# CS 340 README

## About the Project/Project Title

This project is a combination of a backend python module that interacts with MongoDB through pymongo and a frontend web server written in python using Dash. With interaction between both elements, the project provides for filtering, sorting, and visualization of data for viewing from a simple webpage that can be arranged using html.

## Motivation

The specific use case of the project authors is to provide Grazio Salvare with a method of parsing and evaluating data accumulated from Austin Animal Shelters for the purpose of finding dogs suitable for rescue training. This use case necessitated the ability to filter data by various field values simultaneously and also visualize the location data provided in the database.

## Getting Started

To run this project, several dependencies are required:  
\* Python 3

\* \* The pymongo module

\* \* The bson.objectid module

\*\* Dash

\*\* numpy

\*\* pandas

\*\* matplotlib.pyplot

\* A running MongoDB server

\* Jupyter

The python script is provided in a Jupyter notebook, and may be run inside the Jupyter Notebook program.

Note that this file requires the animal\_shelter\_crud.py module to run, and it is recommended they remain in the same folder.

## Installation

To install the required dependencies, one can use Python pip

```

pip install notebook

pip install pandas

pip install dash

pip install numpy

python -m pip install pymongo

python -m pip install -U matplotlib

```

Then, you can download the source files animal\_shelter\_crud.py and Jupyter Notebook project from this repository and run them from inside the Jupyter Notebook program, pressing the ‘Run’ button.

You can open directly to the .ipynb file from the terminal by typing

```jupyter notebook /path/to/ProjectTwoDashboard.ipynb```

## Usage

Below are some examples of the code and how it is used on the SNHU MongoDB server to manage document entries.

### Code Example

The create and query methods comprise the majority of the work being performed on the database:

```

# Create a document in the database with the insert\_one function

def create(self, data):

if data is not None:

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

return result.acknowledged

else:

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

return false

# Read from the database using the find function

def query(self, data):

if data is not None:

result = list(self.database.animals.find(data))

return result

else:

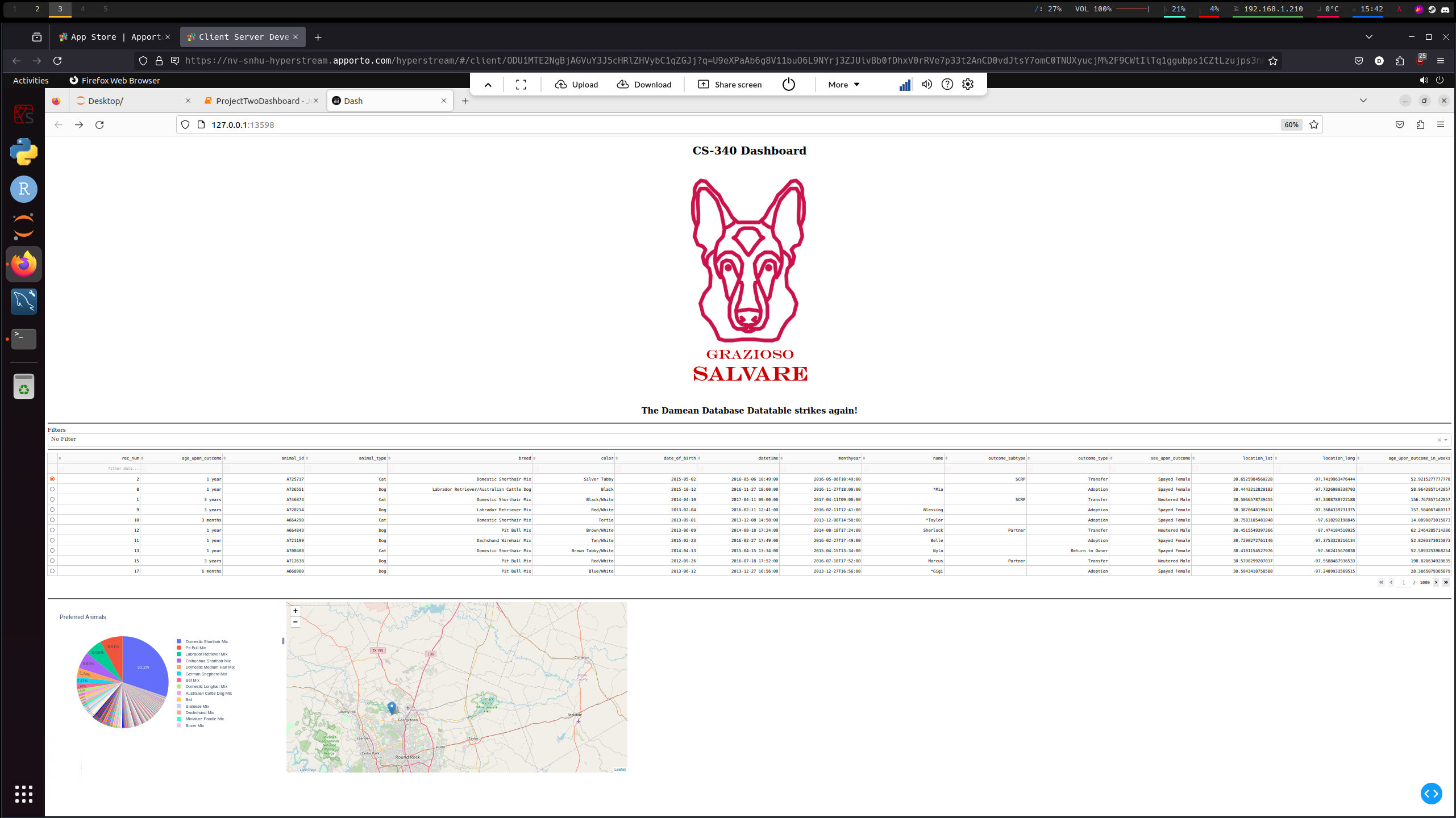
raise Exception("Empty data argument")

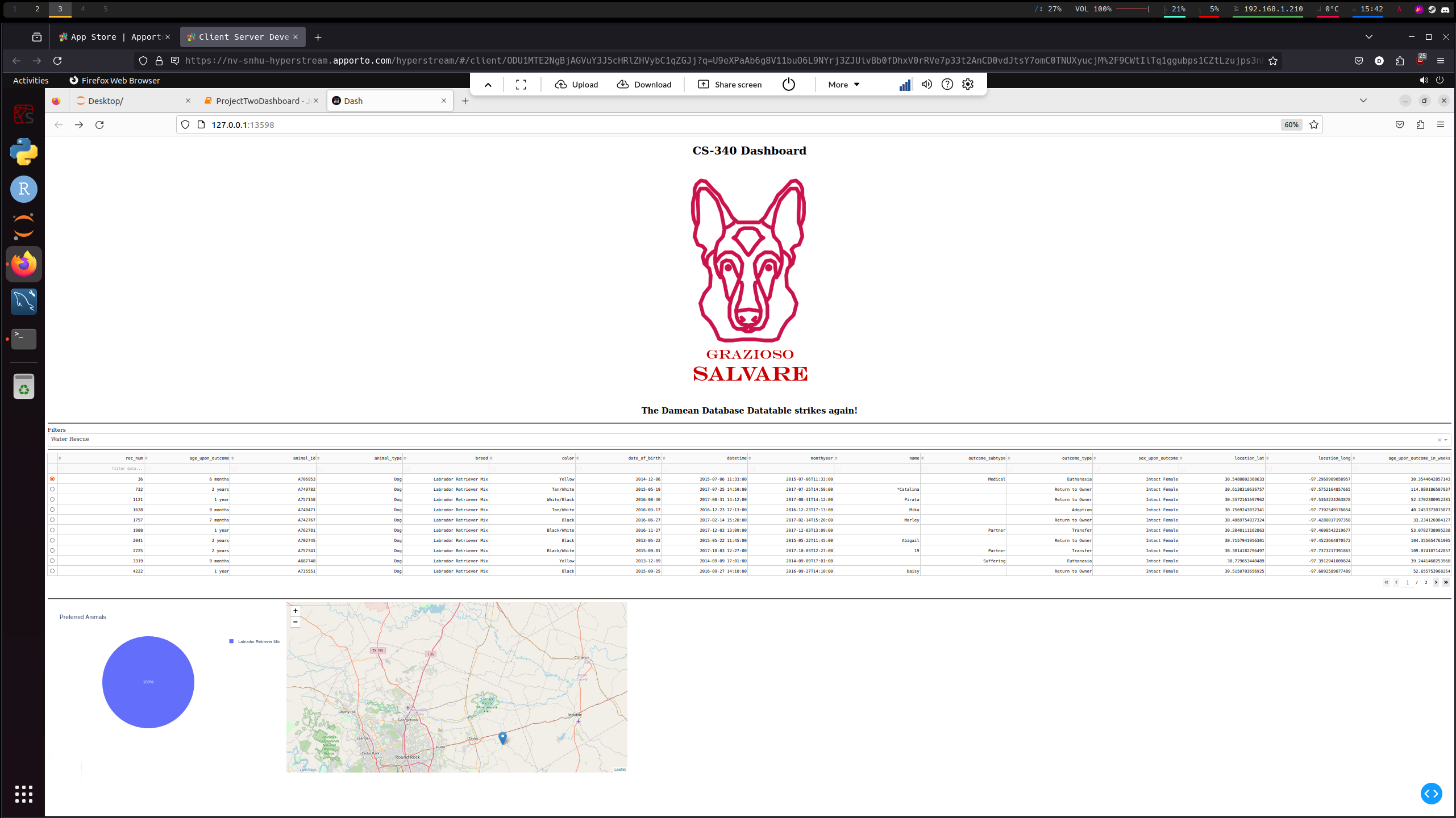
return []

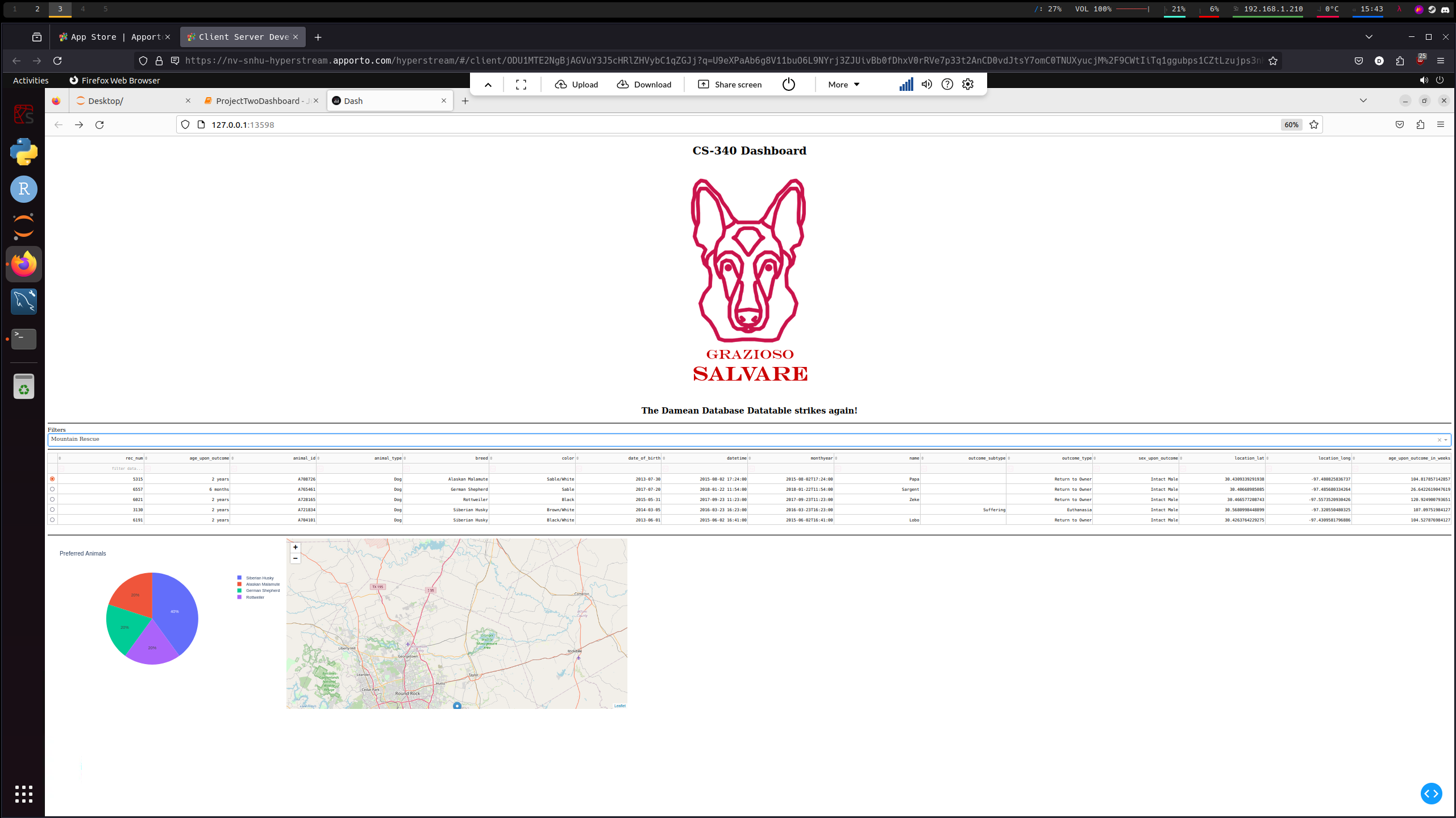
```

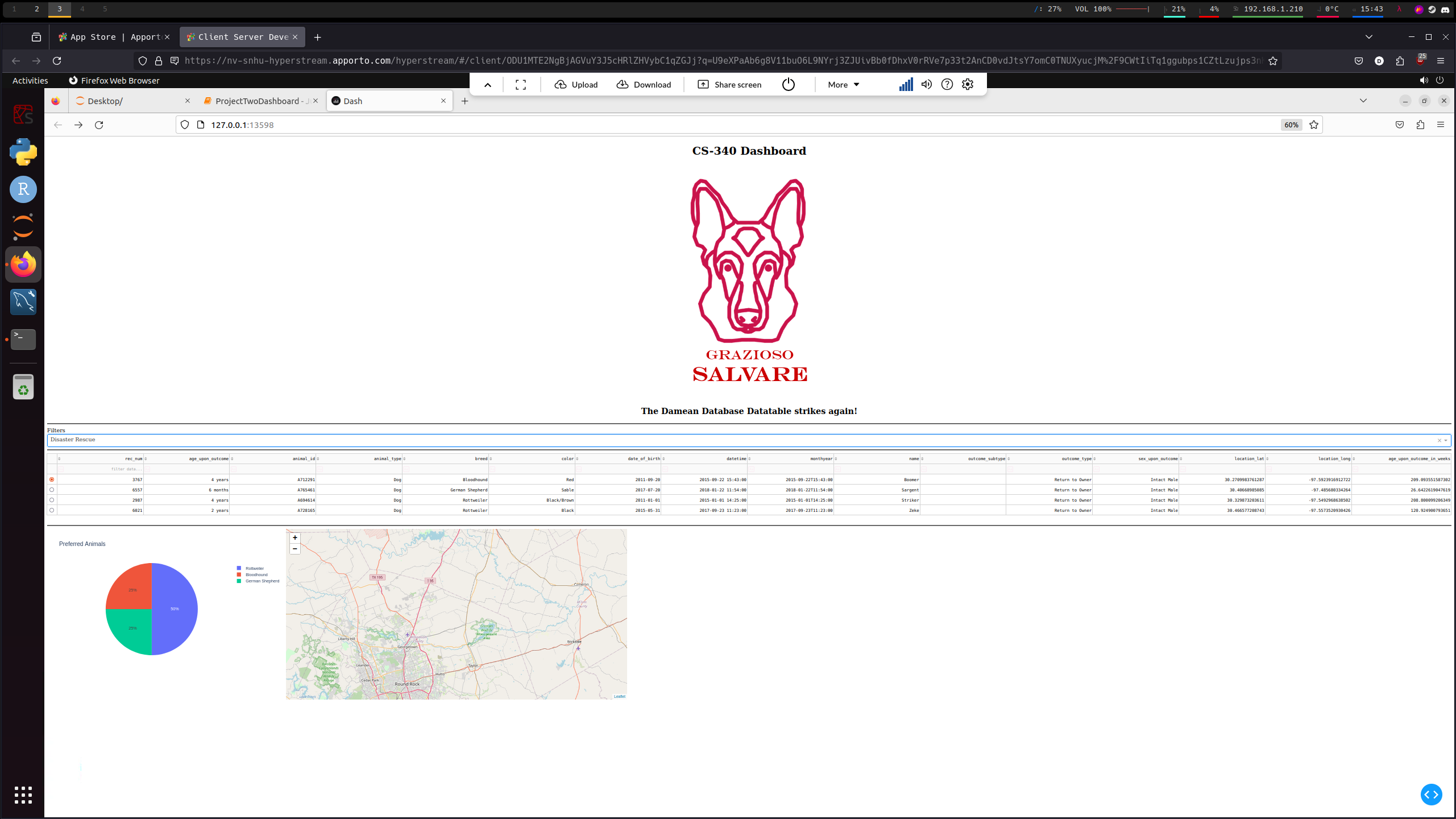
The module creates a class with four methods used to update the database: create, query, update, and delete. These methods will always affect many documents when applicable. Update and delete return the number of documents that they affect. Each method takes as an argument a dictionary to insert, find, or delete matching documents for. The update method takes both a dictionary argument to find and a dictionary argument of updated values for given documents.

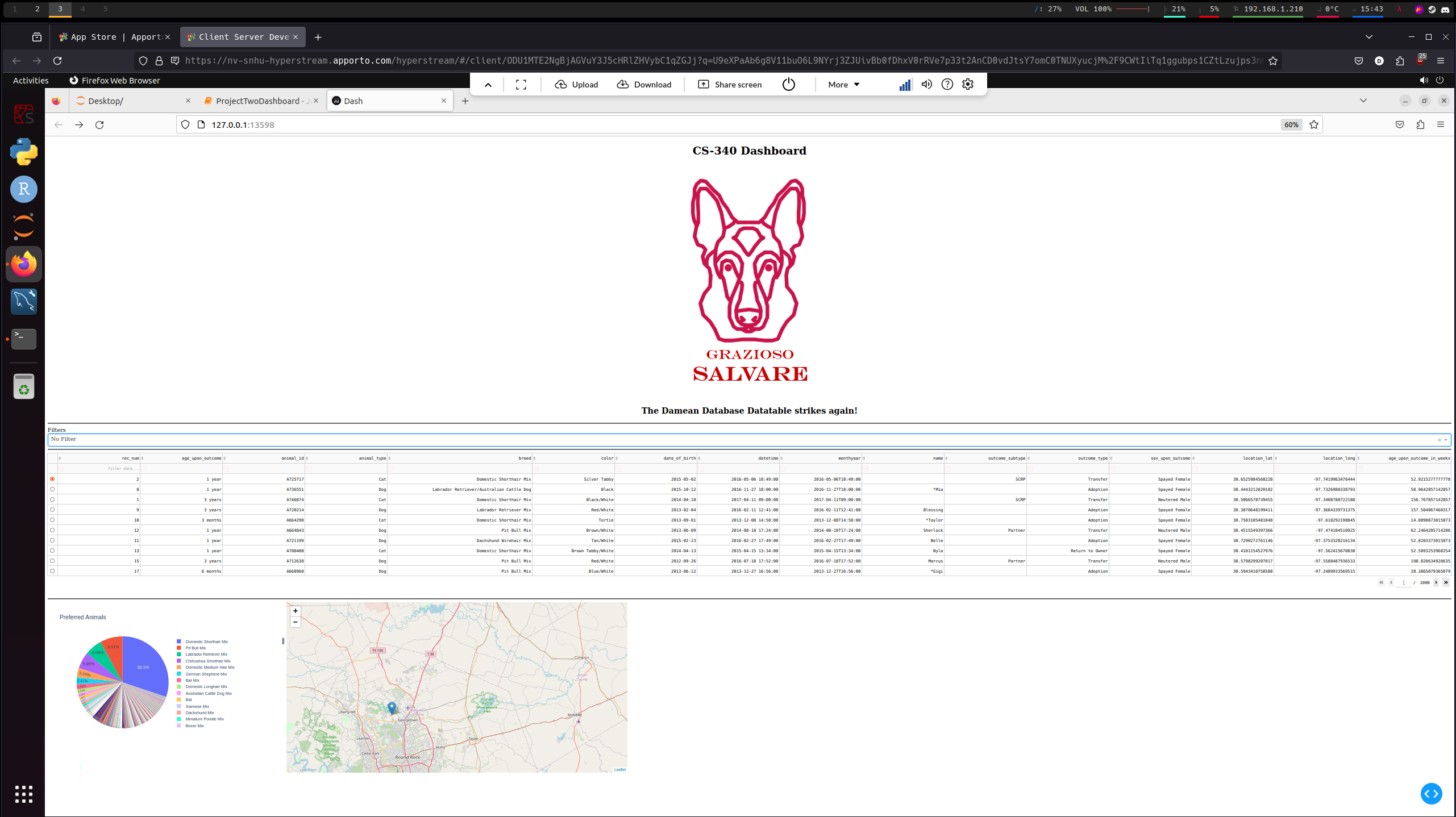
### Screenshots











## Contact

Damean Murphy-Short

damean.murphyshort@snhu.edu