# CS 340 AnimalShelter ReadMe

## About the Project/Project Title

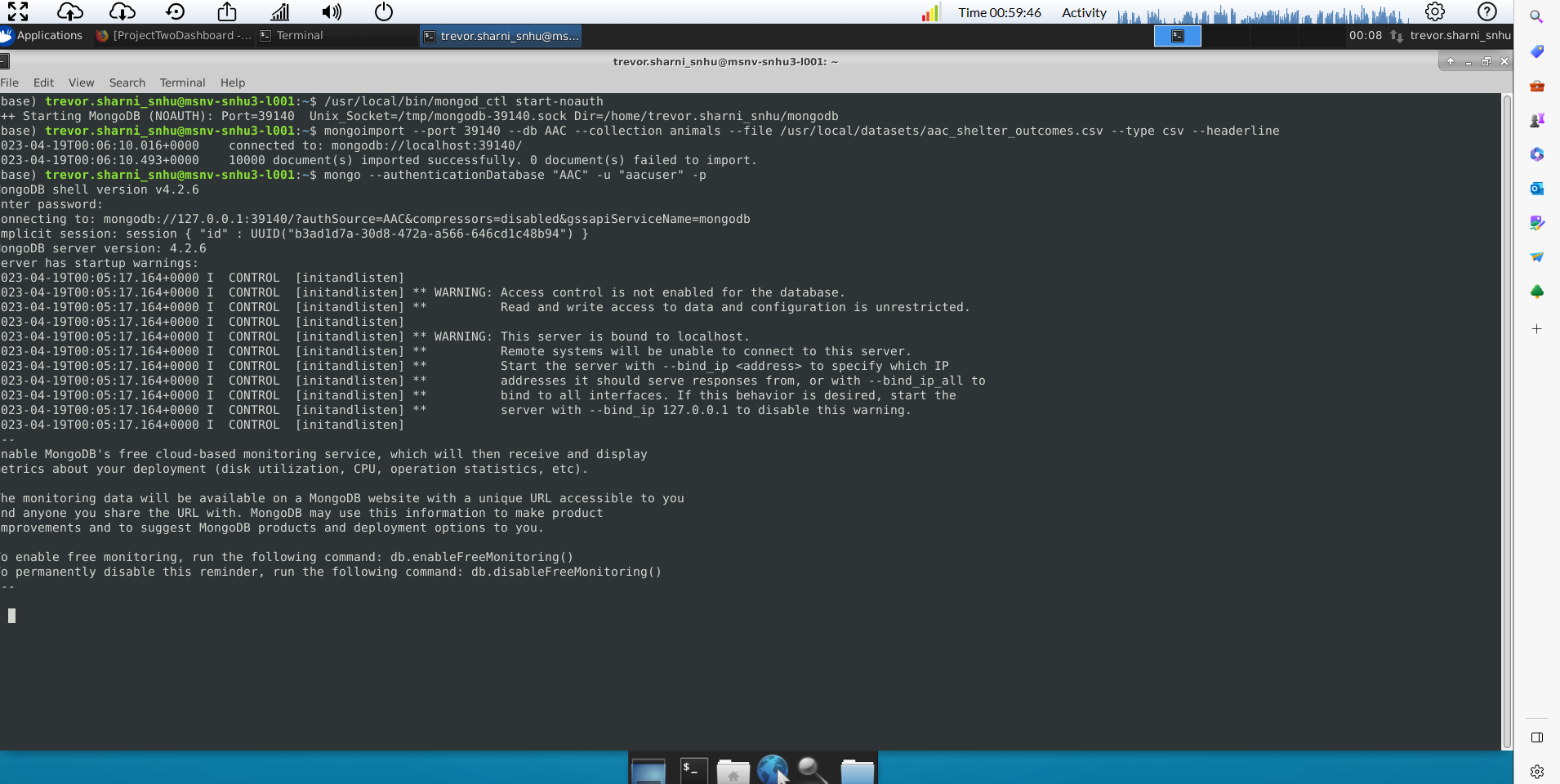
My project is titled AnimalShelter and is a python program which implements Create, Read, Update, and Delete functionality to carry out different tasks within the AAC (Austin Animal Center) database in Mongo. The project is designed to adhere to CRUD principles. This project is a web dashboard designed to display information about animals in a shelter database. The dashboard allows users to filter animals by breed and view their location on a map. It also displays a table of all animals in the database and a pie chart showing the age of animals grouped by breed.

## Motivation

In module 2, we practiced querying within the mongo database command line in linux. To make this process more efficient and simpler, a python script was written to carry out these operations and test their functionality.

## Getting Started

To set up the program, I first needed to import the AAC database data set into mongodb by initializing the environment using the Linux command console and loading the AAC database. Next, I used my username and password to authenticate myself.



## Installation

To use this software, I needed the linux command line, mongodb, jupyter notebook, and python. The python code itself was built within jupyter notebook. Inside the AnimalShelter.py file, I needed to include the pymongo and ObjectId libraries to properly integrate the code with mongo database. During later stages in the development process, I introduced the web dashboard using the Dash Framework. Through Dash, I was able to use built-in components to create tables and maps based on the AAC database. Additionally, I included the Plotly framework so I could include a pie chart in the dashboard to add further visual representation of the data.

* MongoDB Python Driver: <https://pypi.org/project/pymongo/>
* Dash documentation: <https://dash.plotly.com/>
* Plotly documentation: <https://plotly.com/python/>

## Usage

The main purpose of this project is to assist in querying with Mongo Database through the create, read, update, and delete methods in my CRUD python file animal\_shelter.py. MongoDB was chosen as the model component of the development because of its ability to handle large amounts of unstructured data. Given the animal shelter database information is vast and contains a large variety of different data fields, a NoSQL-based database like MongoDB proved to be effective at managing this information.

### Code Example

Below, I have included two groups of screenshots. The first displays my animal\_shelter.py file which contains the necessary imported libraries and the AnimalShelter class. The AnimalShelter class contains a constructor which requires username and password authentication. Additionally, theclass contains create(), read\_all(), read(), update(), and delete() methods.

### Tests

The second group of screen shots contains the test\_scriptPython3.ipynb file with test code which imports my python code and verifies its functionality by testing the create, read, update and delete methods. Both files were created in Jupyter notebook.

### Screenshots (animal\_shelter.py)

*Graphical user interface, text, application

Description automatically generated*

*Text

Description automatically generated*

**(test\_scriptPython3.ipynb)**

* The animal\_shelter.py module is testing using this test script to ensure proper functionality of the methods. First, create a new instance of the AnimalShelter class using the username and password parameters. The **data** variable demonstrates how to create a new animal entry within AAC database.
* Next, the script calls the **read()** method using the recently created animal’s name key-value pair as a parameter. The highlighted object in the screenshot verifies that the new animal object exists in the AAC database.
* Next, the script tests the **update()** method using the animal’s name key-value pair as an identifier, and the object value we desire to change. In this case, I wanted to change Brutus’ id value to MTF216. The output verifies that the ID was successfully changed.
* Lastly, the script tests the **delete()** method using the animal’s name key-value pair as an identifier. The output verifies that the object was successfully deleted.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

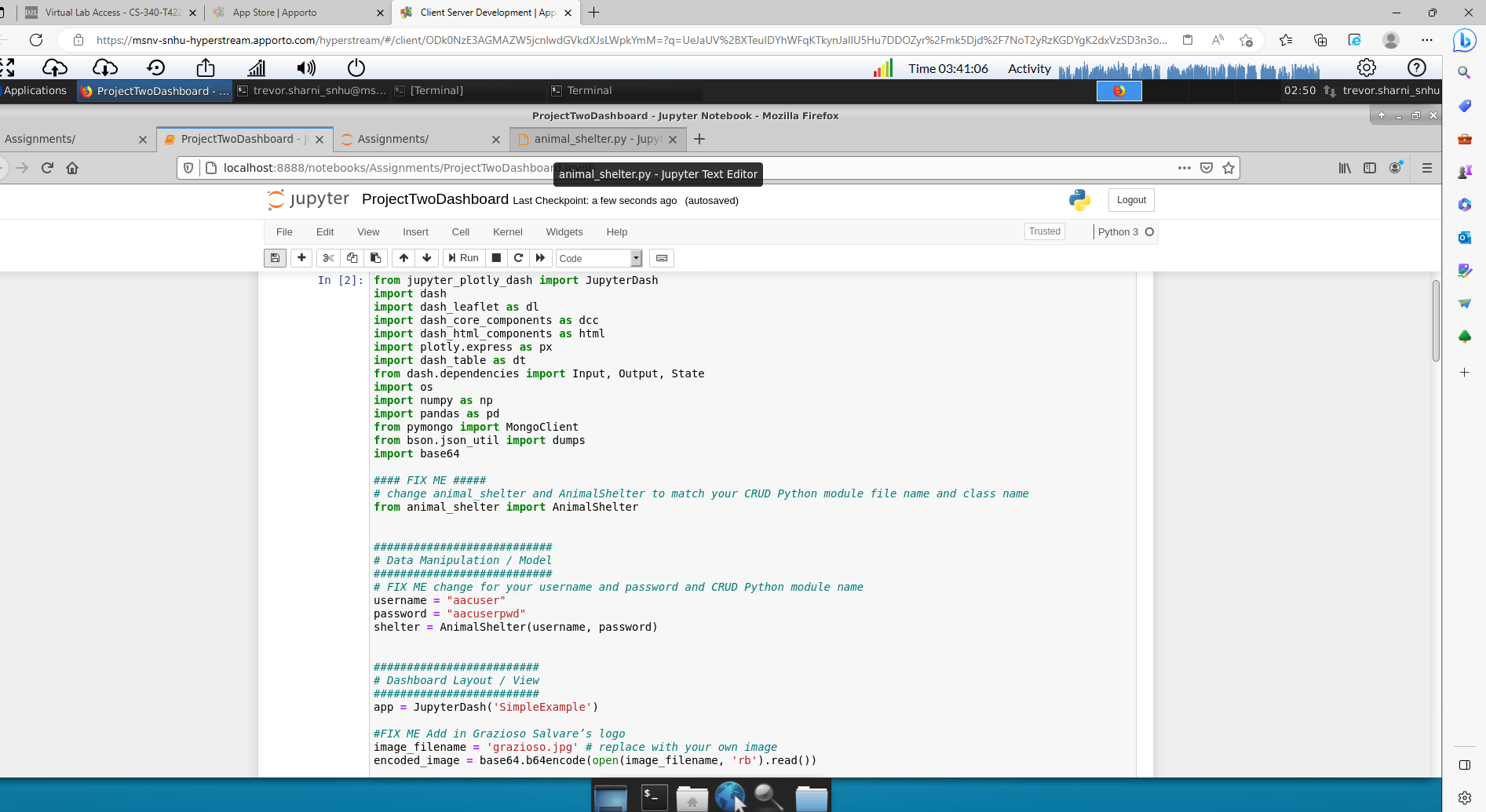
Description automatically generated

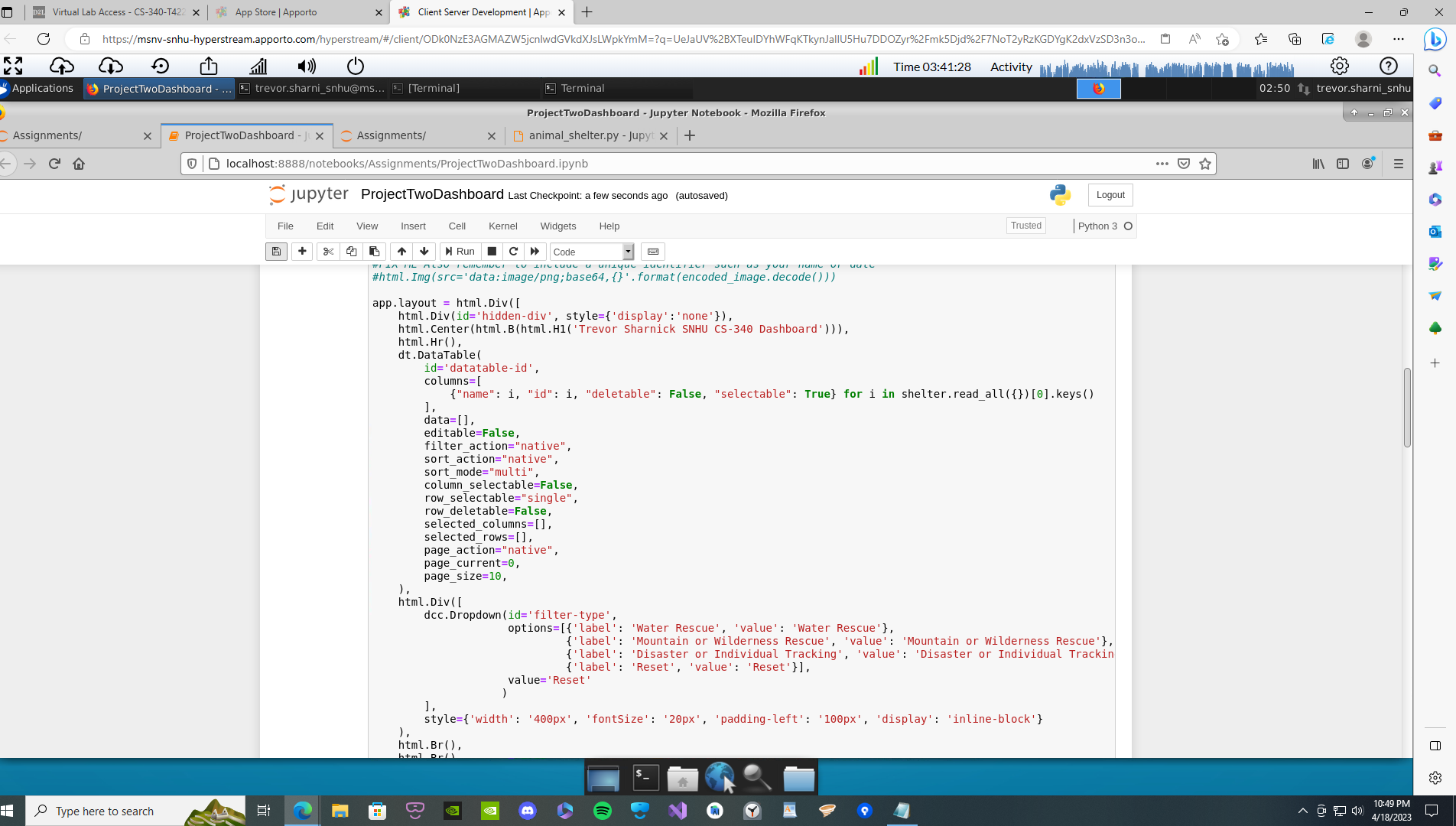
Text

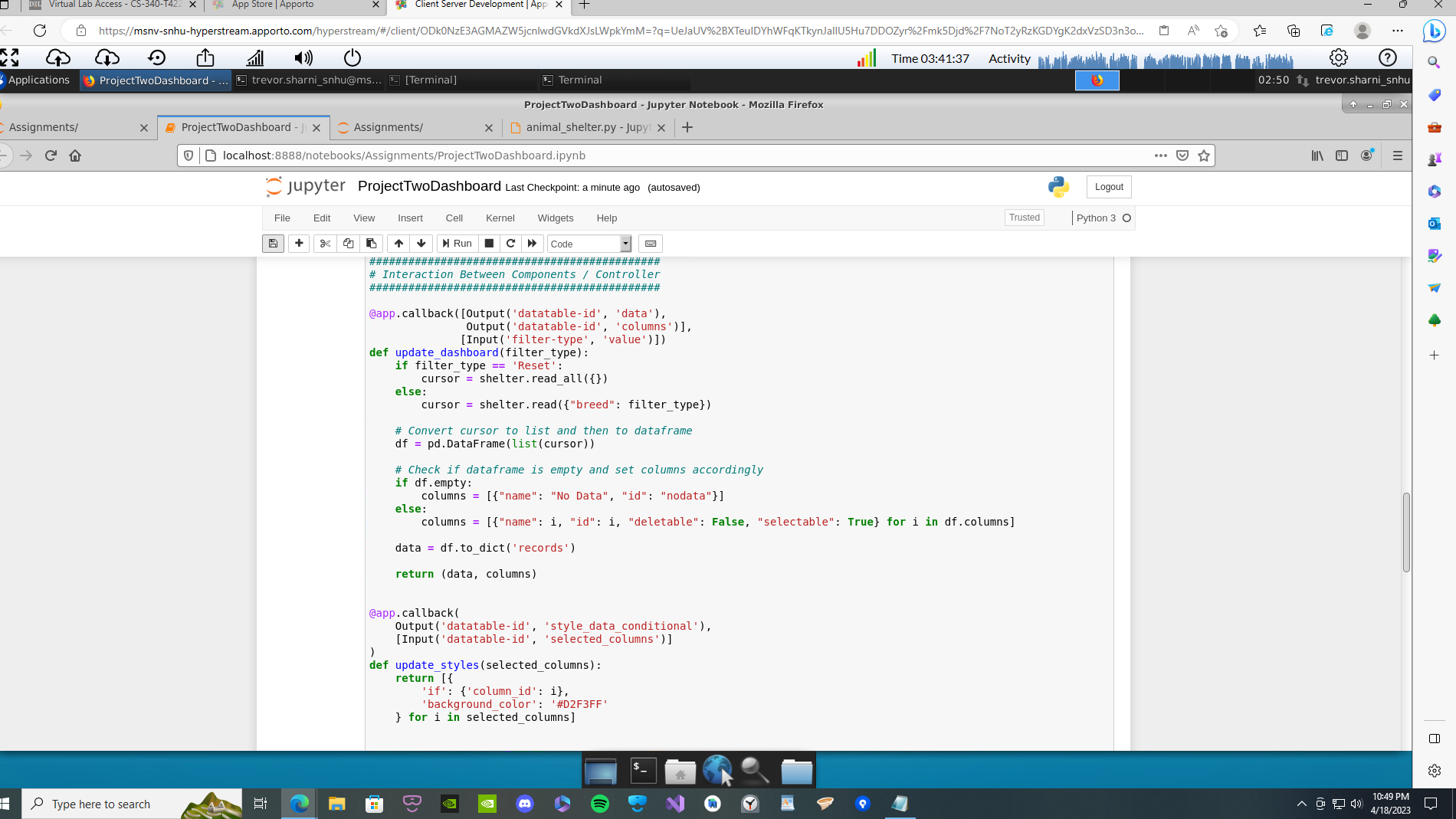
Description automatically generated

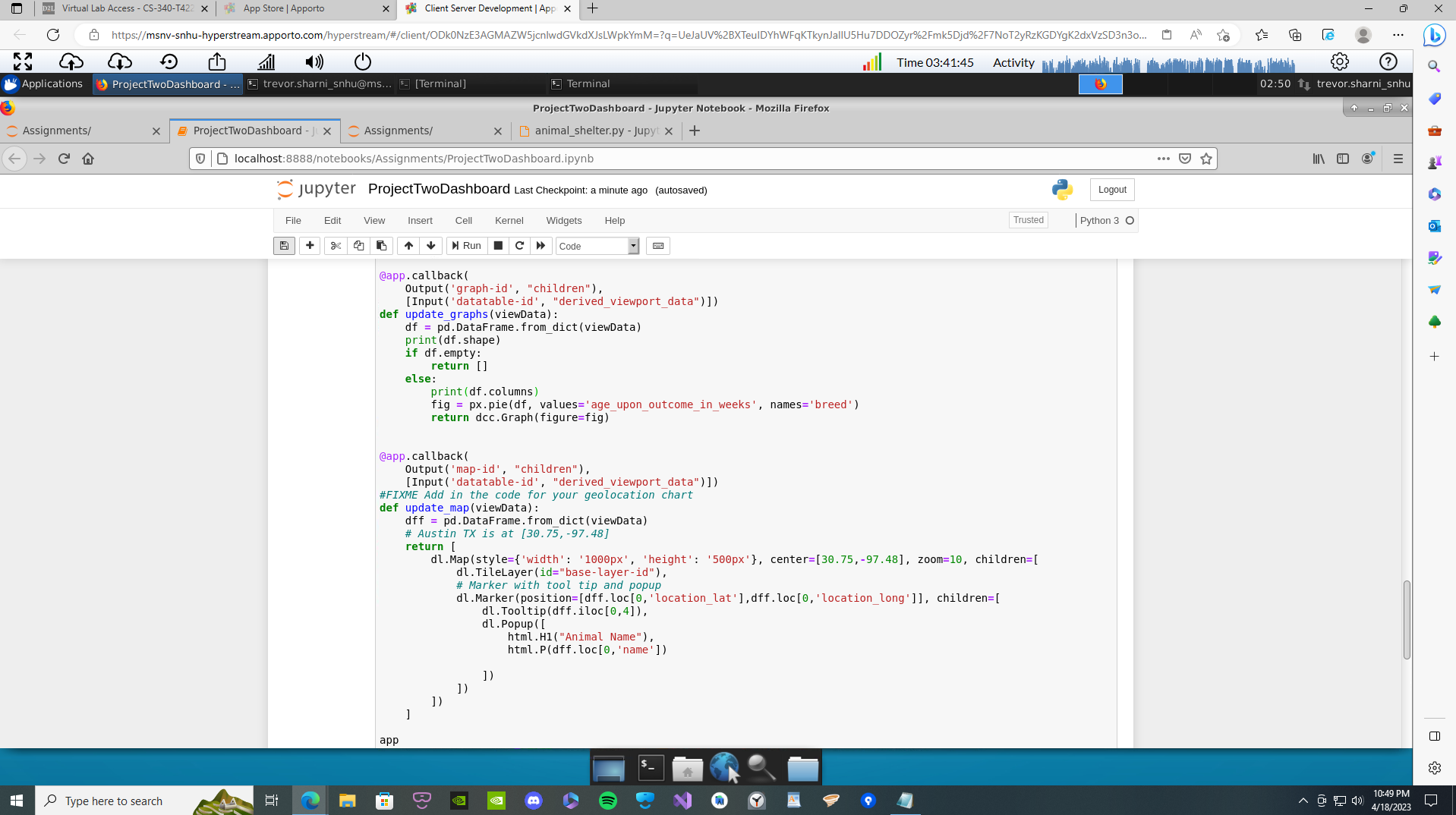
**Creating the Dash Framework:**

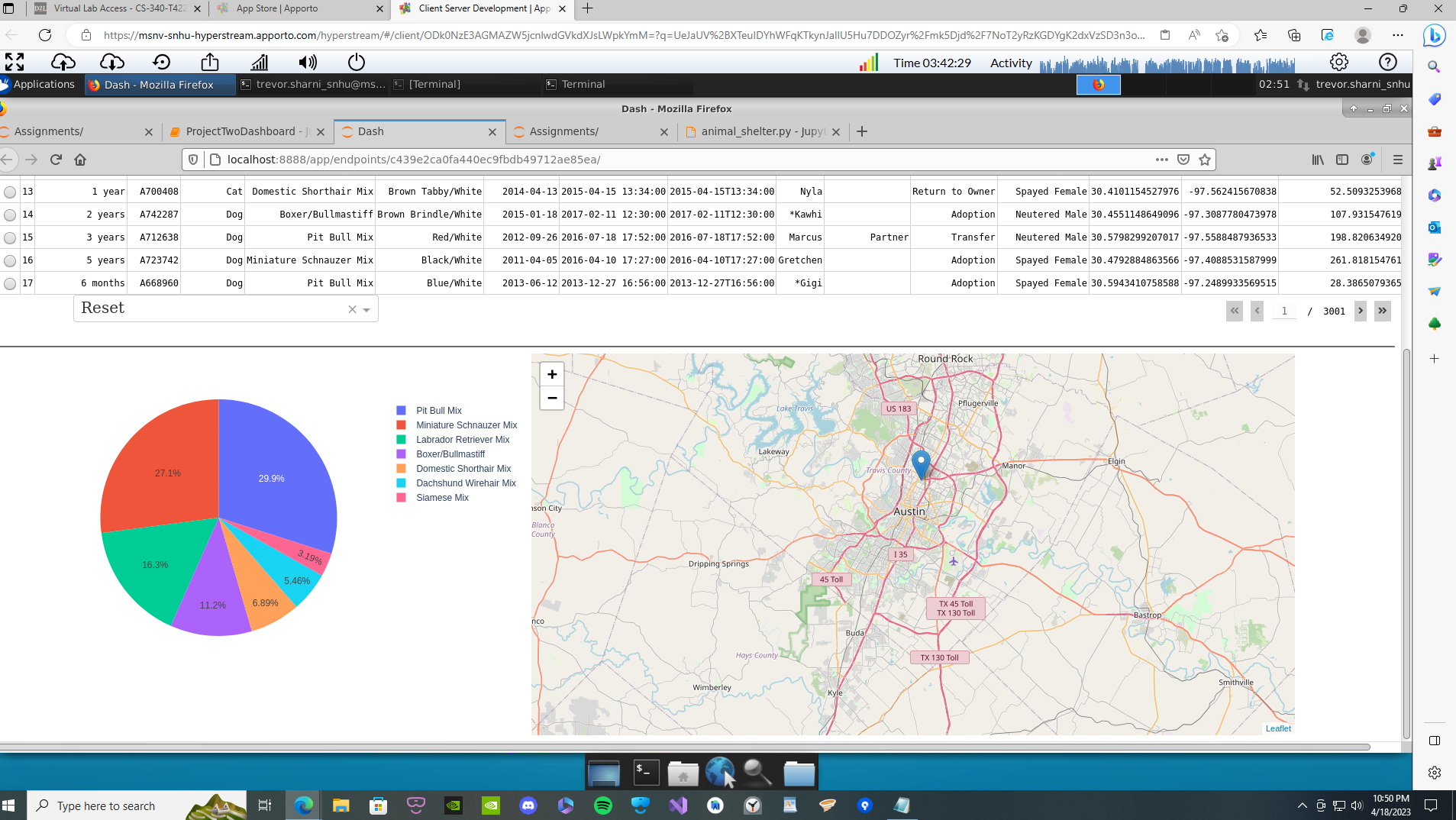
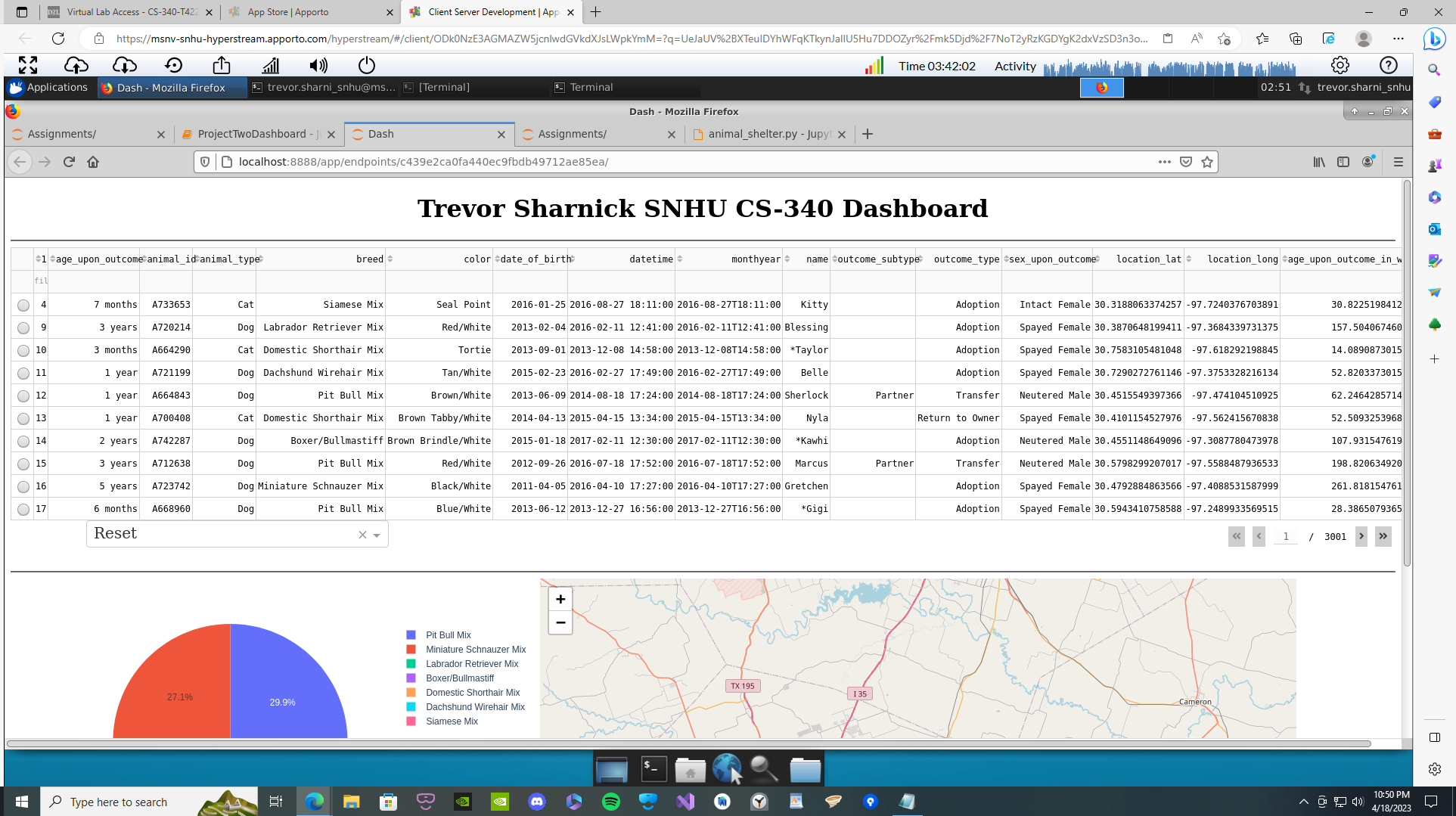
Dash is a Python framework for building analytical web applications. It is used in tandem with Python and has a wide range of built-in components for creating graphs, tables, maps, and charts. It also utilizes a call-back system that allows for real-time updating of the dashboard.











**Challenges faced during the development process:**

When building this program, I faced a great deal of challenges. First, I received numerous errors relating to the dataset being too large to categorize into a table without crashing the application. Upon further inspection, I discovered that my update\_map() method was not properly collecting data from the specified region (Austin TX) but rather, it did not have any constraints. Therefore, the dataset was too large to load given the system constraints. Additionally, the method for filtering the database data into different categories required a great deal of trial and error as well. Upon filtering data results into one of the specified data fields, exiting the filter didn’t initially re-update the table to match the dataset. To fix this, I implemented a reset button tool which reloaded the initial dataset without filtering.

## Contact

Your name: Trevor Sharnick