# CS 340 README

## About the Project/Project Title

*Title: AAC Crud*

*This project exists to supply CRUD functionality for an animal shelter database. The finished project will include a user interface to interact with data stored in a database and powered by Mongo DB. The project will be used by Austin Animal Shelter to retrieve and update data related to breed, age, sex, and status of animals in the shelter.*

## Motivation:

*This project exists to supply a reusable library to create and read documents in a Mongo database. This solves the problem of having to rewrite code that creates and* *utilizes the MongoDB client and provides a simpler interface to interact with a known animal shelter database. The Austin Animal Shelter will use this tool to query and update information related to the animals housed in the shelter. MongoDB will power the user-friendly browser-based dashboard for the employees to interact with the database.*

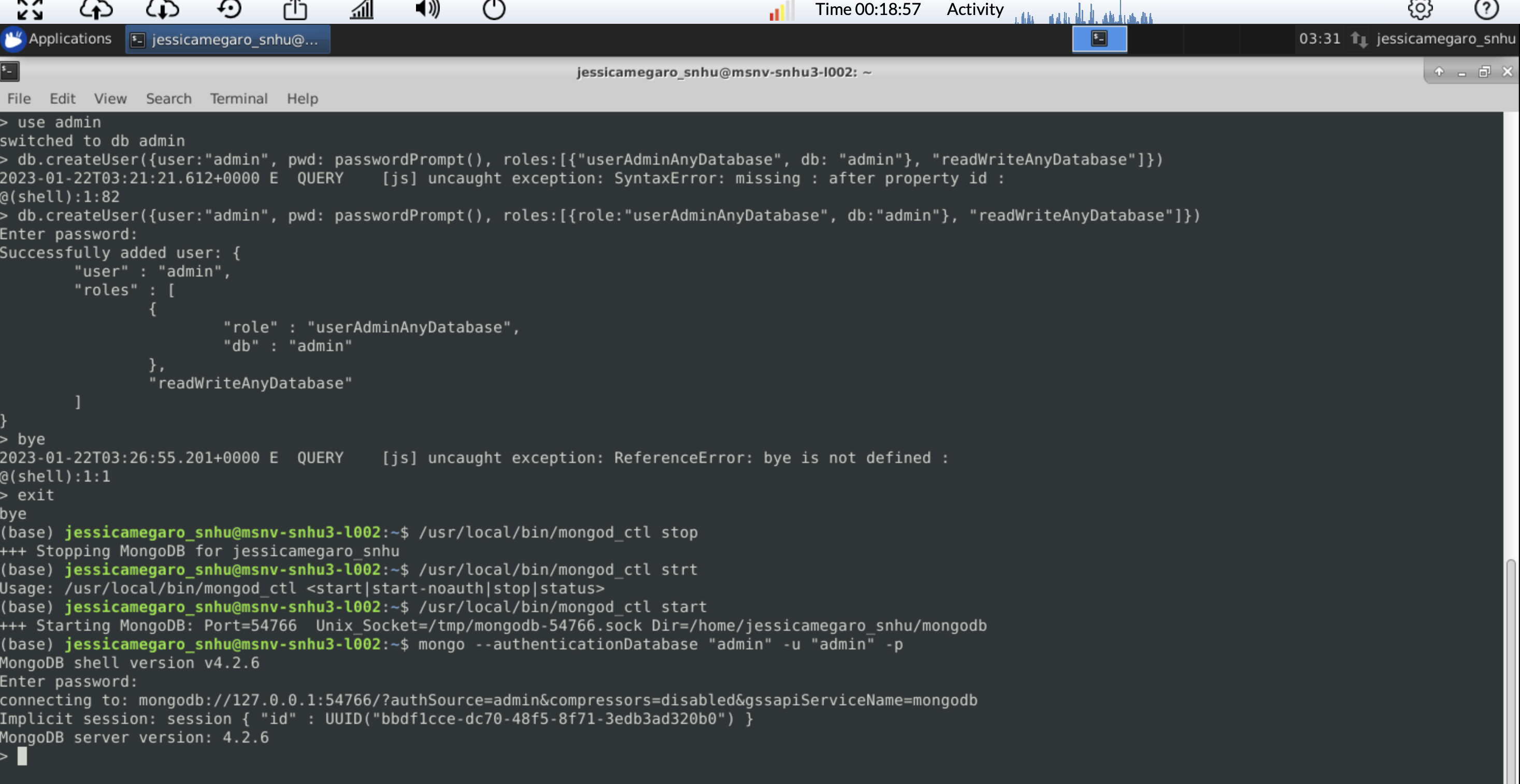
*Using Dash:*

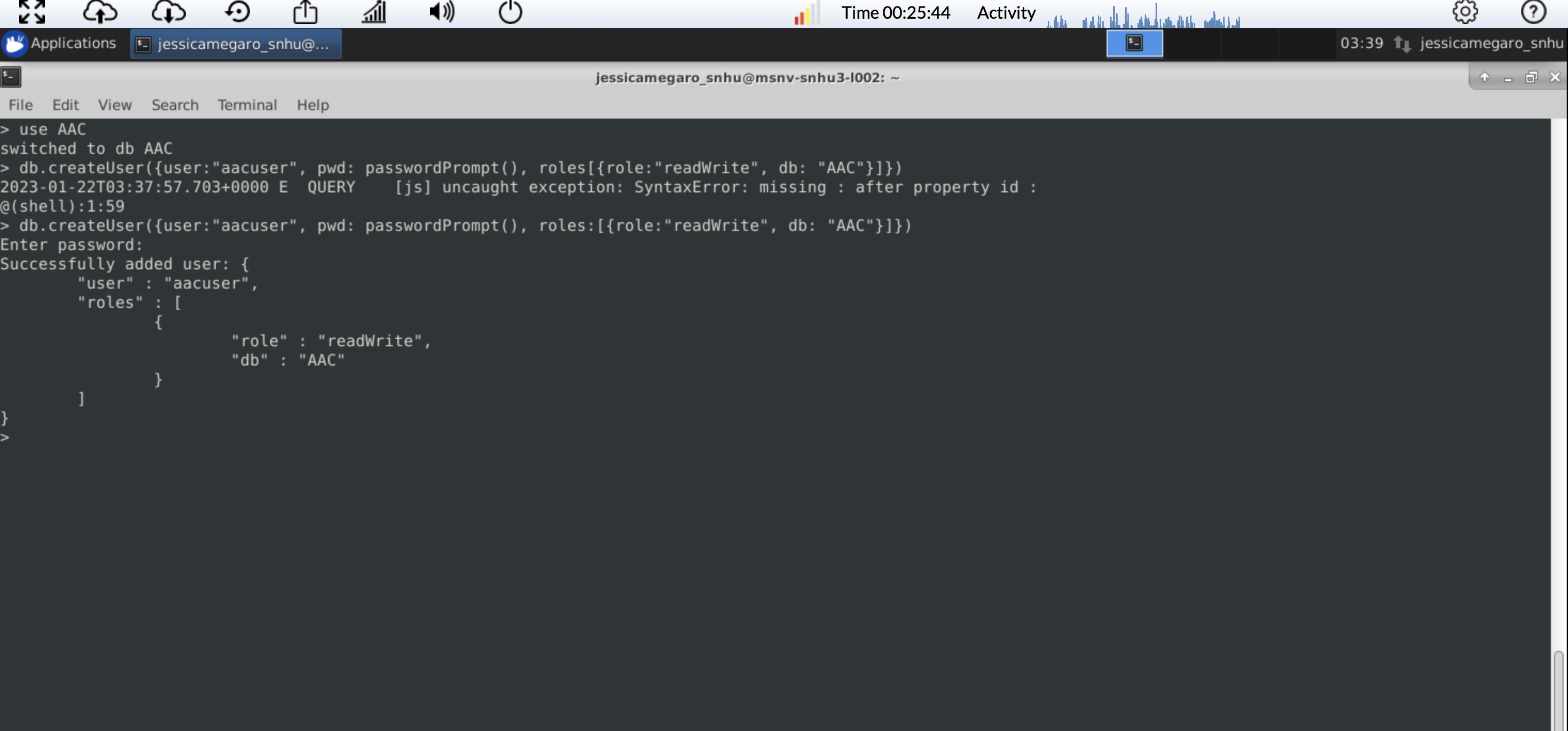
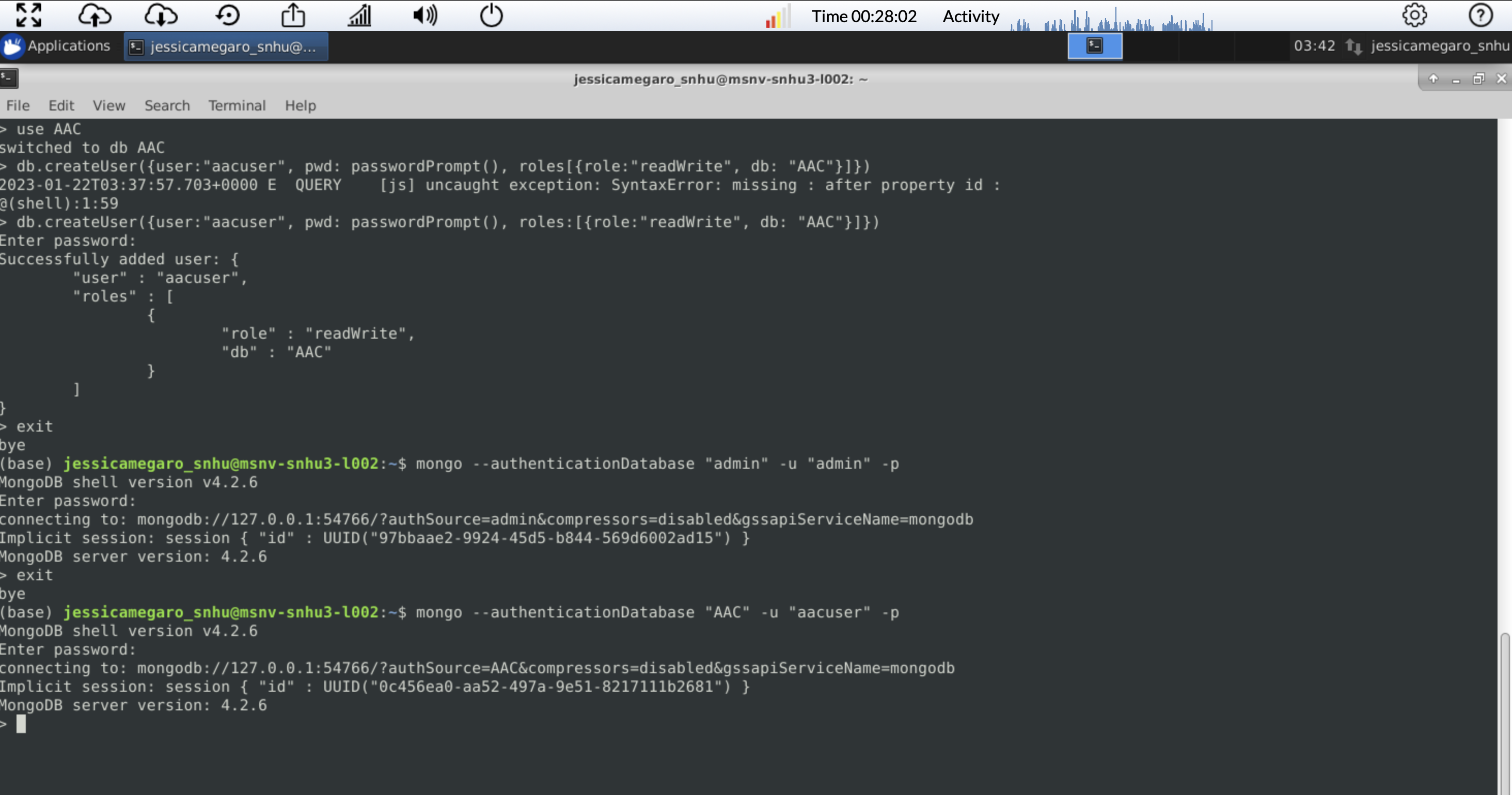
*Dash was the tool used to build the dashboard for several reasons. Dash is a JavaScript based tool that supplies an effortless way to build a responsive framework. Dash is well documented and has a highly responsive audience on GitHub and Stack overflow, as well as an open-source library under MIT’s permissive licensing. See full official documentation at: www.dash.plotly.com/introduction.*

## Getting Started:

1. ***Start Mongo***
2. ***Create a Mongo Database called AAC.***
3. ***Import CSV file into the AAC database: aac\_shelter\_outcomes.csv***
4. ***Create a user for accessing the database and set authentication database to AAC.***
   1. ***Username and password will be provided upon instantiation.***

***Screenshot of import and result:***

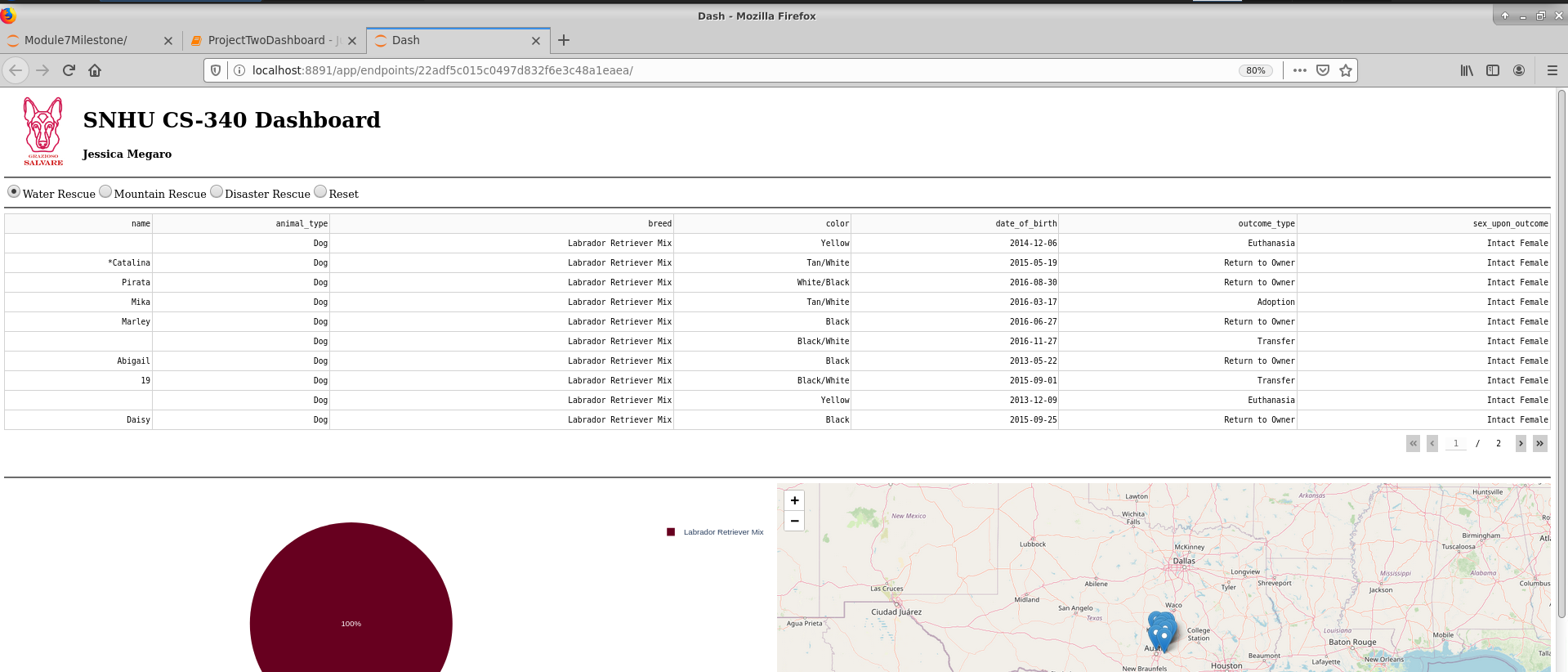




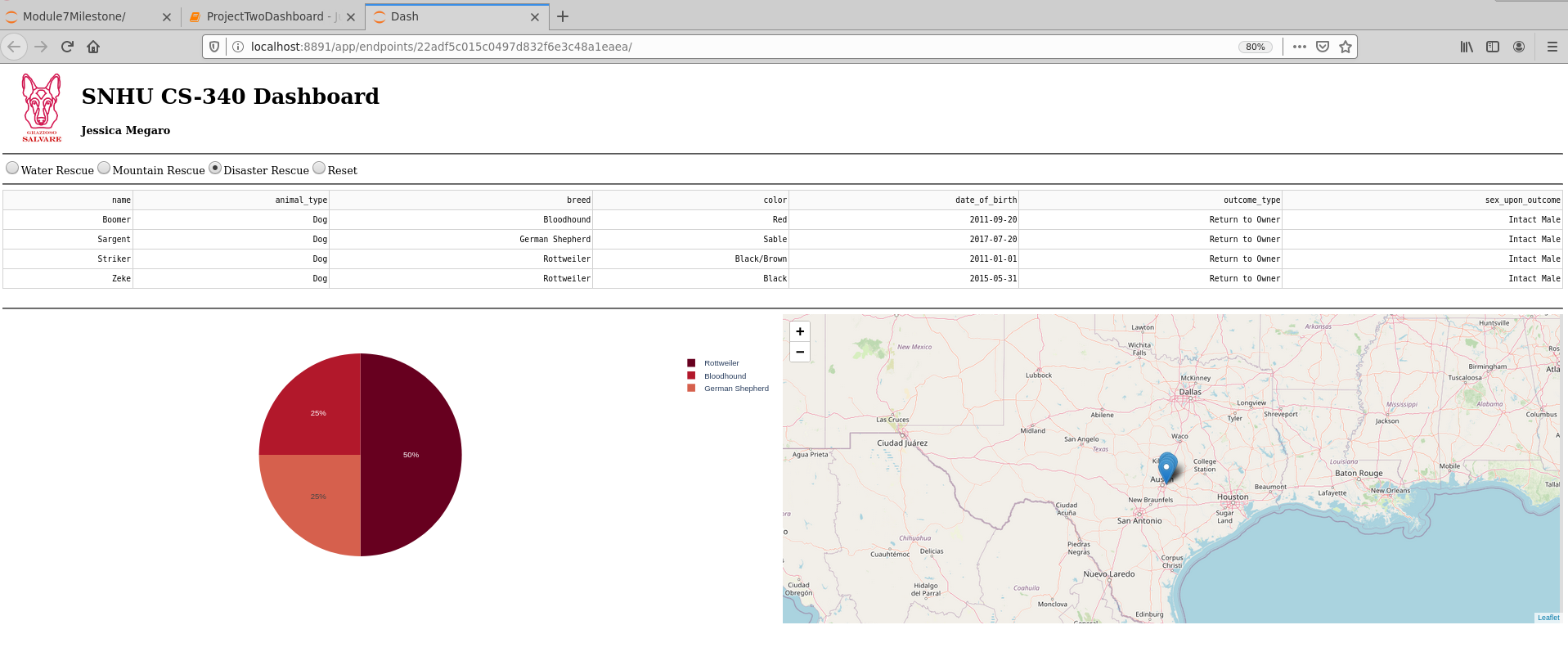
1. After the import, the database is ready to be used.
2. Create data table on dashboard that shows an unfiltered view of the Austin Animal Center outcomes data set. Populate using the previous CRUD Python Module.
3. Develop database queries to match the required functionality. In this case it will be sorted based on the type of rescue animal needed.
4. Create interactive options that allow for the selection based on the filtering functions. These will enable control of the dashboard widgets/geolocation map/pie chart.
5. Modify the dashboard to respond based on changes made from user input.

See examples of each screen shot after a widget has been chosen:

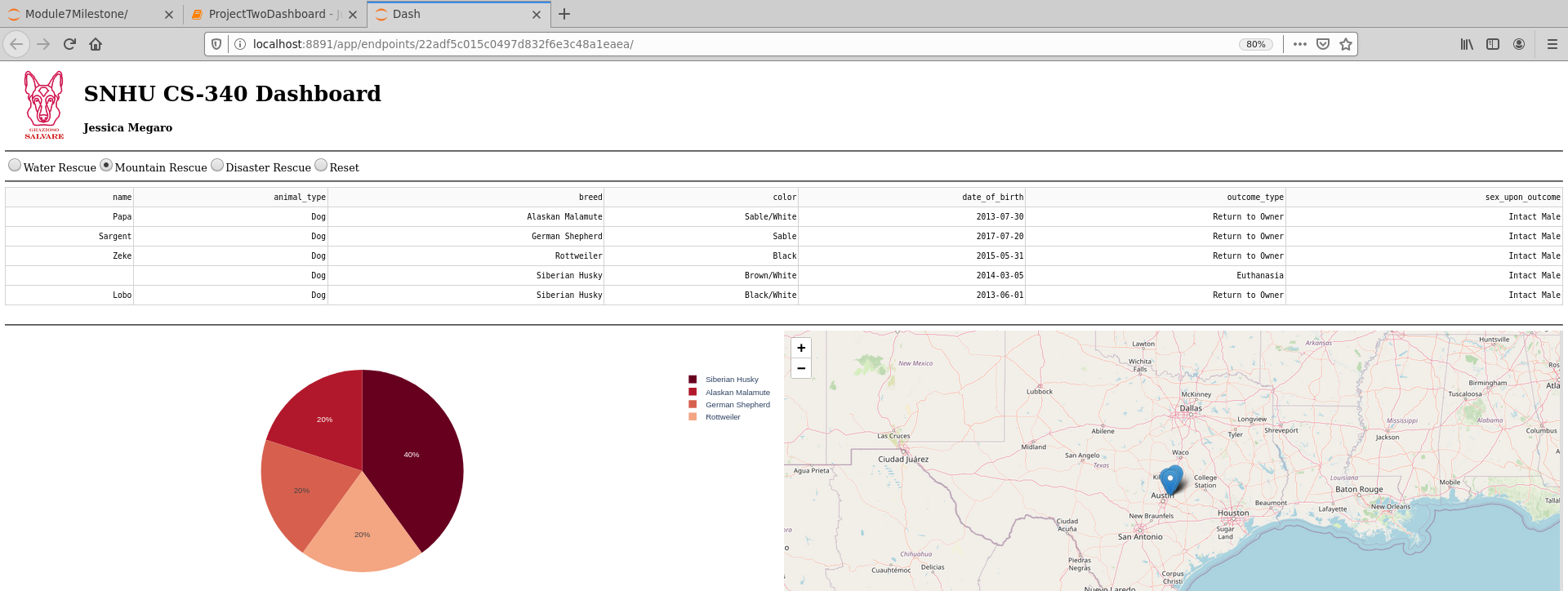
Water rescue widget:



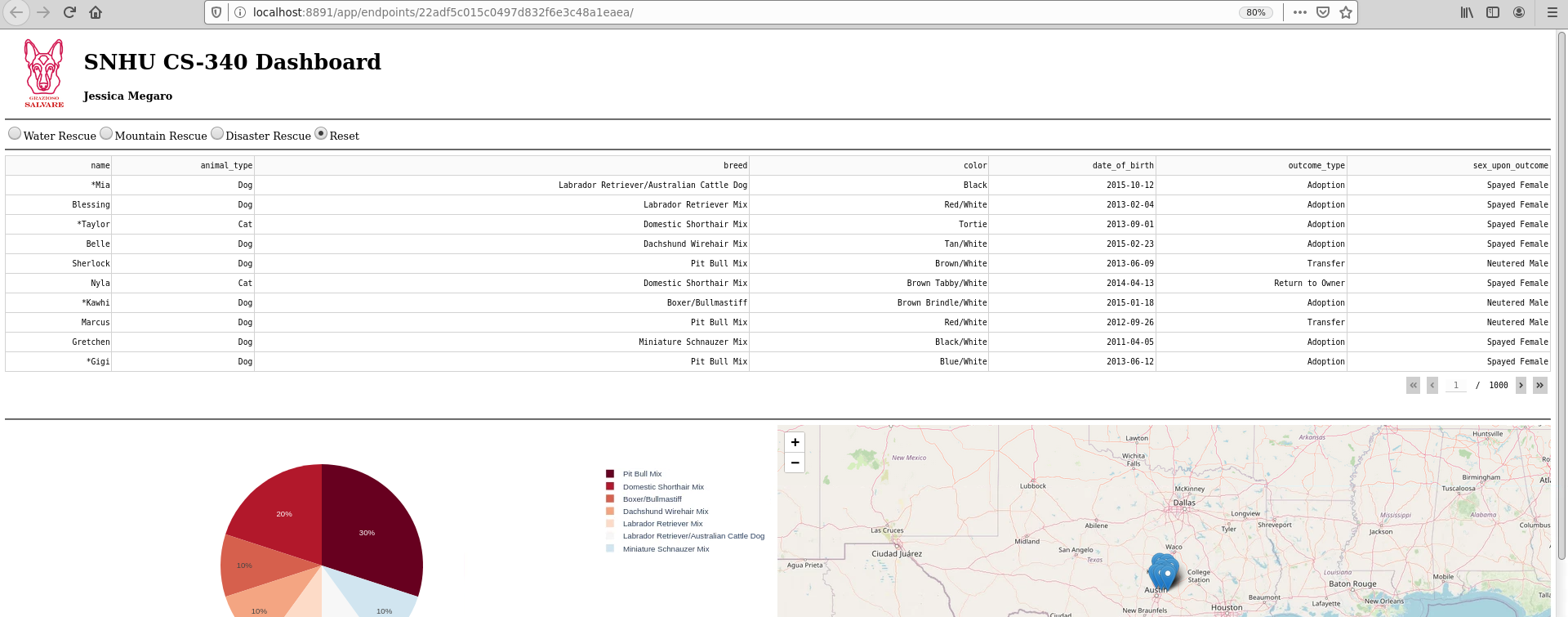
Disaster rescue:



Mountain rescue:



## Rest back to unfiltered:



## Installation

1. MongoDB
   1. *Detailed installation instructions can be found at* <https://www.mongodb.com/docs/manual/installation/>
2. Python
   1. Installation instructions can be found at <https://docs.python.org/3/using/index.html>
      1. Command prompt on windows for access.
      2. For Mac or Linux use terminal.
3. Jupyter Notebook
   1. <https://jupyter.org/install> for detailed installation instructions.
      1. Can be installed in any major OS.
      2. Use this text editor with a built-in python environment to insert documents into database and read it using the specified ID during creation.
4. Plotly
   1. To generate charts, Plotly must be imported directly into the Python module.
      1. <https://dash.plotly.com/> for support.
5. Dash
   1. Dash is an open-source library released under the permissive MIT license.
      1. <https://dash.plotly.com/installation> for support.

## Usage

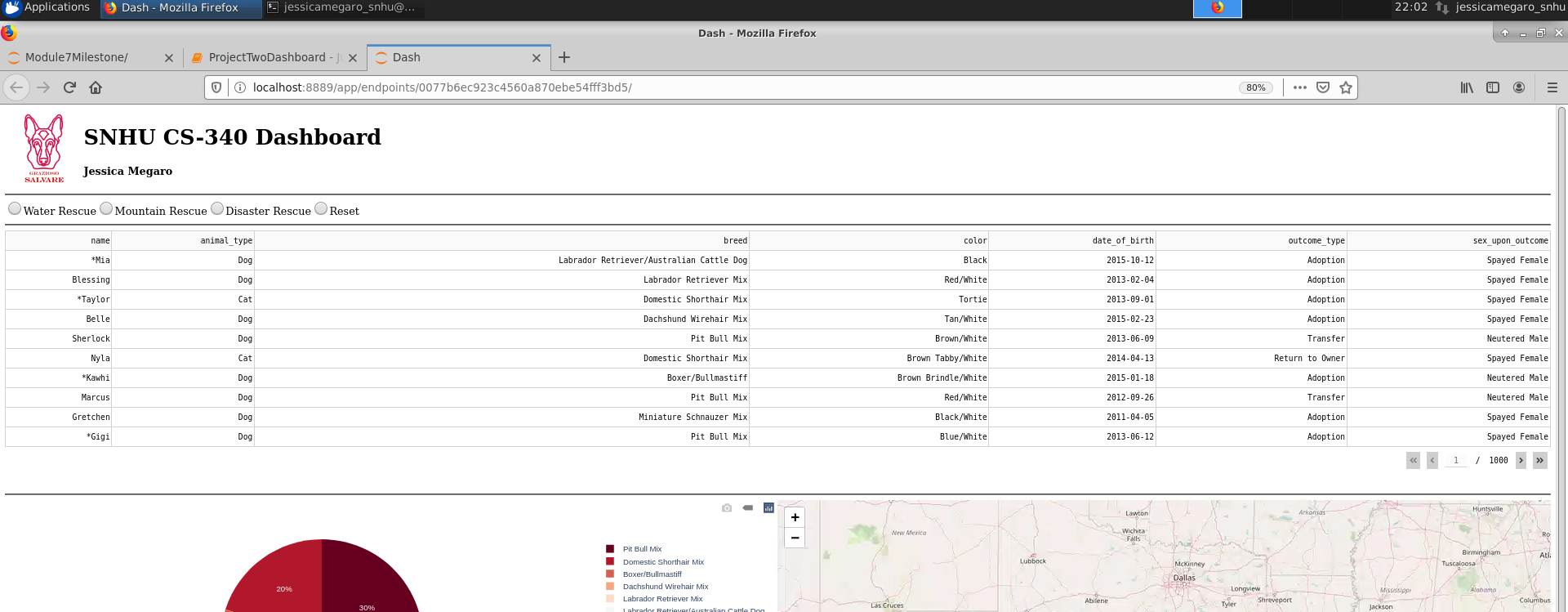
The application will use a CRUD Python module to perform four operations. The create operation will add new records to the database. The read operation will return data from the database as defined by the user. The update operation will modify existing data. The delete operation allows users to remove data from the database. Using a CRUD module in this way will allow the application to perform quickly and easily while also keeping the ability to scale if needed.

The client being used to access MongoDB is PyMongo. PyMongo is the library to communicate with the MongoDB server. This supplies an easy way to have flexibility and scalability during development and have access to all features of the NoSQL database.

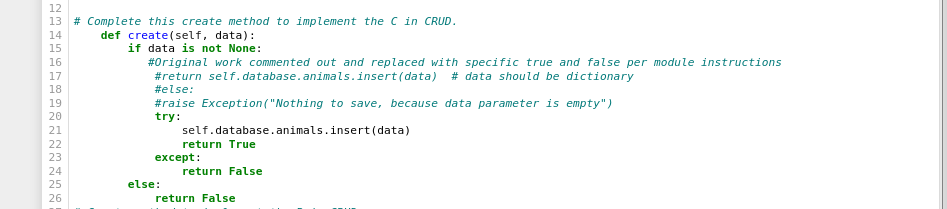
As the user accesses the dashboard, a query can be chosen, and the information will update accordingly. A user can choose among the options set by the specification sheet or go to reset at the end to have the information return to the unfiltered state. *After the code was created and tested, an interactive dashboard application can be started in a new window and used according to the user desires to see query results.*

*Please see screencast for widget functionality, geolocation chart as well as pie chart., submitted separately.*

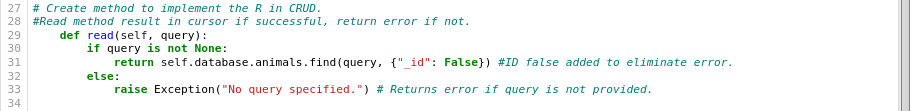
Dashboard example:



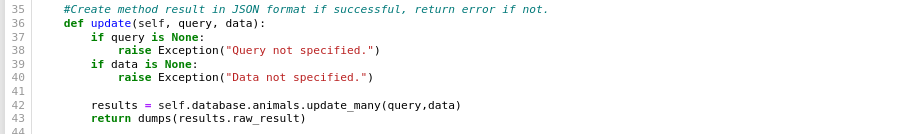
*See example of code included for create functionality:*



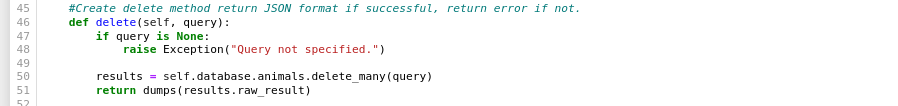
*Example of read functionality:*



Example of update functionality:



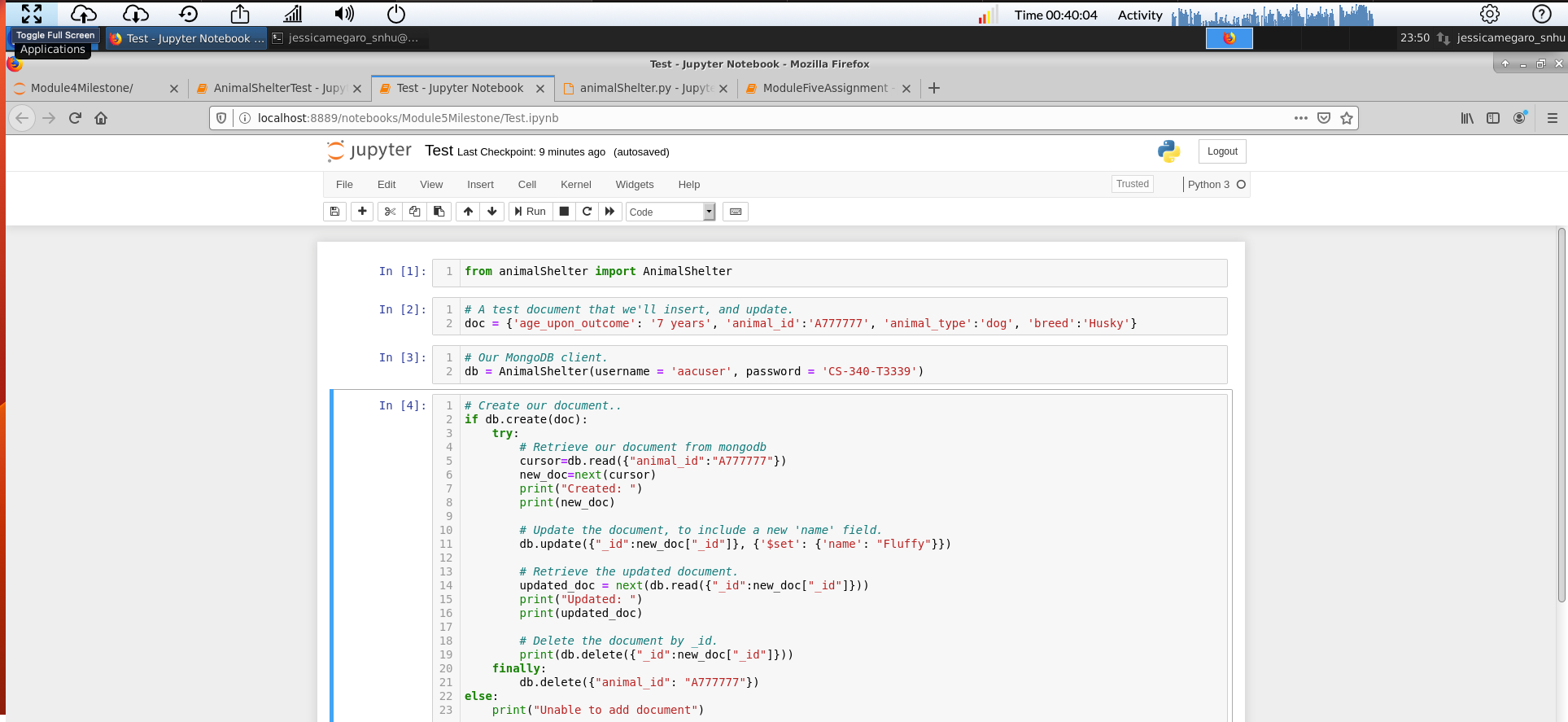
Example of delete functionality:

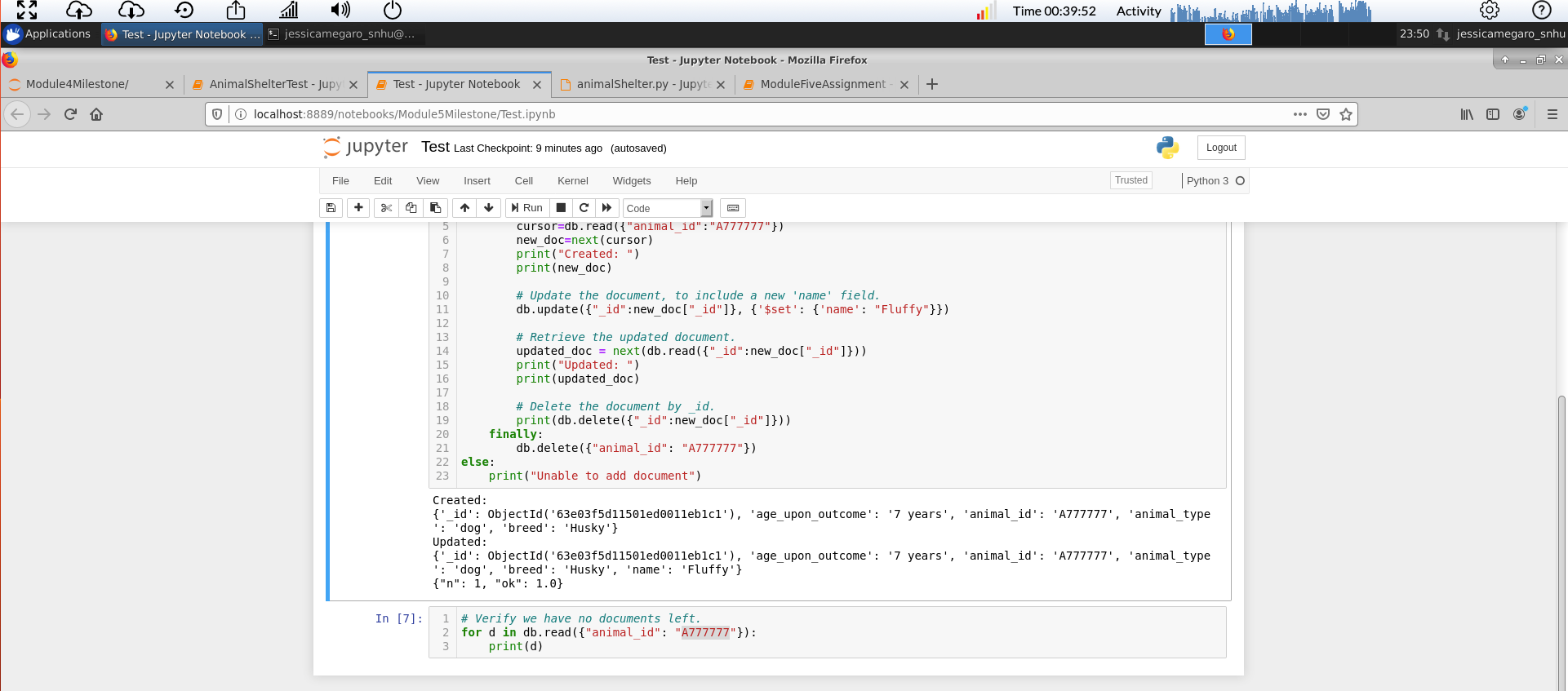


### Tests

*See example below for testing against document created using specified ID in Jupyter notebook:*

*All CRUD operations tested and verified, working without issue.*





*Challenges:*

*No challenges were incurred by this user.*

*For any challenges or issues that may arise, please reference proper manuals for the version of MongoDB being used such as: https://www.mongodb.com/docs/v4.2/*

## Contact

Your name: Jessica Megaro