# CS 340 README Template

## About the Project/Project Title

We are adding CRUD functionality to the Austin Animal Center’s database. The create, read, update, and delete functionality are the focus of this project and work will be done in python and then implemented into MongoDB. This python module will be crucial in developing a web application that connects a client-side user interface with a database. The second part of the project that has been completed is fully working on the user facing dashboard and connecting it to the backend database. Widgets have been added and various features for the user.

## Motivation

The project description is as follows, Grazioso Salvare needs a software application that can work with their existing data from animal shelters to identify and categorize available dogs. Global Rain has contracted for a full stack development of this application that will include a database and a client-facing web application dashboard. Users at Grazioso Salvare will access the database. In this phase of the project we are completing the database, later on in phase two we will update the database and create a dashboard.

## Getting Started

Initially one must set up the mongo shell and get users and authentication sorted. After this is done, work can be done in python to setup the modules that will be later imported into MongoDB. Once the modules are completed, we can test functionality and then implement them fully. After this has been done, work on the dashboard and user interface can begin.

## Installation

MongoDb - <https://www.mongodb.com/>

Jupyter Notebook - <https://jupyter.org/>

These tools can be installed by going to the above websites and following the installation instructions.

Within mongodb we used the mongoimport function extensively, see this link for more information - <https://www.mongodb.com/docs/database-tools/mongoimport/>

Dash framework was an important part of the second part of the project. It provided the view and controller structure for the web app. This point and click interface in python provided the base that was worked off of.

## Usage

The CRUD python module works by using the AnimalShelter class object containing create, read, update, and delete methods seen below. The purpose is to allow documents in a database to be created, read, updated, and deleted in an efficient and scalable manner if needed. We use PyMongo due to its highly scalable nature, and excellent cloud computing abilities. It is also the official python driver for MongoDB. The attributes and working functionality of the CRUD operations can be seen below in both the code examples and screenshots.

### Code Example

# Complete this create method to implement the C in CRUD.

def create(self, data):

if data is not None:

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

if insert!=0:

return True

else:

return False

else:

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

# Create method to implement the R in CRUD.

def read(self,criteria=None):

# if the criteria is not None then this find will return all rows which matches the criteria

if criteria:

# {'\_id':False} just omits the unique ID of each row

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

else:

#if there is no search criteria then all the rows will be return

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

return data

# Create method to implement the C in CRUD

def update(self, data, newData):

if data is not None:

print("data updated")

return self.database.animals.update\_one(data, {'$set': newData})

else:

print("Nothing to update, because data parameter is empty")

return False

# Create method to implement the D in CRUD

def delete(self, data):

if data is not None:

print("data deleted")

return self.database.animals.delete\_one(data)

else:

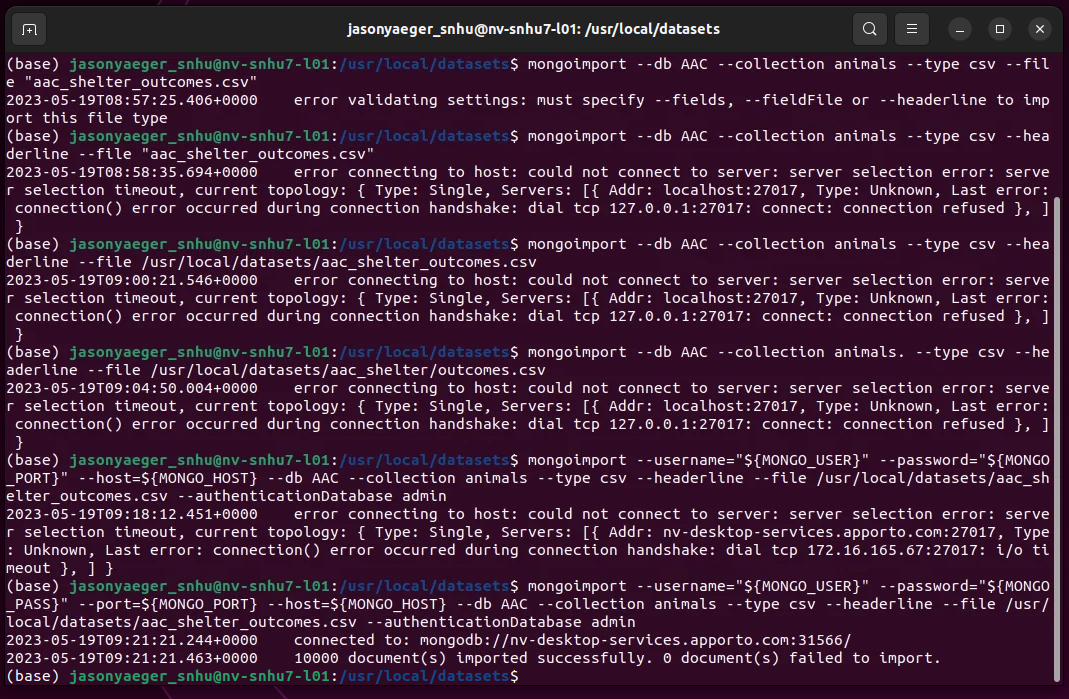
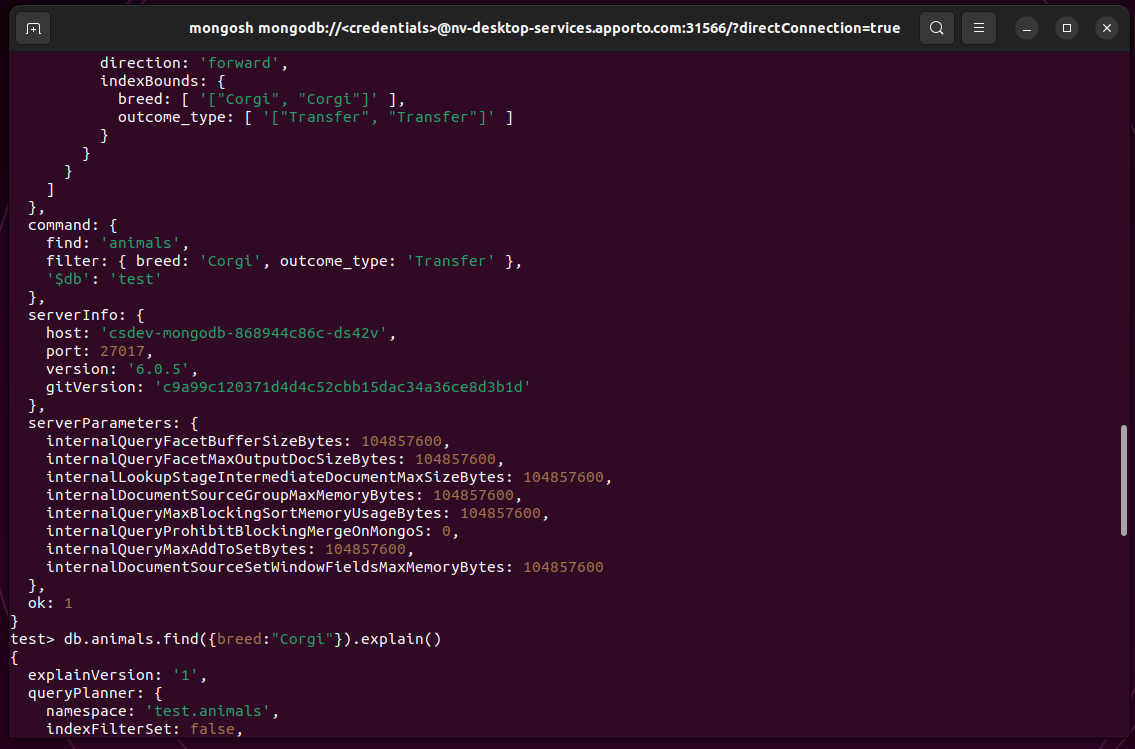
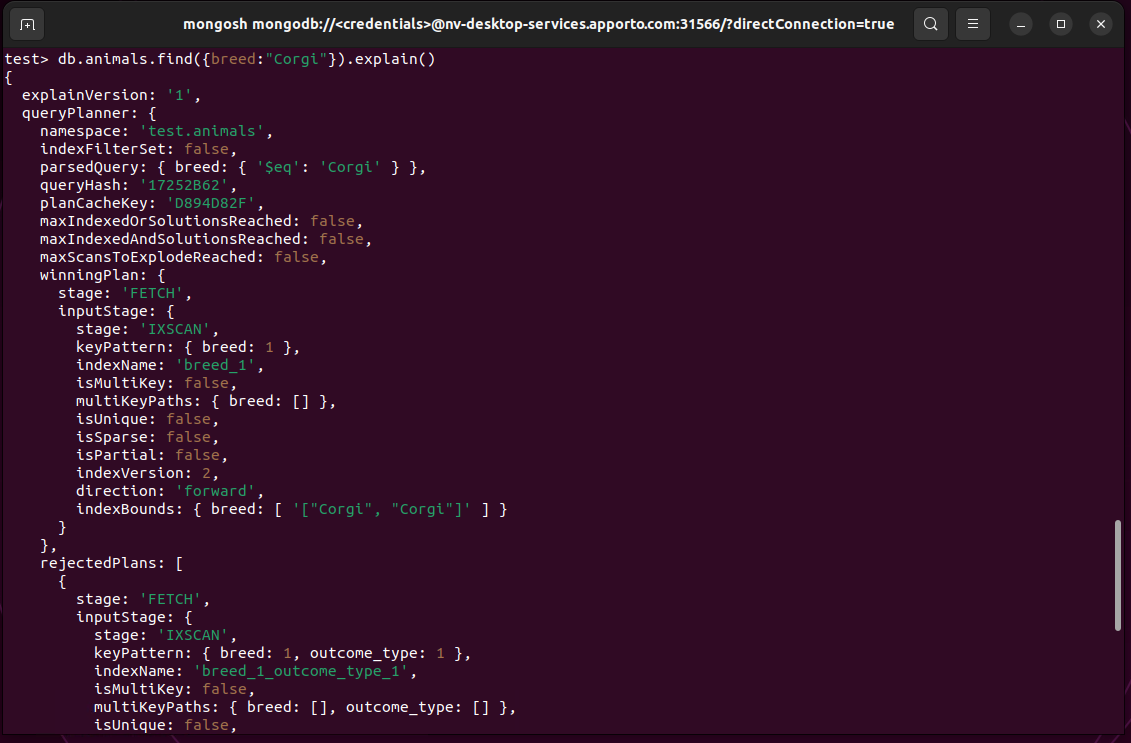
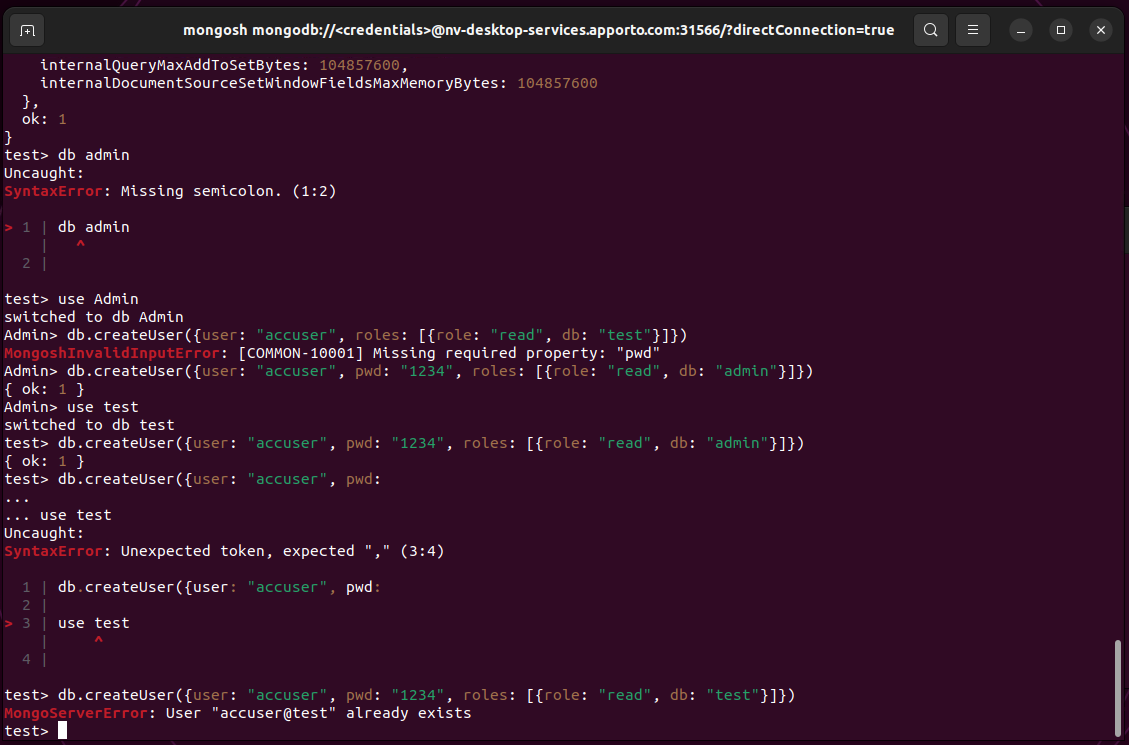
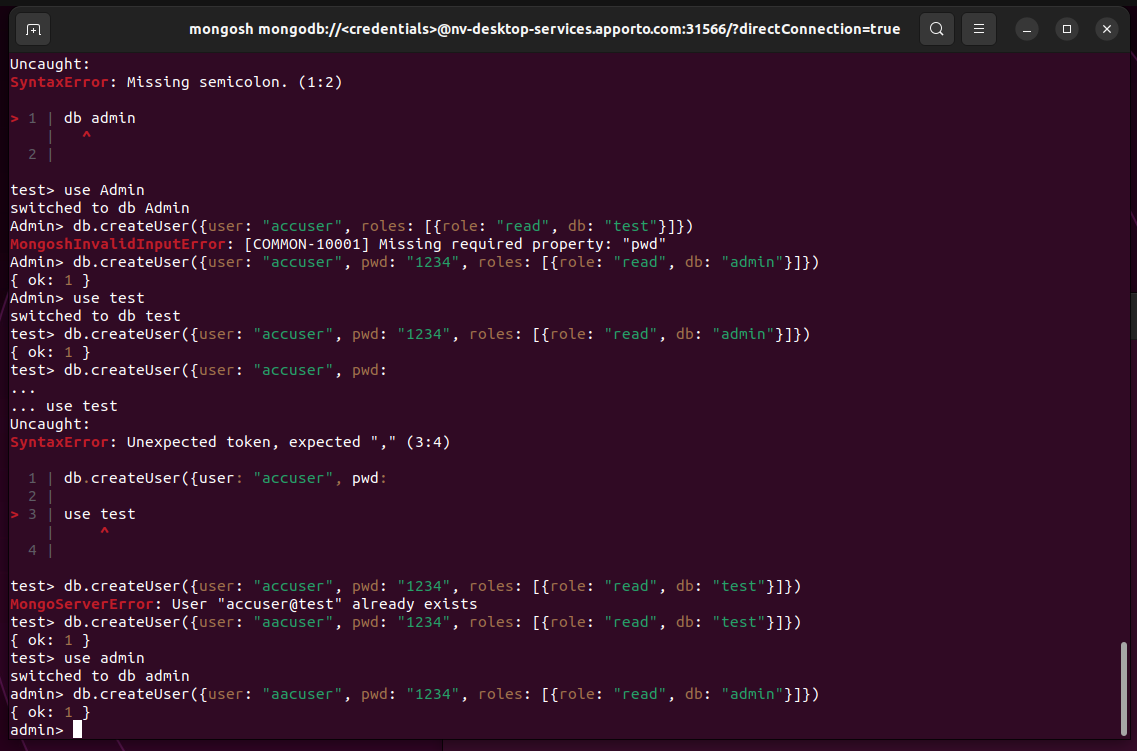
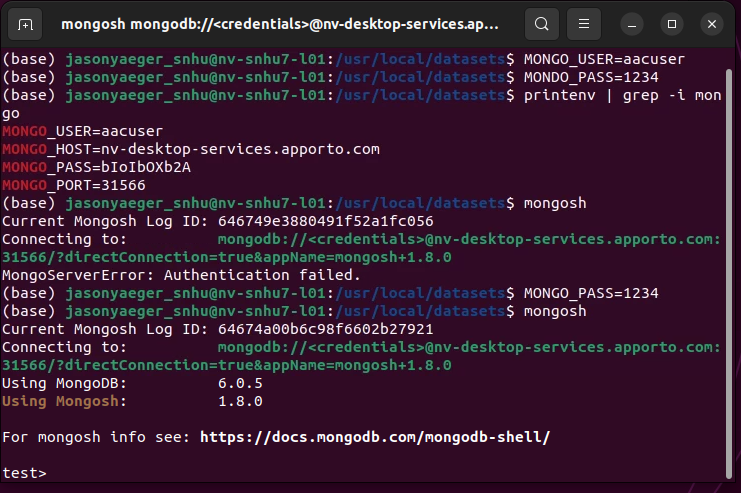
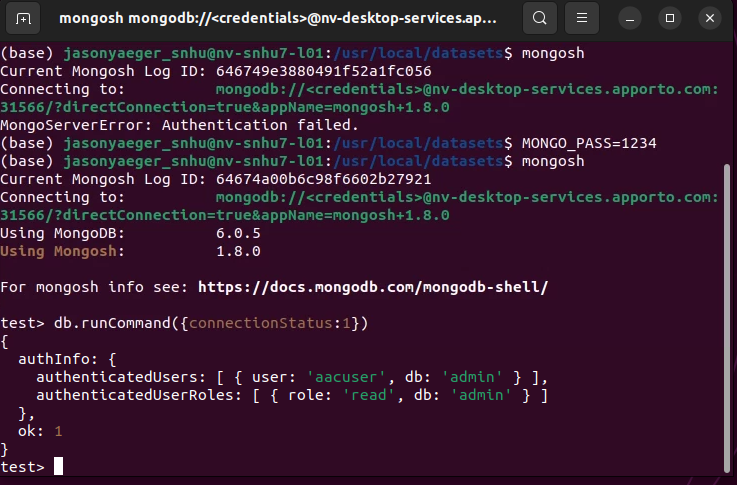
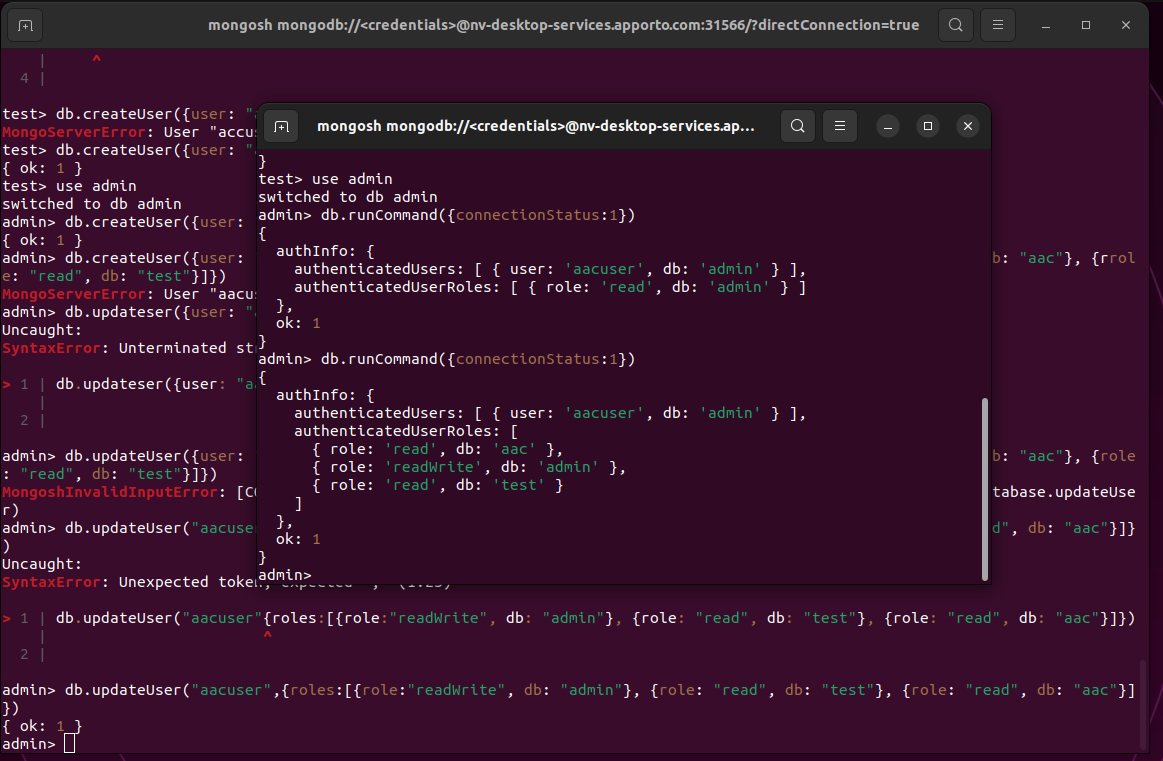
return False

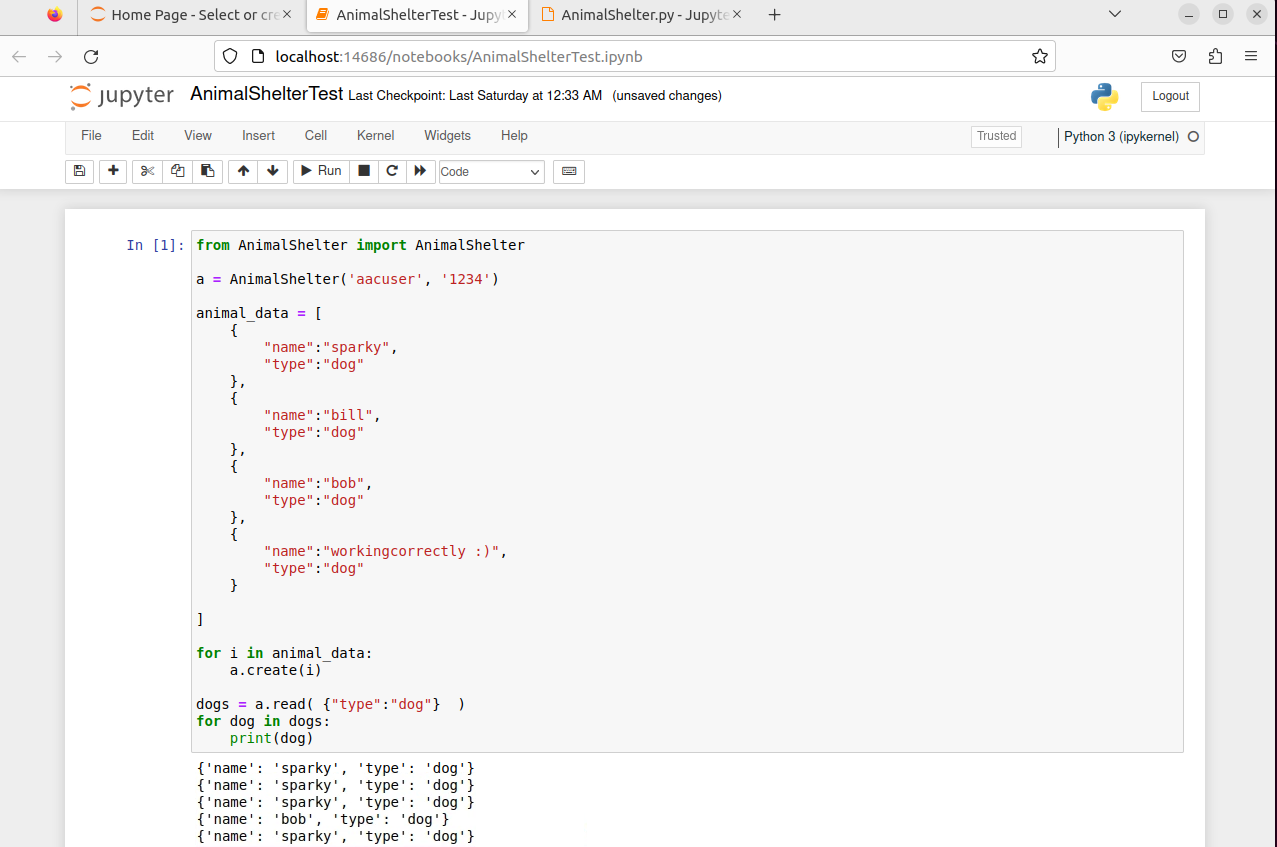
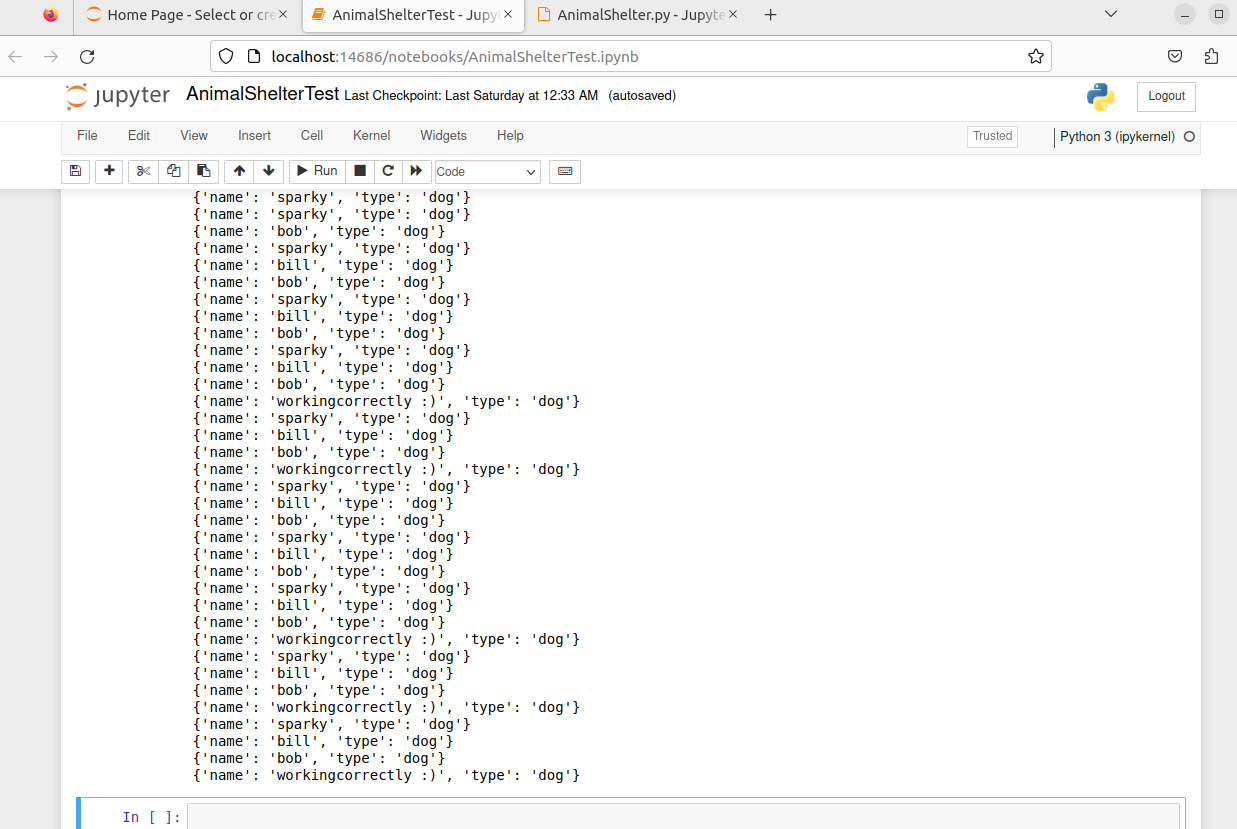
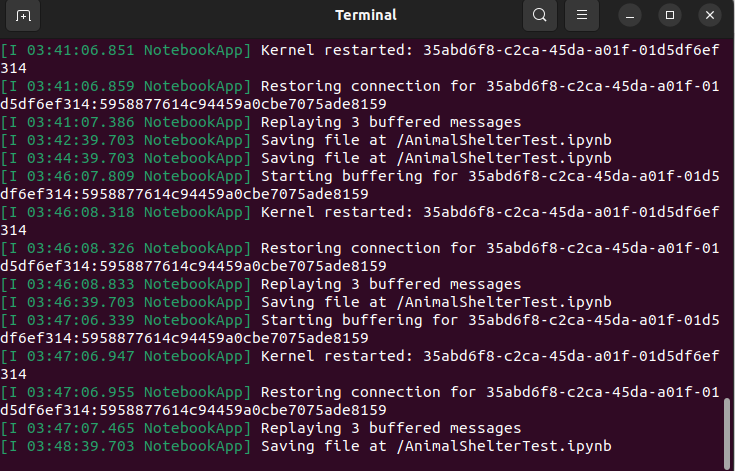
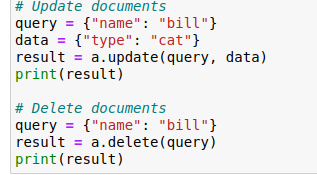
The dashboard and its usage can be seen towards the end of the screenshots section below. Dash framework was used and HTML was used to add widgets, buttons, and various interface features included titles/headers. Datatables and graphs were also included.

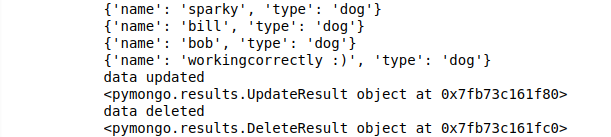
### Tests

Using the AnimalShelterTest file one can run tests to further see how the CRUD code works. Running the projecttwodash can bring up the app in localhost to then see the user interface of the web app as seen below.

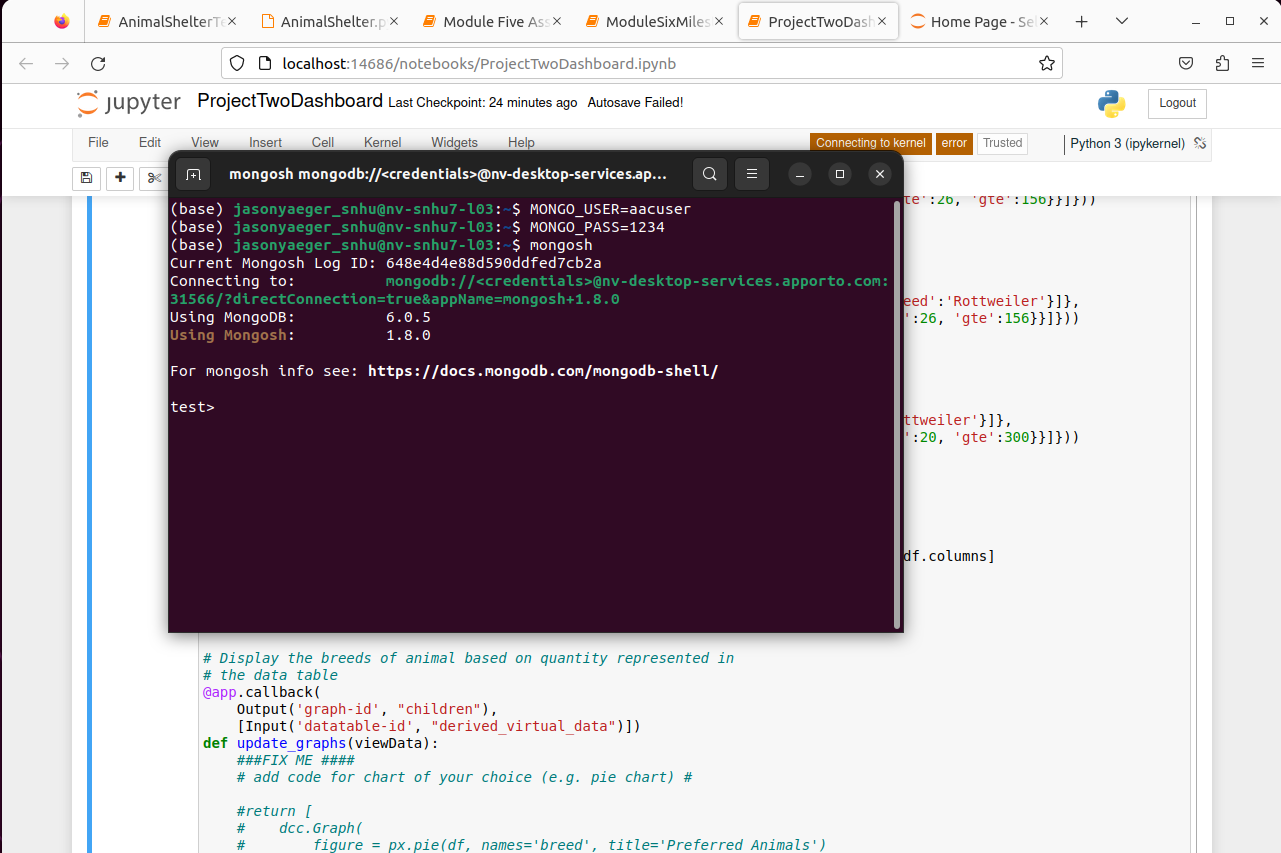
### Screenshots

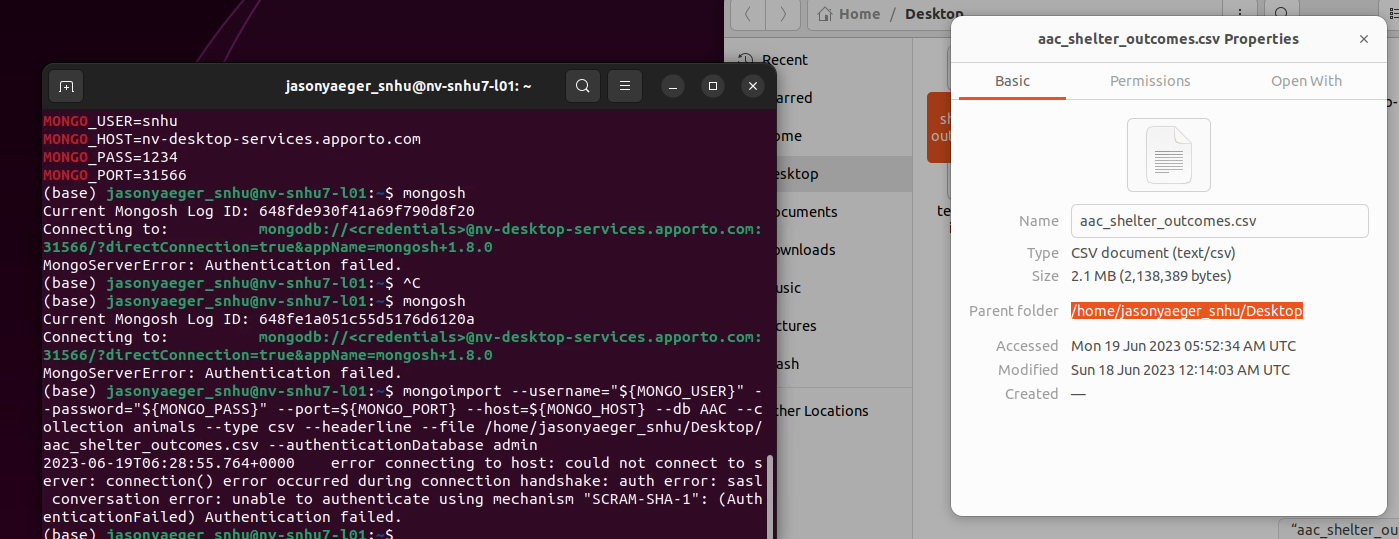
  
  
  
  
  
  
  
  
  


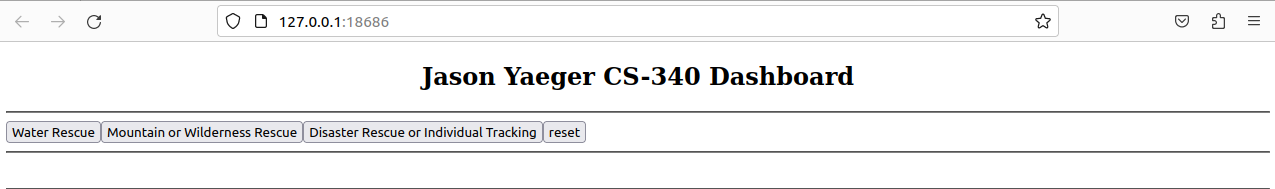
  
  
  




PROJECT TWO STARTING APPORTO







## Roadmap/Features (Optional)

*This issue has been resolved*- Current known issue regarding the passing of authentication information to the client. There have been authentication errors that are being troubleshooted.

*These features have been added-* Update and delete functionality will be added in future milestones before completion of the project.

In the second portion of the project I ran into a lot of issues but had fun trying to solve them. I would like to completely figure out why some things still are not working (like getting the image displayed) after much trial and error. If a zoom call or something is possible to better learn that would be great.

## Contact

Jason Yaeger