# CS 340 README Template

## Nicole Penner

CS-340 Client-Server Development

7-1 Project 2

## About the Project/Project Title:

This project has two major purposes – the first is to be able to access a database, while creating a code in Python to create a new instance of the database, as well as read items from the database. The name of the project is to “Create, Read, Update, and Delete (CRUD) in Python.” There needs to be a Python file where the functionality of the code is written and a Jupyter Notebook IPYNB file that tests the code and ensures it is running and functioning correctly. All this functionality exists so that the Austin Animal Center (AAC) animals database can be added to, entries can be read, updated, or deleted. This adds functional operations for the database so it is accessible beyond the MongoDB shell, and is able to be accessed by users that have access to the database. That is why an “aacuser” account is being used to access the database, because it was already set up to access it. The second purpose is to use the database to be able to create a dashboard so that a user can view and be able to navigate animals at the Austin Animal Center in Austin, TX. The dashboard ideally will first show an unfiltered view of the animals at the shelter, and allow the user to be able to filter the list to find certain criteria. A map of where the pet is located, as well as charts, and the company’s logo should all be visible on the dashboard. Also, a graph and the Grazioso Salvare logo should be present.

## Motivation For the Project:

This project exists so that users who have access to the Austin Animal Control (AAC) database, can access the database outside of the mongo shell. This is so that these users can access and add entries or read current entries in the specified database. The users who have access to this database for the animal control center need to be able to view it and add entries as needed depending on animals currently at the center. User’s also need to be able to update and delete entries as needed, depending on animals whose details change or a mistaken entry that needs to be removed. Further motivation is for users to view the list of animals available on the dashboard, be able to sort the list, and see animals that follow the query selected. This helps the user to be able to keep track of animals in the AAC database.

## Motivation for using Mongo DB

Mongo DB allows for a quick and easy setup for using the system itself and use of a .csv document. Mongo DB also allows for an easy setup of a database from a .csv file. Mongo DB is also extremely Python friendly, and allows for ease of use for the database and later using Python to create a dashboard for it. Python also has CRUD functionalities that allow for easy manipulation of Mongo DB databases. Overall, it was an easy and logical choice to choose Mongo DB.

## Motivation for using Dash:

Dash was chosen for it’s ease of use with Python, and its ease to create and set-up framework for a dashboard. Dash uses HTML that helps to use and different outputs to different dashboard components. Callbacks can easily be used within Python as well if data is changed. Dash itself is a tool that lends itself easy to learn for beginners. Jupyter Notebook was used to run the Python and Python test code as well.

## Getting Started:

First, to set this project up the Linux shell needs to be opened and mongo needs to be started. Without this step, we would not be able to access the database located in the mongo shell. After that, you would need to actually start and open mongo in the Linux shell. This allows the Jupyter Notebook and Python file created to test to actually be able to access the database needed to perform the actions specified. A user account should also be created in mongo called “aacuser” that can read and write items in the “AAC” database. The .py file is used for the “CR” functionality, or create and read. The create method is comprised of “self” which accesses the AAC database that aacuser can access and make changes to. The bulk of this code relies on if the data field is not empty, “.insert” is used to the database to add that entry. If there was nothing entered, an exception is thrown because there is no data to add to the database. The read functionality is created so that if we search for a specific key-pair, and it is found, it can be output to the user. This is namely done by using “.find” in the AAC database. The updateOne() functionality is needed to be able to update an already existing entry to change its information. There are also functions that are able updateMany() in case of an acronym change or change of terms used at the shelter, a key pair can be searched for and updated so that many are changed. The next functionality that needs to exist is a delete function to be able to delete entries as needed. The function to delete one entry is deleteOne(), this allows us to search for a key pair and delete it. This is not for our current purposes, but deleteMany() can be used to search if multiple entries have the same key pair, and delete all those entries. All of these functions ensure that all the “CRUD” functionalities are present in the python file. The main challenges I ran into were connection errors, I had many issues connecting to the database. This took some simple troubleshooting and sometimes it was as simple as restarting Jupyter to reconnect.

## Installation:

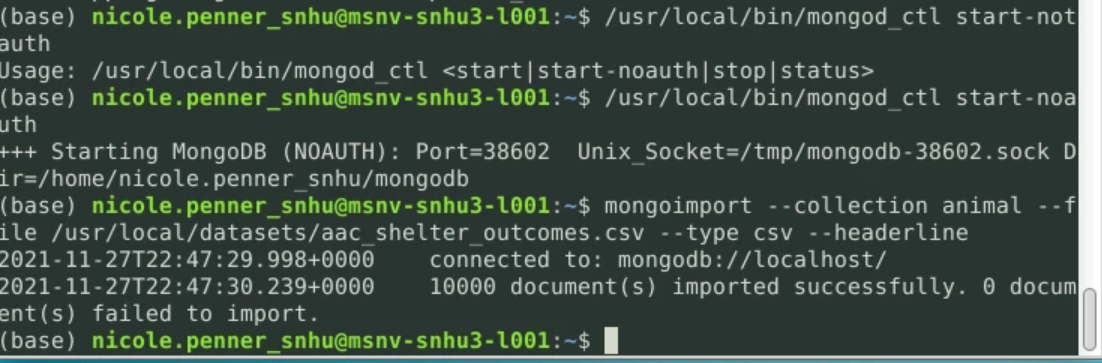
There are a few different tools that are necessary to be able to run this code correctly. A Linux shell that is able to run Mongo needs to be available. Jupyter notebook also needs to be installed and accessible. Finally, for this code to run correctly the AAC database needs to be uploaded and available to be accessed by a user account in mongo. With all of this, the Python code and a Jupyter Notebook IPYNB file that tests the Python code, it should be able to be run successfully. To upload the AAC database so it can be used “/usr/local/bin/mongod\_ctl start-noauth” should be entered in the Linux shell. Then “mongoimport -- collection animal -- file /usr/datasets/aac\_shelter\_outcomes.csv -- type csv - - headerline” should be entered. Mongo should be then be stopped, and followed up by starting mongo in the Linux shell. Jupyter Notebook should then be opened, and the .py and .ipnyb files can be opened. In these files pymongo should be imported from MongoClient to ensure that the .py code runs properly. I had authorization errors and that was simply because I forgot to have Mongo started up, which was a simple fix.

The following are screenshots of the separate steps to import and install all files to be able to use the “CRUD” and dashboard files successfully:

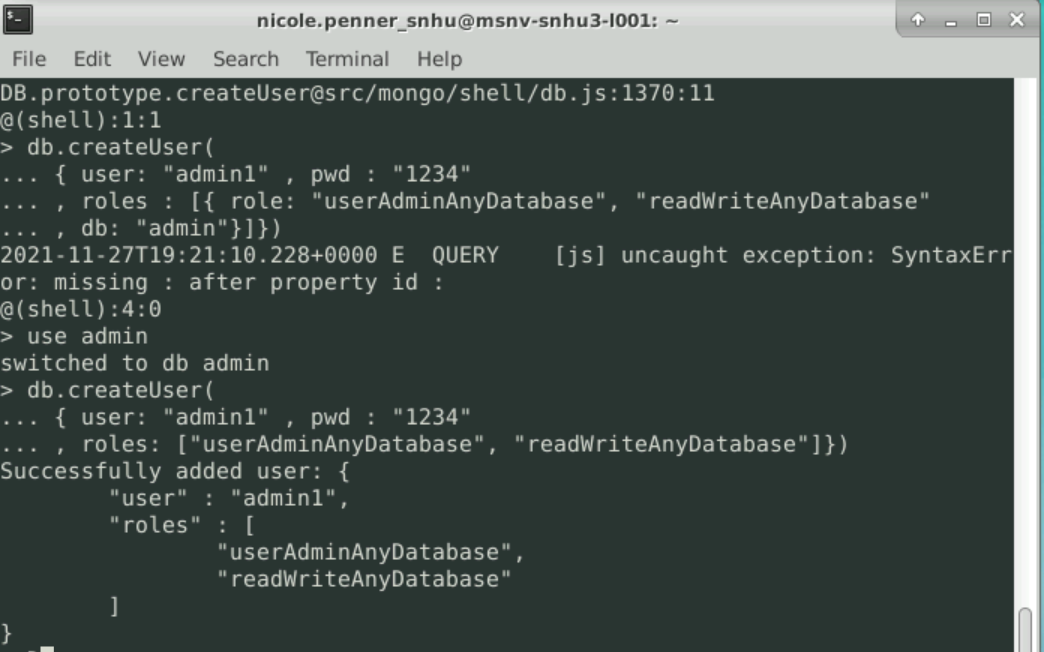
1. Open a Linux terminal window
2. Use the following commands in the Linux terminal to be set up to open Mongo
   1. cd /usr/local/datasets
   2. /usr/local/bin/mongod\_ctl start-noauth
3. After Mongo is start with no authorization a port number will show up, mine will be in my screenshots, if you are trying to replicate this process use your own port number.
4. To import the correct file for AAC the following code is entered into the Linux shell:
   1. mongoimport --port \*\*\*\*\* --db AAC --collection animals --type=csv --headerline ./aac\_shelter\_outcomes.csv  \*\*\*this is where your port number is entered\*\*\*

The following are screenshots that further outline the process to load the files, create an admin, and a user account for the database “aac”:

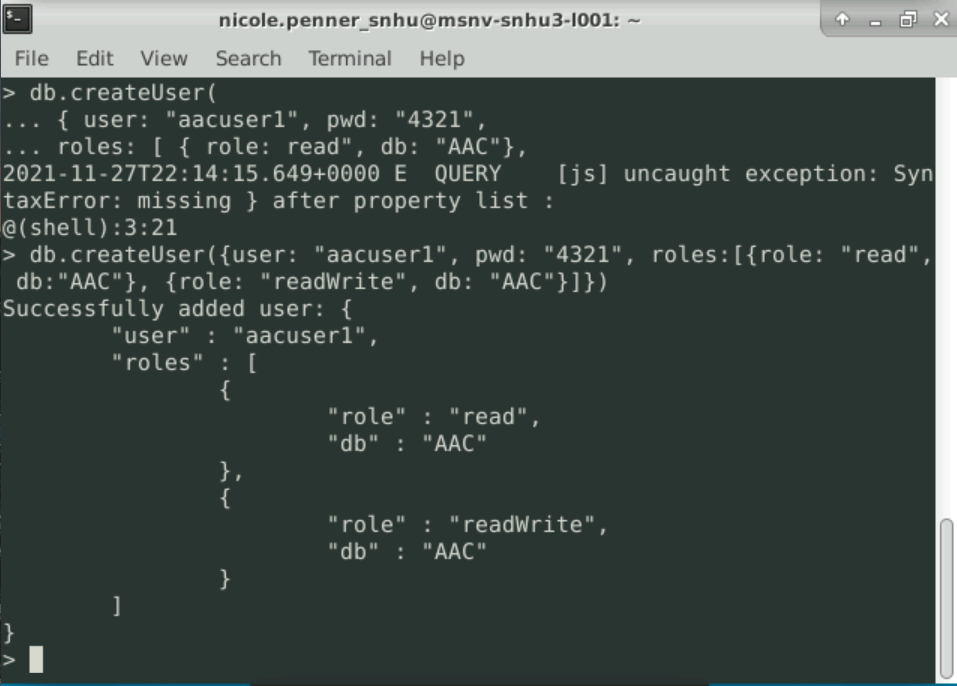
* Loaded aac\_shelter\_outcomes.csv to the database:



* Created an administrator account to access and edit any database in the system:

\

* Successfully creating “aacuser” which will be used to access and edit the AAC database:



## Other tools used:

* + - 1. Pandas is used in this project, specifically to help create a data-frame for Python.
      2. Jupyter Notebook is used as the command line for the Python files that are using the AAC database.
      3. Plotly is imported in our test files because it is a charting tool. Therefore it is useful to create graphs.

## Usage:

### Code Example:

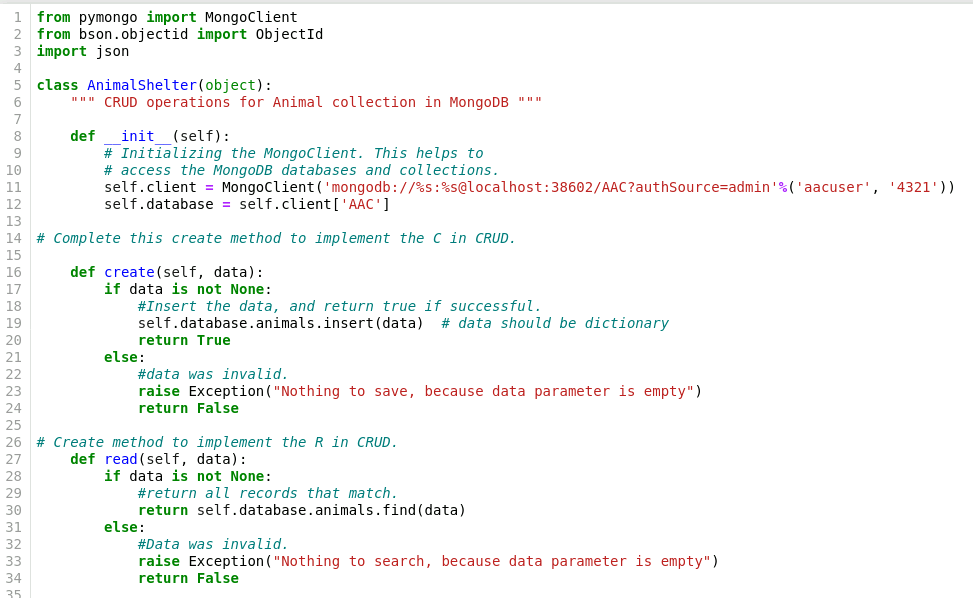
Here is the code “CRUD” functionality in a .py file. There are comments so that each piece can be understood individually.

### 

### 

### Screenshots:

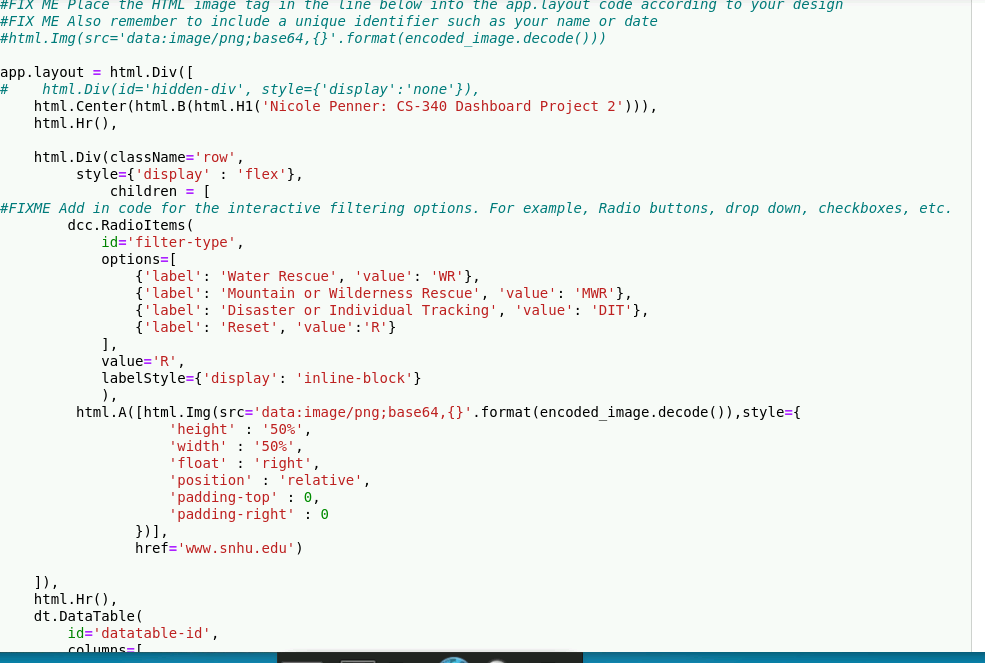
“CRUD” Functionality:

**

Code to create a second graph:

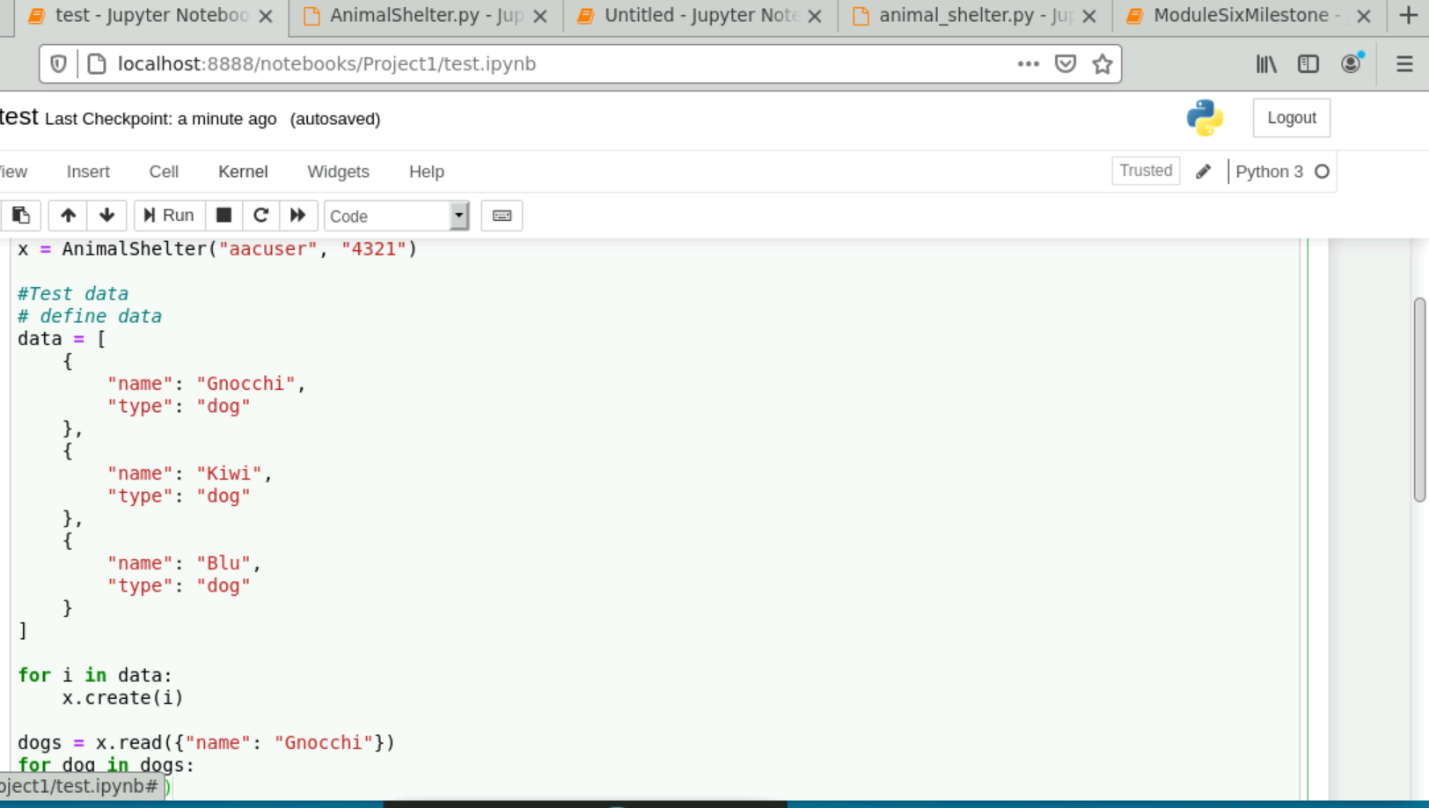


Add code to filter results on the table:



### Tests

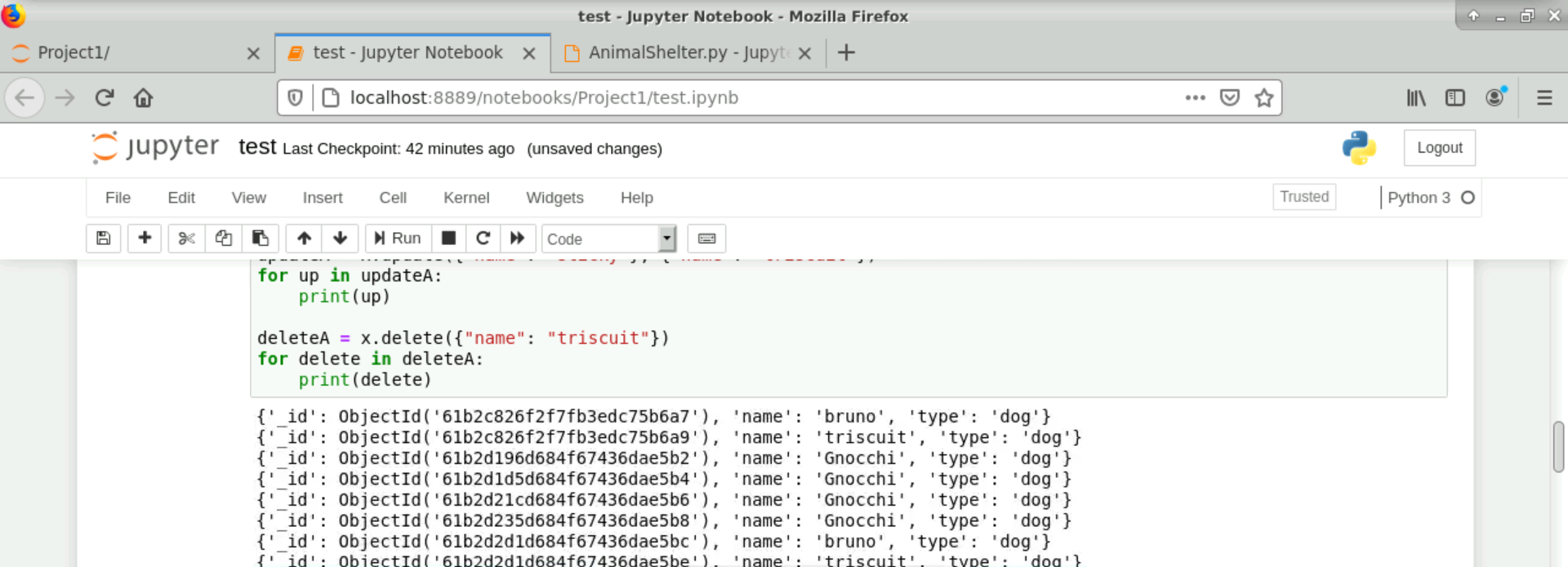
**CREATE:**

**

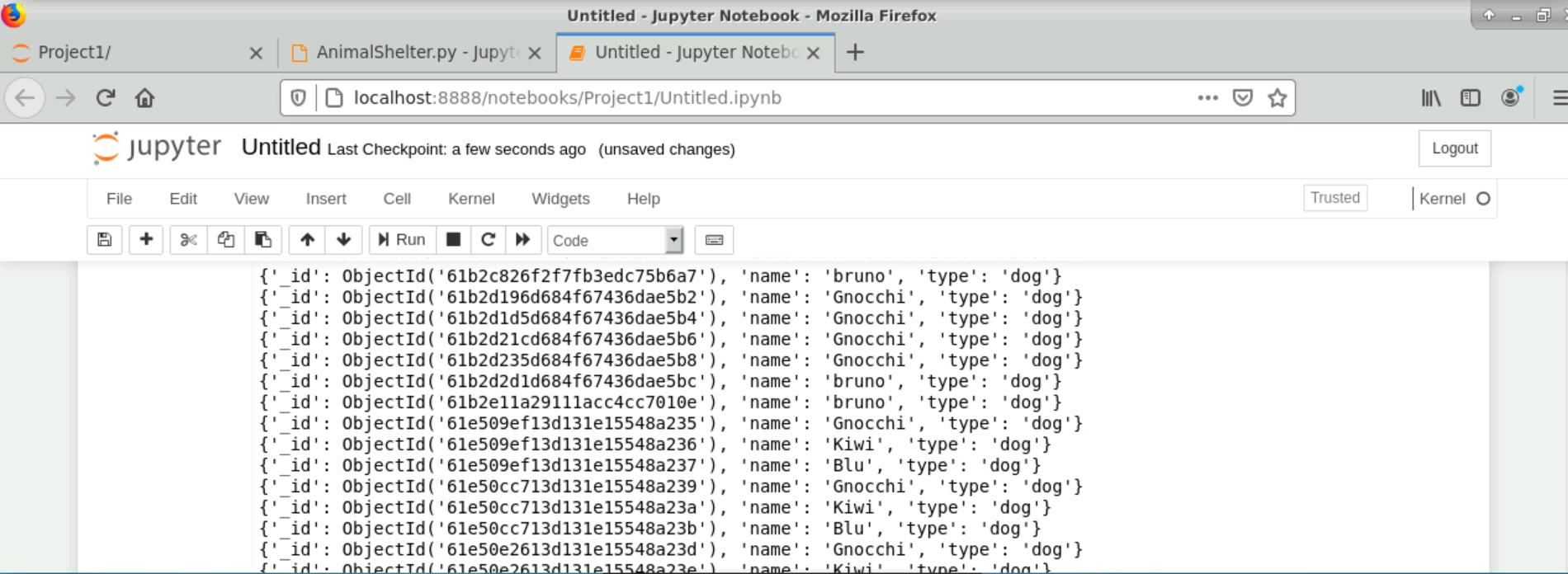
**READ: Shows entries were successfully created and read**

**

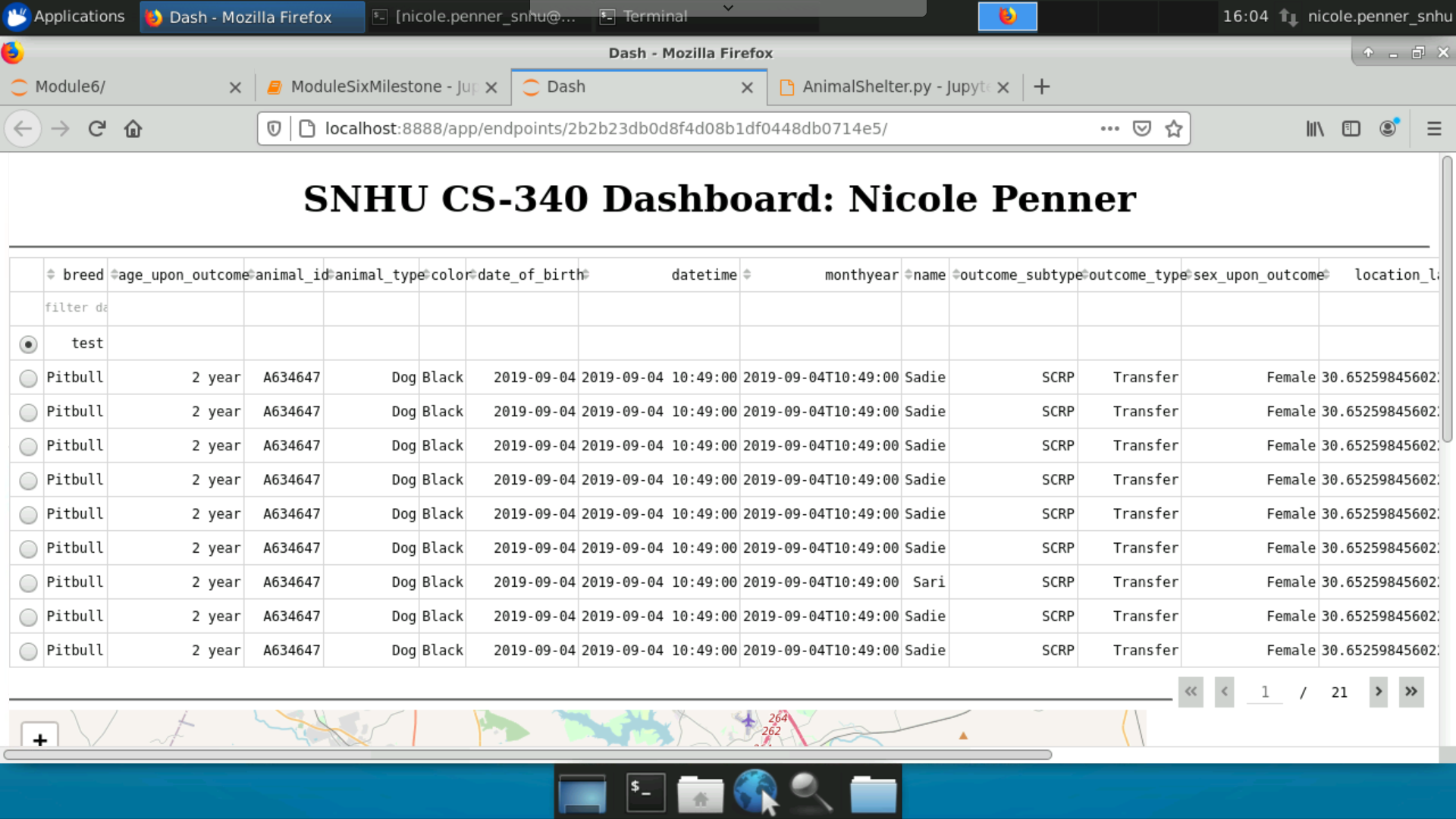
**UPDATE (sticky to triscuit):**

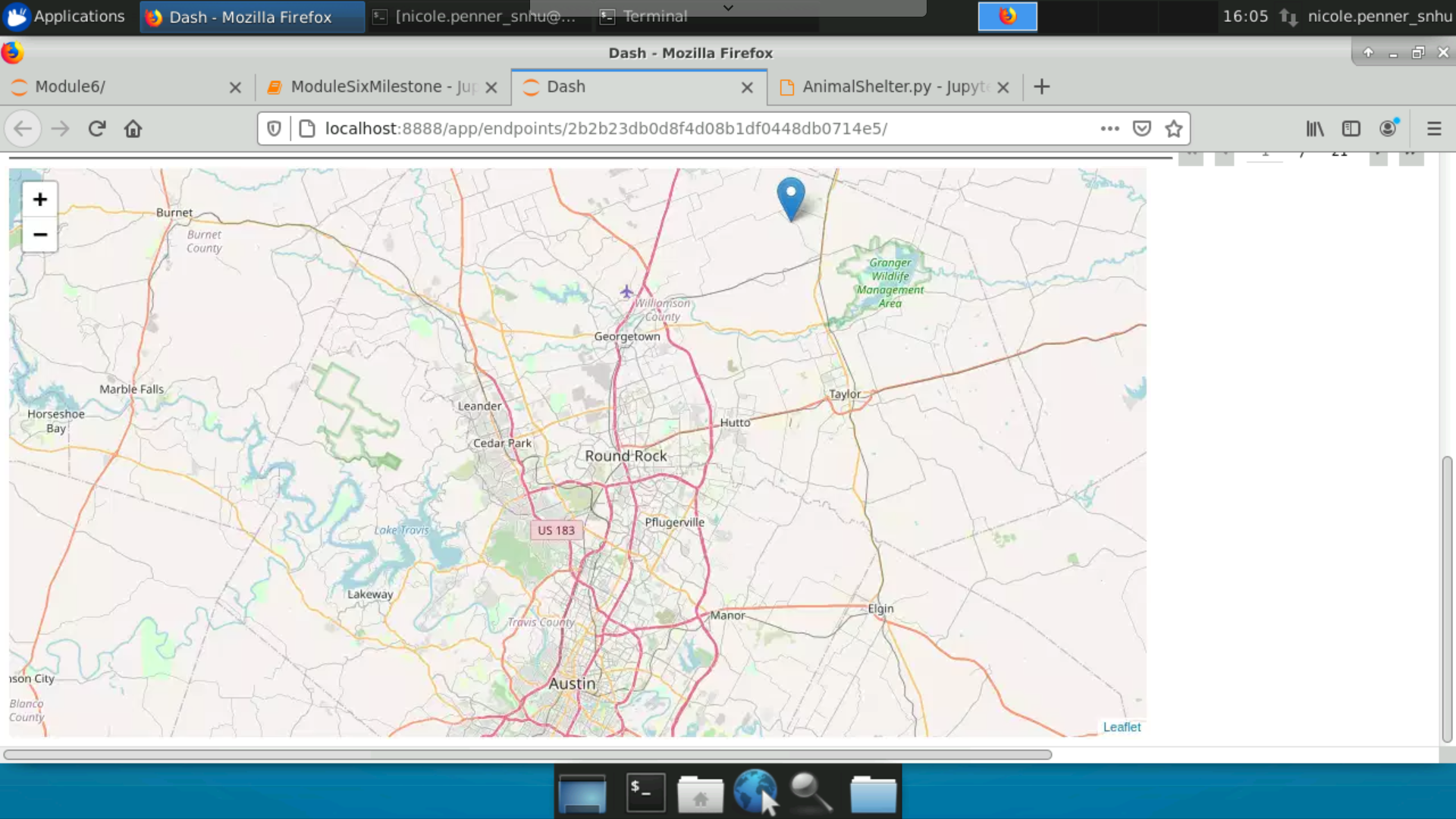
**

**DELETE: Triscuit successfully removed**

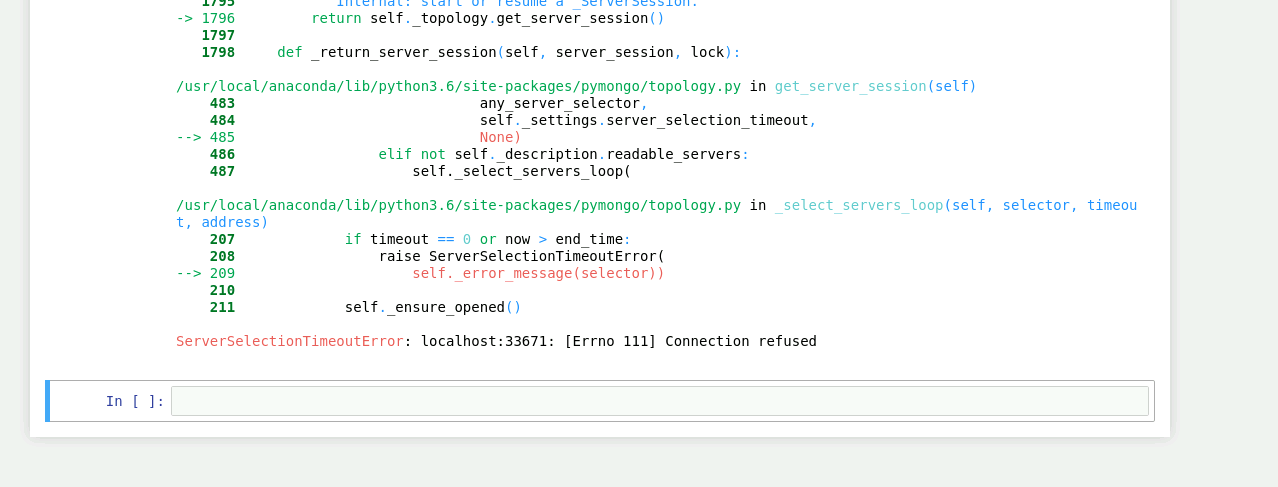
****

**Dashboard Screenshots**







***When creating the dashboard I did keep receiving an error that mongo could not connect:*** **

## Contact

Your name: Nicole Penner, [nicole.penner@snhu.edu](mailto:nicole.penner@snhu.edu)