# CS 340 README Project Two

## About the Project/Project Title

This project was set up for us to build a web-based dashboard for a company called Grazioso Salvare. Grazioso Salvare is a company that identifies dogs that are good candidates for search-and-rescue training. In this we were to create software that could search through databases of five different animal shelters in Austin, Texas with certain search criteria.

## Motivation

The motivation behind this project was to address the critical need for search-and-rescue dogs for training. We need to create software that is effective in finding potential candidates for specific needs, and hope that others will use this software eventually. This software can be helpful in other ways by changing the filter search perimeters.

## Getting Started

To get a local copy up and running, follow these simple example steps:

* Open terminal and run “cd /usr/local/datasets” and make sure the “acc\_shelter\_outcomes.csv” is loaded up and hit enter.
* Make sure your see what database you need to call by using the command “show dbs”
* this should be db- “AAC” and the collection name is “animals”

A screenshot of a computer

Description automatically generated

* go to GITHUB and find the project and copy it from there.
* Once downloaded, open up jupyter notebook upload the project file and find the folder called Project2 and enter in.
* Once in open up the two files a CRUD.py file and the Untitled.ipynb file
* Once you have run the. .ipynb creates a Dash where you can then utilize the dashboard

## Installation

This project consisted of 3 different programs that work together to bring forth a cohesive software for the user. The tools that must be installed for this project to function properly:

* Python
* Pymongo
* MongoDB
* Jupyter Notebook
* Jupyter plotly-Dash
* API that needs to be imported for graphing and such:
  + Base64 (for encoding the logo)
  + Numpy
  + Pandas
* The “acc\_shelter\_outcomes.csv” excel file

**Tools/Components**

**Jupyter Notebook –** Jupyter Notebook is a web-base application that can create, share, and run documents just like any other Python IDE. Jupyter is a simpler version of a IDE that allows the user to break down the code and show quicker visualization of the segment of code.

**MongoDB** - For this project we used MongoDB because it used a dynamic schema which alleviates errors from minor problems. This lets the software be more flexible and helps prevent those errors from occurring and hauling the development process.

**Python** - Python was the selected language for this project because of its versatility. Python has a wide range of tools that can connect the front-end and back-end development, making it easier to create a project. Also, with the ability to build html dashboards with dash libraries makes it the most suitable choice. Plus, we were working on Linux and Python is the preferred language for that OS.

**Plotly-Dash** - This is an API that can be used to create html dashes that centers around JavaScript tools. With these libraries we can create aspects like graphs, charts, maps, and widgets uses for the data tables.

## Usage

When you open these files, you will see a CRUD.py file that has the CRUD functions in it. We will use each of these in a different way and the like so:

* **Constructor**

This will call self and initialize MongoDB and allow us to connect to the database. **Note:** This is where you see the connection variables that you must make sure are connected to the user individual MongoDB. This would be where your password and username would be assigned.

* **Create()**

This will create a record for the database if the new entry meets the requirements, then it will come back true. If this comes back as true, then a new entry will be stored in the database. However, if it does not meet the requirements then it will return False and it will not add the new entry to the database. If there is no argument passed through it throws an exception that will throw an error letting the user know its unsuccessful.

* **Read()**

This function can be used to search the database to find animal entries by specified criteria. Us the print(animals.read({“criteria”: “search”}) we can pull the entries that match what the user is searching for. Unless nothing is passed and then the exception is thrown and the error shows letting the user know there is nothing to read.

* **Update()**

The update function allows us to use the search function to find the entry needed and make changes to that entry and update it in the database. unless nothing is passed and then the exception is thrown and the error shows letting the user know there nothing to update.

* **Delete()**

As it looks, the delete function allows us to search for an entry and delete it, unless nothing is passed and then the exception is thrown and the error shows letting the user know there is nothing to delete.

### Code Example

Below is an example of the CRUD.py file, this snippet is just the Create and Read functions along with all the necessary variables need to connect to my database. Since we were only required to focus on the CR of the CRUD methods. Please refer to the point above for a more detailed breakdown of each of these functions.class CRUD(object):

""" CRUD operations for Animal collection in MongoDB """

def \_\_init\_\_(self):

USER = 'aacuser'

PASS = 'SNHU1234'

HOST = 'nv-desktop-services.apporto.com'

PORT = 30620

DB = 'AAC'

COL = 'animals'

#

# Initialize Connection

#

self.client = MongoClient('mongodb://%s:%s@%s:%d' % (USER,PASS,HOST,PORT))

self.database = self.client['%s' % (DB)]

self.collection = self.database['%s' % (COL)]

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

def create(self, data):

if data is not None:

validate = self.database.animals.insert\_one(data)

if inserted != 0:

return True

return False

else:

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

# Create method to implement the R in CRUD.

def read(self, searchData):

if searchData is not None:

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

else:

raise Exception("Nothing to read, searchData parameter is empty")

return read

# Create method to implement the U in CRUD.

def update(self, searchData, updateData):

if searchData and updateData is not None:

updated = self.database.animals.update\_many(searchData, {“$set”: updateData})

else:

raise Exception("Nothing to update, searchData or updateData parameter is empty")

ruturn updated.modifide\_count

# Create method to implement the D in CRUD.

def delete(self,deleteData);

if deleteData is not None:

deleted = self.database.animals.delete\_many(deleteData)

else:

raise Exception("Nothing to delete, deleteData parameter is empty")

return deleted.deleted\_count

### Tests/Screenshots:

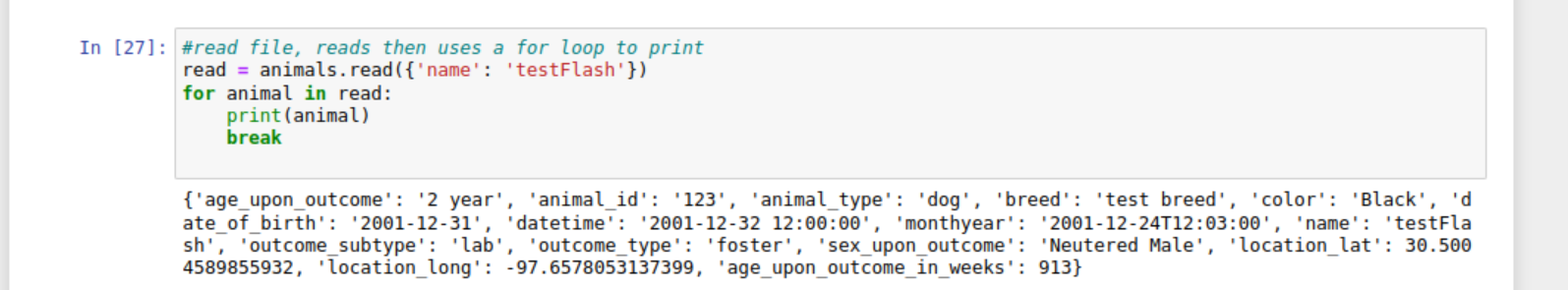
**Using animalShelter.py (this has been renamed to CRUD.py, everything still functions the same)**



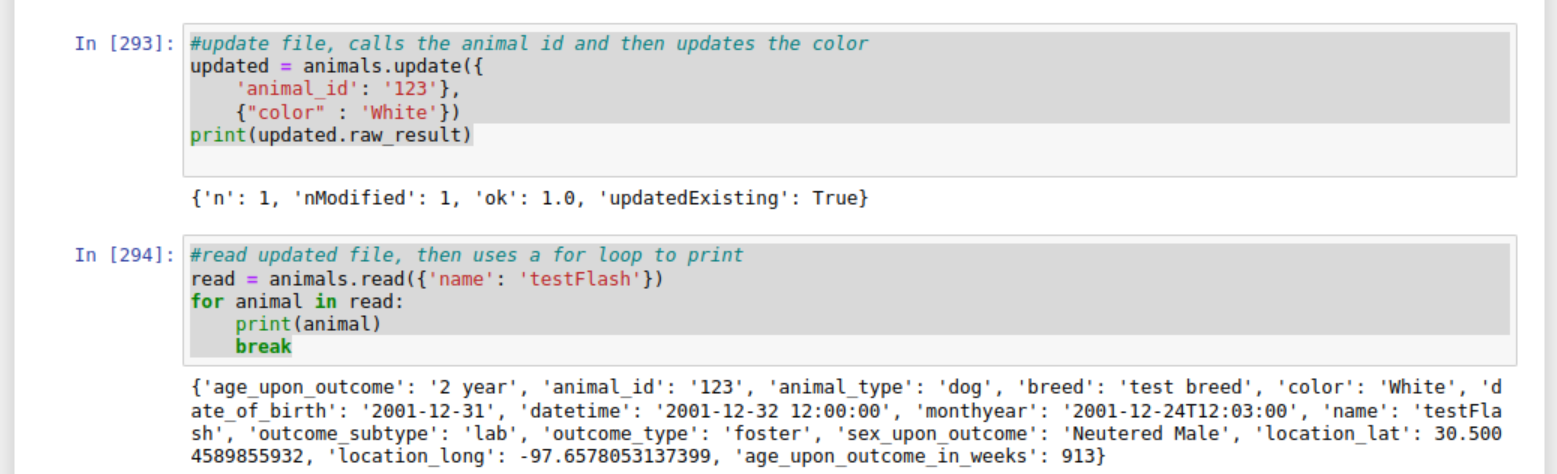
**Create Test:**



**Read test:**



**Update Test:**



**Delete Test:**



**User authentication to the database Screenshots:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

## Product Walkthrough

After the projects have been set up, we then can start making sure the dashboard is working and has what the client requested. While doing this there are key components that were requested for this project to be successful.

**Logo** – We need to have a logo that promotes their website. For my project I made sure to make it front and center on the top of the page.

**Web-link** – We wasn’t directed to create a link, but one would be need when the software is deployed.

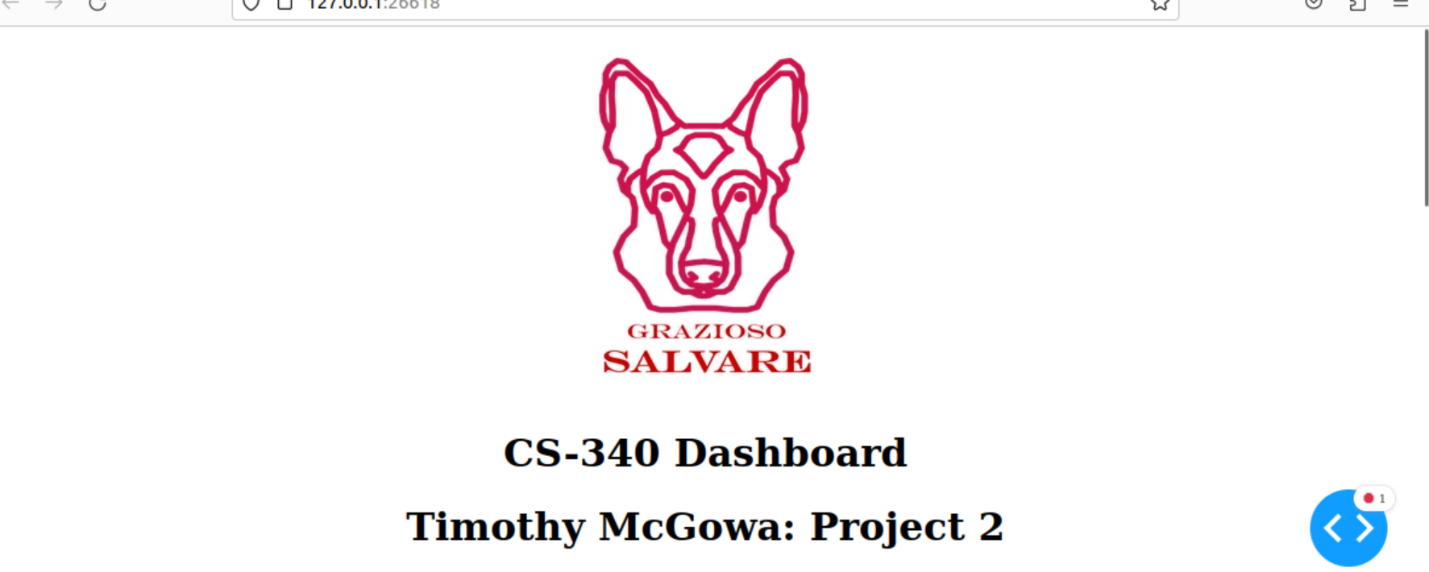
**Data table** This provides the data in a table formation form the animal shelters from the five Animal Shelters in the Austin, TX area.

* Custom interactive filters to search dataset for required needs:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster Rescue or Individual Tracking
  + Reset (returns all widgets to their original, unfiltered options)

**Widgets** – We were asked to create widgets for this projects interface those were:

* Pie chart – that show the availability of the breeds, when filter is selected
* Geo-Map – Show the geo-location of the selected candidate.

**Header/Logo:**

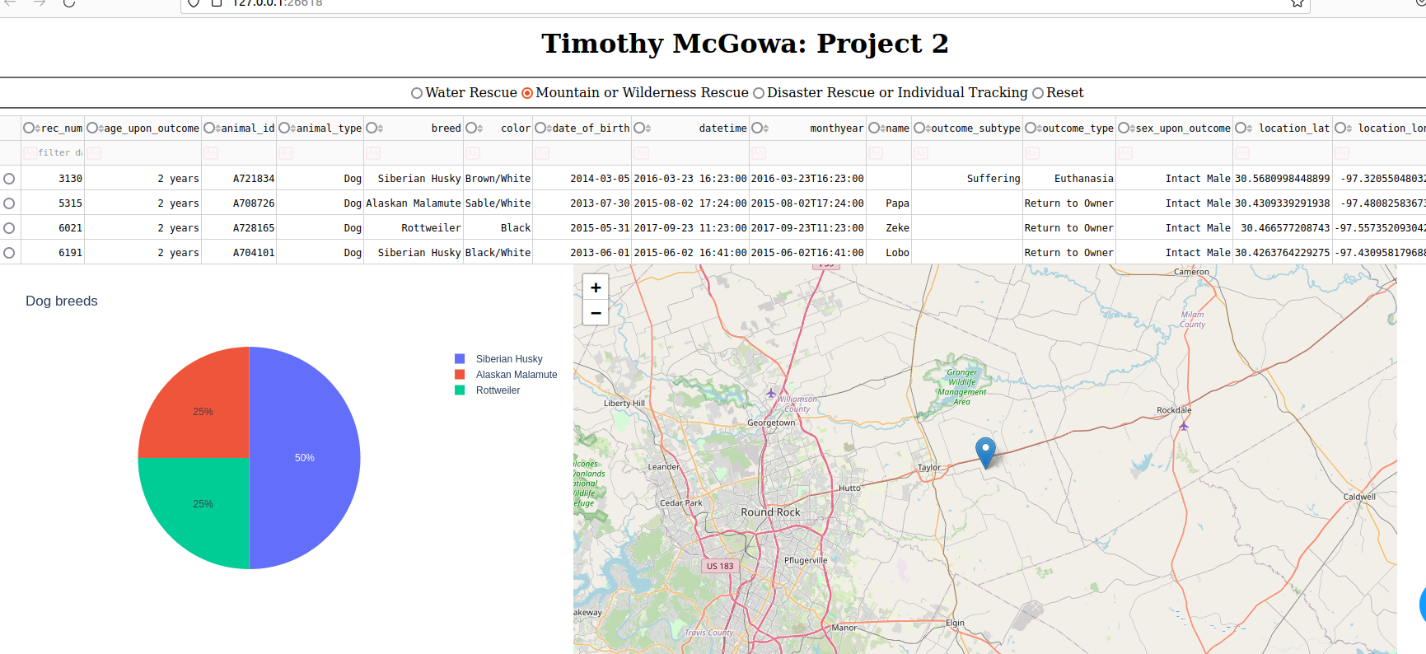


**Water Rescue filter:**

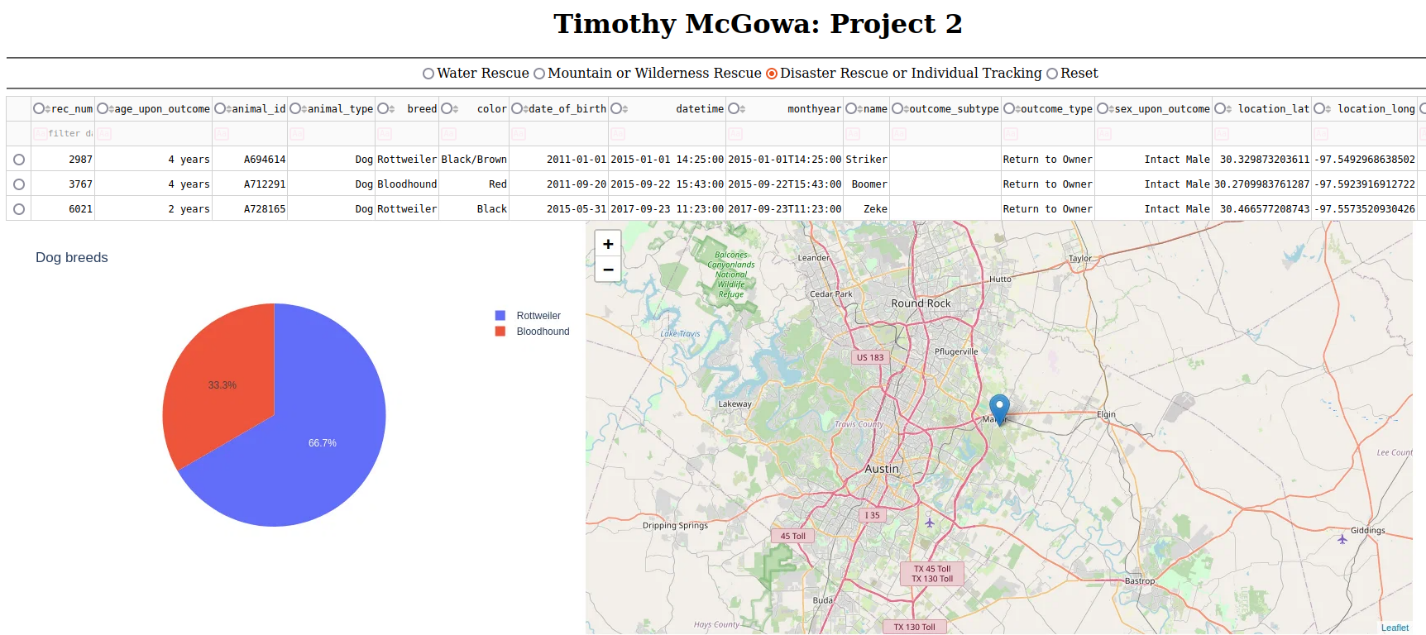
A screenshot of a computer

Description automatically generated

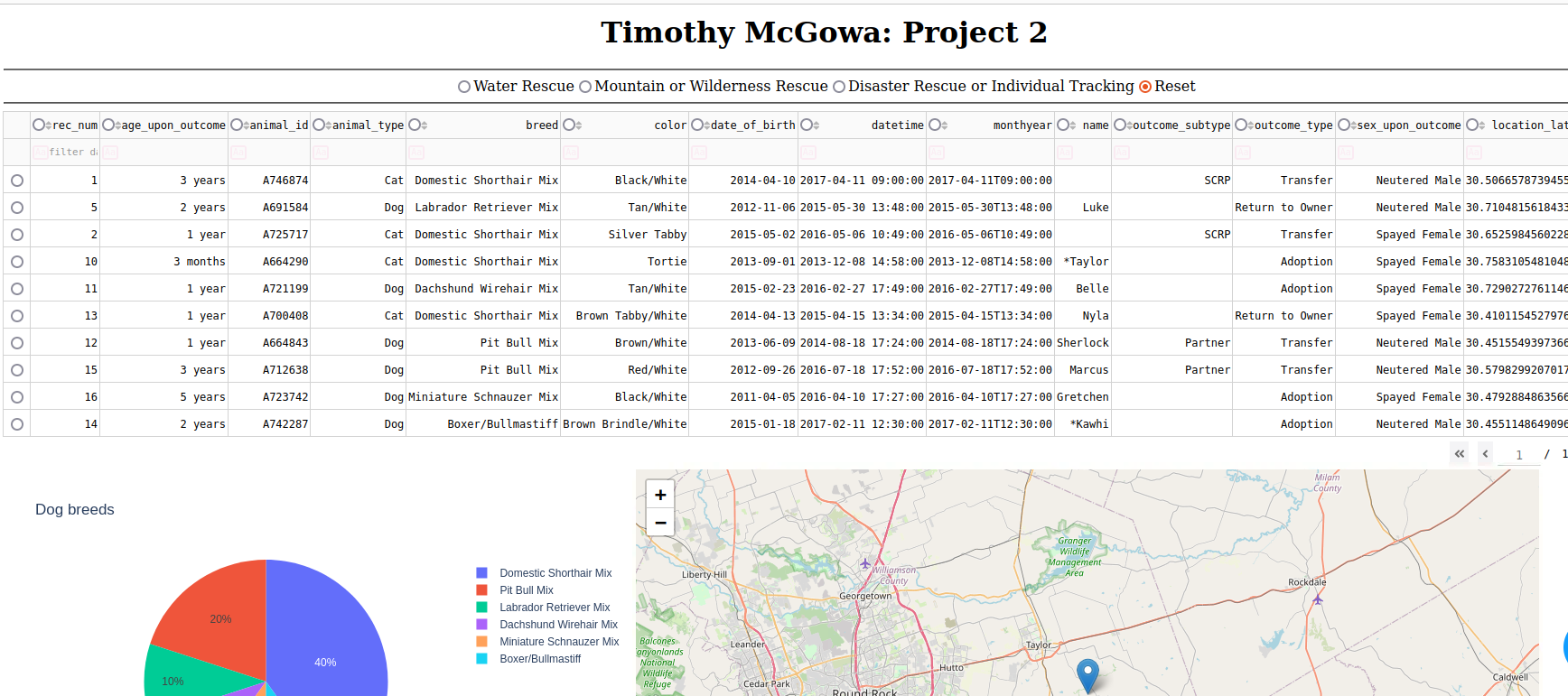
**Mountain or Wilderness Rescue filter:**



**Disaster Rescue or Individual Tracking filter:**

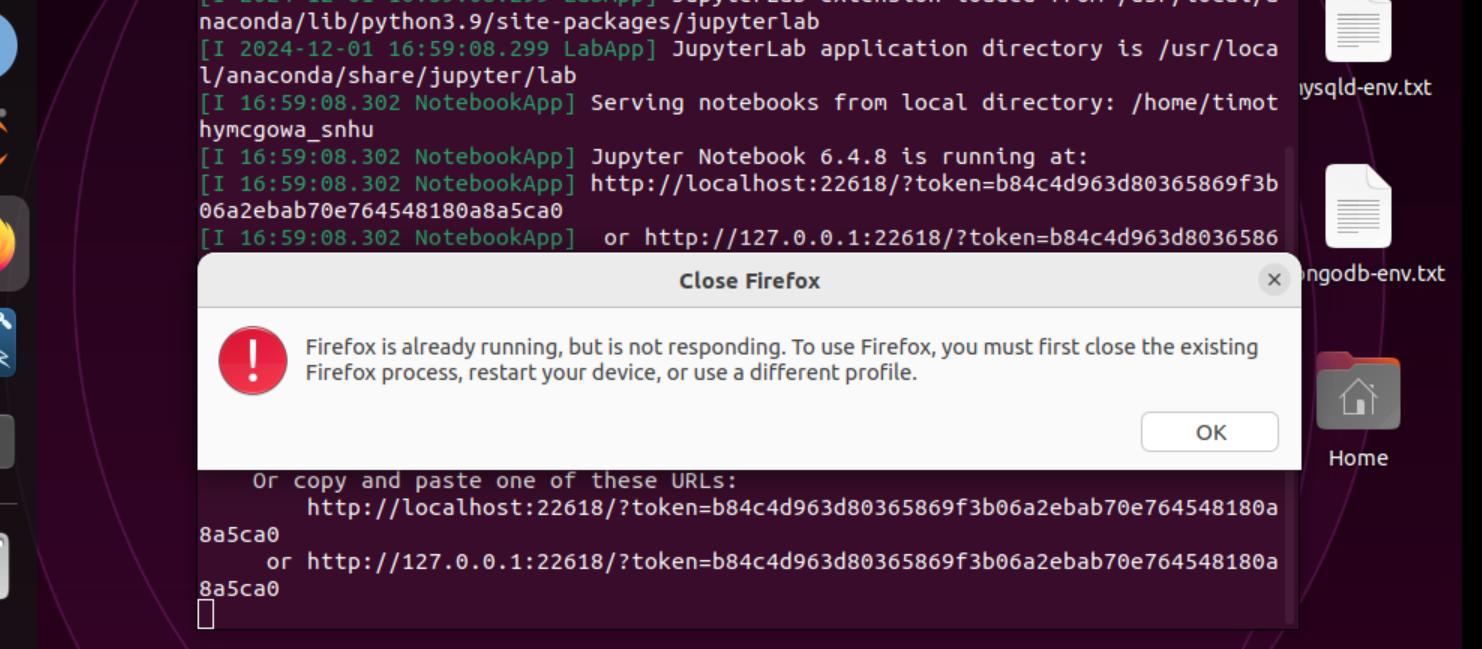


**Reset filter:**



**Challenges**

Throught this project, I had some issues pop up that had my head turning for a few hours. The first problem I had was the project was trying to run a trash file I had on the desktop where I was practicing the code. This kept running an error and it stumped me for a while till I finally noticed what it was doing. The next problem I ran into was the virtual lab, it was slow and laggy. It also kept showing an error (see screenshot below) that kept me from working there. I had to download Notebook on my local machine and create code and run it when it works. Luckly, I started early on, and it didn’t cause me to be late or anything of the sorts, it was just frustrating. The last problem I had was Notebook, developing code in there was hard, it would throw errors but since it wasn’t an editor it wouldn’t show exactly, I opted to still use PyCharm while writing code.



Contact

Your name: Timohty McGowan, 2024