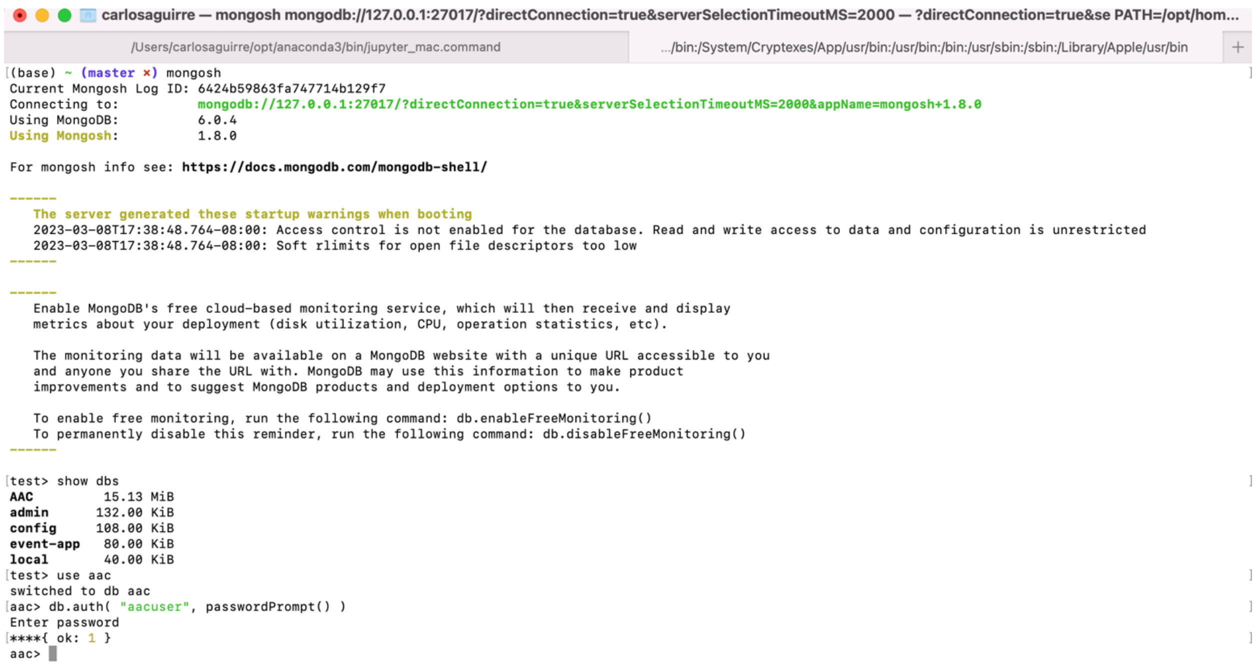
```
   2023-03-08T17:38:48.764-08:00: Soft rlimits for open file descriptors too low
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible t
o you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring
()
------

test> show dbs
AAC        15.13 MiB
admin      40.00 KiB
config     72.00 KiB
event-app  80.00 KiB
local      40.00 KiB
test> use aac
switched to db aac
aac> db.createUser(
...    {
...      user: "aacuser",
...      pwd: "1234",
...      roles: [ { role: "dbAdmin", db: "AAC" } ]
...    }
... )
{ ok: 1 }
aac> 
```

Users can access MongoDB resources thanks to Roles, according to Mongo DB Administrators can utilize a variety of built-in roles provided by MongoDB to manage user access to a MongoDB system. But you can construct new roles in a specific database if current roles are unable to express the appropriate set of privileges.



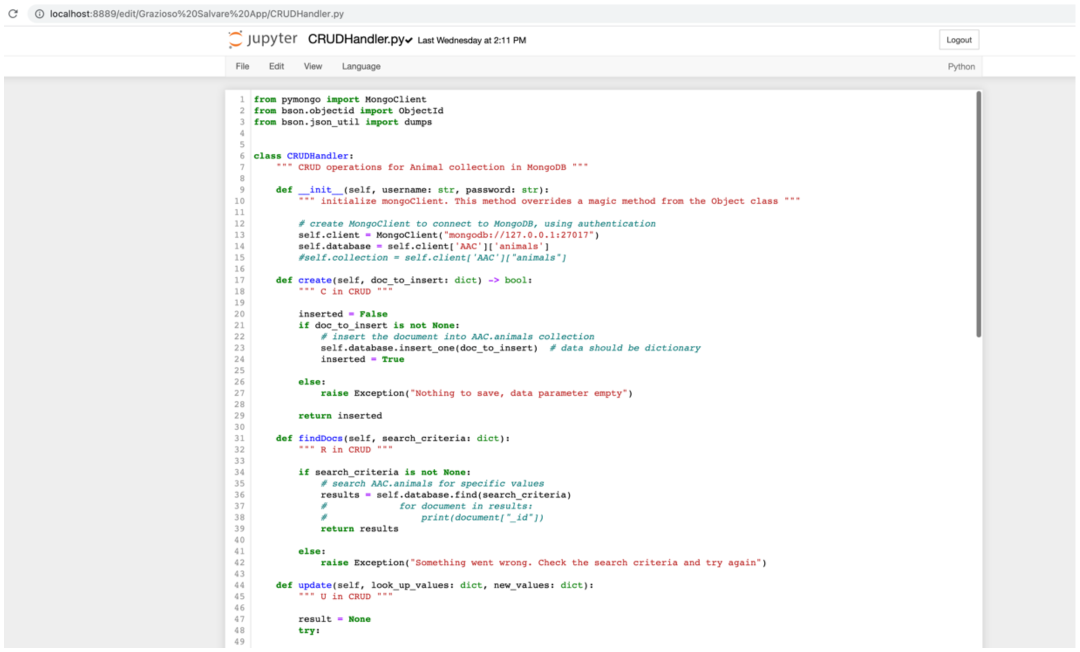
```
aac> use admin
switched to db admin
admin> db.system.users.find()
[
  {
    _id: 'aac.aacuser',
    userId: new UUID("021182bc-5f8c-4ebb-b585-87a1b6c892ac"),
    user: 'aacuser',
    db: 'aac',
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'n4TfYY/X+6zl0+DAoDPBGA==',
        storedKey: 'zQ4abYfu+YCm59vSIF2MQskDPqE=',
        serverKey: 'NhG7/KekaBrtSbu3kD7dAY1CH60='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: 'Y2hlpUcg6UvJhpuiSVXzpOyr9qv8yPIO1XWw+w==',
        storedKey: 'rk73flGUE6caTp7Di7bOh2OnoA2ESXc5QWMBBGWS/y4=',
        serverKey: 'm7QH/60mTnb0EQz+HlbHDdklu5dl8OXpbVc2DN9v+Qk='
      }
    },
    roles: [ { role: 'dbAdmin', db: 'AAC' } ]
  }
]
admin>
```

I also confirmed that the username and password worked as expected:



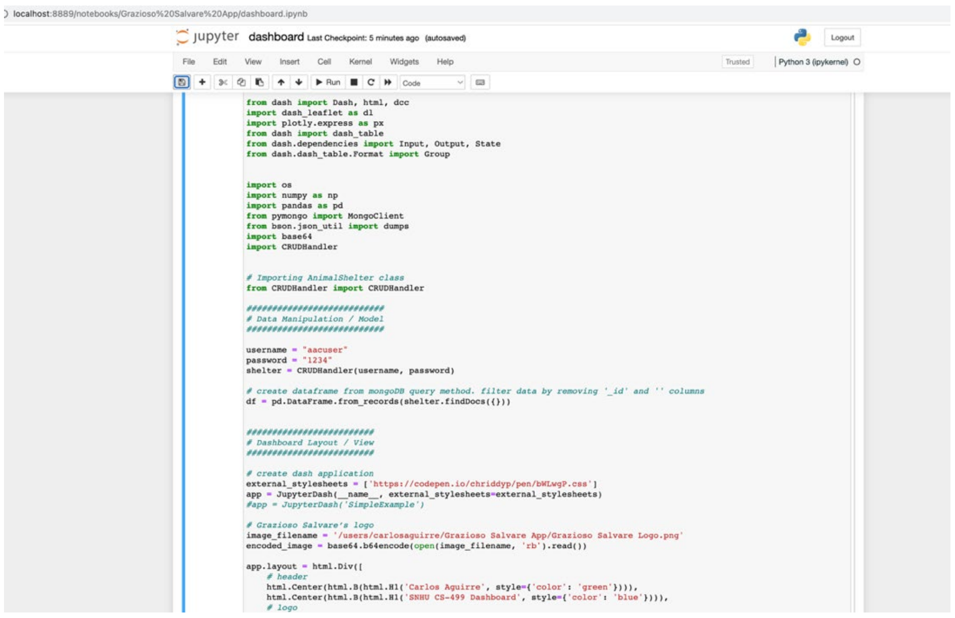
```
carlosaguirre — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — ?directConnection=true&se PATH=/opt/hom...
/Users/carlosaguirre/opt/anaconda3/bin/jupyter_mac.command    .../bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(base) ~ (master ×) mongosh
Current Mongosh Log ID: 6424b59863fa747714b129f7
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.8.0
Using MongoDB:          6.0.4
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-03-08T17:38:48.764-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2023-03-08T17:38:48.764-08:00: Soft rlimits for open file descriptors too low
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

test> show dbs
AAC         15.13 MiB
admin      132.00 KiB
config     108.00 KiB
event-app   80.00 KiB
local       40.00 KiB
test> use aac
switched to db aac
aac> db.auth( "aacuser", passwordPrompt() )
Enter password
****{ ok: 1 }
aac>
```

The next steps to continue building this application involved to create the CRUD operations, which were successfully completed using Python. A user interface that enables users to view, search for, and modify data in a database is referred to as a CRUD operation in MongoDB. MongoDB stores data as JSON objects. Documents are what MongoDB refers to as a collection's records.


localhost:8889/edit/Grazioso%20Salvare%20App/CRUDHandler.py
jupyter CRUDHandler.py Last Wednesday at 2:11 PM
Logout
File Edit View Language
Python

```python
from pymongo import MongoClient
from bson.objectid import ObjectId
from bson.json_util import dumps


class CRUDHandler:
    """ CRUD operations for Animal collection in MongoDB """

    def __init__(self, username: str, password: str):
        """ initialize mongoClient. This method overrides a magic method from the Object class """

        # create MongoClient to connect to MongoDB, using authentication
        self.client = MongoClient("mongodb://127.0.0.1:27017")
        self.database = self.client['AAC']['animals']
        #self.collection = self.client['AAC']["animals"]

    def create(self, doc_to_insert: dict) -> bool:
        """ C in CRUD """

        inserted = False
        if doc_to_insert is not None:
            # insert the document into AAC.animals collection
            self.database.insert_one(doc_to_insert)  # data should be dictionary
            inserted = True

        else:
            raise Exception("Nothing to save, data parameter empty")

        return inserted

    def findDocs(self, search_criteria: dict):
        """ R in CRUD """

        if search_criteria is not None:
            # search AAC.animals for specific values
            results = self.database.find(search_criteria)
            #           for document in results:
            #               print(document["_id"])
            return results

        else:
            raise Exception("Something went wrong. Check the search criteria and try again")

    def update(self, look_up_values: dict, new_values: dict):
        """ U in CRUD """

        result = None
        try:
```

I eventually began constructing the dashboard using plotly and dash. Plotly developed the Python framework called Dash for building interactive web applications. Flask, Plotly, and React all have the word "Dash" written on top of them. With Dash, you simply need to understand Python to construct interactive dashboards instead of HTML, CSS, and Javascript.

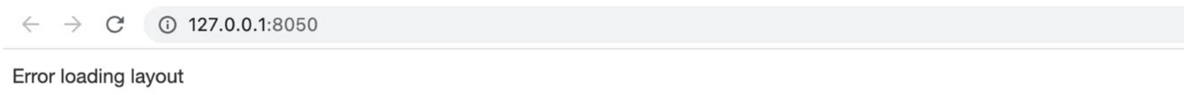

localhost:8889/notebooks/Grazioso%20Salvare%20App/dashboard.ipynb
jupyter dashboard Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```python
from dash import Dash, html, dcc
import dash_leaflet as dl
import plotly.express as px
from dash import dash_table
from dash.dependencies import Input, Output, State
from dash.dash_table.Format import Group


import os
import numpy as np
import pandas as pd
from pymongo import MongoClient
from bson.json_util import dumps
import base64
import CRUDHandler


# Importing AnimalShelter class
from CRUDHandler import CRUDHandler

###########################
# Data Manipulation / Model
###########################

username = "aacuser"
password = "1234"
shelter = CRUDHandler(username, password)

# create dataframe from mongoDB query method. filter data by removing '_id' and '' columns
df = pd.DataFrame.from_records(shelter.findDocs({}))


#########################
# Dashboard Layout / View
#########################

# create dash application
external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']
app = JupyterDash(__name__, external_stylesheets=external_stylesheets)
#app = JupyterDash('SimpleExample')

# Grazioso Salvare's logo
image_filename = '/users/carlosaguirre/Grazioso Salvare App/Grazioso Salvare Logo.png'
encoded_image = base64.b64encode(open(image_filename, 'rb').read())

app.layout = html.Div([
    # header
    html.Center(html.B(html.H1('Carlos Aguirre', style={'color': 'green'}))),
    html.Center(html.B(html.H1('SNHU CS-499 Dashboard', style={'color': 'blue'}))),
    # logo
```

Milestone Three: Milestone Four: Enhancement Three: Databases

The application has been improved because this time we currently have a ready and reliable database, Because of its adaptable schema, it is simple to store data in a way that is simple for programmers to use. In addition, MongoDB supports all the key features of contemporary databases, including transactions, and is designed to scale very quickly.

So far, the only challenge that I am facing the dash dashboard is not loading; it did load previously, but now it is giving me an error message that reads, "Error loading layout." According to some research I've done, this issue may be related to the dash libraries. When I run the code in terminal, it executed successfully. The URL displayed the It's possible that some libraries or required files are missing from your dash installation. The packages listed below could be installed using pip or conda from the terminal, but I do not know that yet.



```
127.0.0.1:8050
Error loading layout
```

References:

Mongo DB documentation., Retrieved from:

https://www.mongodb.com/docs/manual/tutorial/getting-started/

Plotly and Dash installation. Retrieved from: https://dash.plotly.com/installation

Jupiter Notebooks. Retrieved from: https://jupyter.org/