# CS 340 README Conneway

## About the Project/Project Title

*Animal Shelter has a data set of all the animals in shelters. It lists the different aspects of each animal such as their name, breed, age, etc. The CRUD functionality takes the dataset and allows for creating, reading, updating, and deleting.*

## Motivation

*Grazioso Salvare sought a software application that could work with existing data from different animal shelters to identify and categorize available dogs. The application was motivated by the client to allow them the capability of finding certain dogs in these shelters they could train. They requested the application to search the dataset based on the specific query they preferred in dogs. Furthermore, they required the dashboard to allow filtering based on rescue types such as Water Rescue, Mountain or Wilderness Rescue, and Disaster Rescue or Individual Tracking. The filter ‘Reset’ will revert the dashboard back to its original state.*

## Getting Started

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

*Use the code:*

*From AnimalShelter import AnimalShelter*

*# then create an object from the class*

*# with your username, password*

*Example = AnimalShelter(username, password)*

*# from here you can use each method.*

## Installation

*I used MongoClient by importing it from pymongo. I also used PyMongoError by importing it from pymongo.errors. I used ObjectId by importing it from bson.objectid. These were useful to create the class and the PyMongoError allowed the try/catch error to be utilized. This was especially helpful in determing where issues were taking place.*

*You will run the dashboard using Jupyter Notebook or a Python IDE. After you run the program, you will click the link to take you to the dashboard. It will be running on a specific port.*

## Usage

### Code Example

*A screenshot of a computer program

Description automatically generated*

# Method to Implement the C in CRUD

def create(self, data):

""" Insert a document into the collection. """

if data is not None:

try:

self.collection.insert\_one(data)

print("Document imported successfully")

return True # Return True if insert was successful

except PyMongoError as e:

print(f"Error during insert: {e}")

return False

else:

raise Exception("Data parameter is empty")

A screenshot of a computer program

Description automatically generated

The data table is created with checklists to filter what data is shown. There are four options, and the id is used to connect the callback.

### Tests

*To run tests, create an instance of the AnimalShelter class. Create an animal to test and use the create method to add it to the shelter.*

*from AnimalShelter import AnimalShelter*

*# Instantiate the class*

*shelter = AnimalShelter(username, password)*

*# Insert a new animal record*

*test\_animal = {"name": "Lily", "species": "Dog"}*

*shelter.create(test\_animal)*

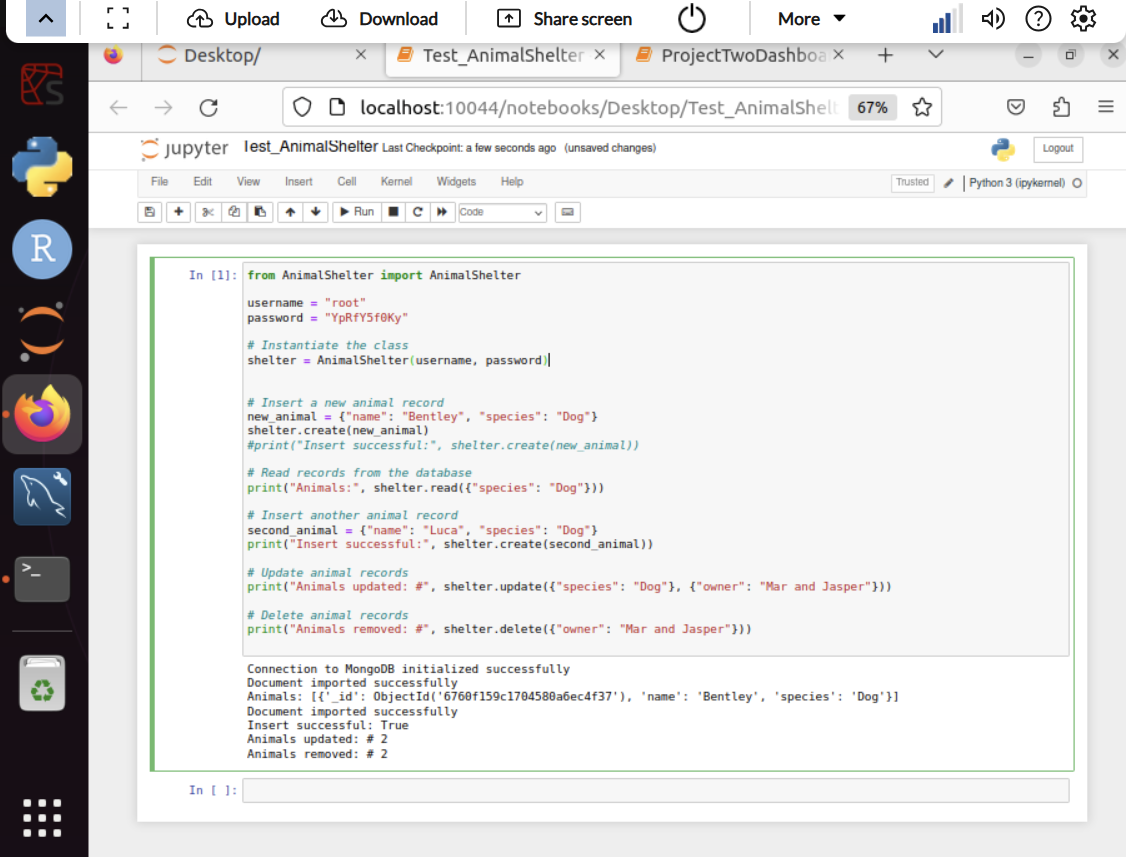
*#print("Insert successful:", shelter.create(test\_animal))*

*The print comment does not appear to work but you can use the read method to verify the animal was added.*

### Screenshots

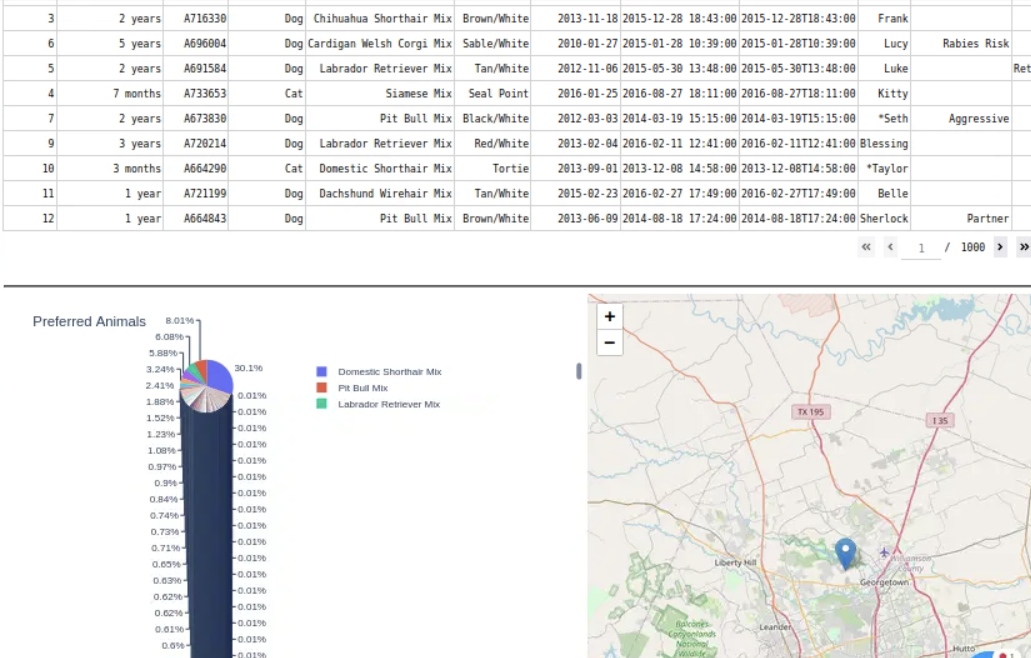
A screenshot of a computer program

Description automatically generated

**



*Your dashboard will look like this depending on how you edit the code.*

**

*The dashboard allows you to shift pages of the table and look for each animal through a geolocation chart. The pie chart shows the data table in pie chart form and will be updated based on the filters used. You can scroll what each breed is shown or hover over the chart.*

## Roadmap/Features

*I could not determine a fix for the errors I was receiving for my filter checklists.*

## Contact

Your name: Jasper Conneway