# Anatomy of a README

## About the Project/Project Title

**Grazioso Salvare Animal Training and Rescue Project**

This project is a fullstack application aimed at identifiying and training dogs suitable for search-and-rescue training. It provides an easy interface to view rescue animals from many local animal shelters. The project includes a MongoDB database and a client-facing web application for users to easily work with animal shelter data. PyMongo was chosen as it is the official supported MongoDB driver for Python. PyMongo offers a way to directly interface with the database programmatically using Python code. This allows us to write reusable code that can be used to access the database in predefined ways, eventually allowing us to offer a client-side user interface giving an easier and more user-friendly way to work with the database.

## Motivation

Search-and-rescue dogs are underappreciated heroes of emergency response teams. Finding dogs suitable for this highly sensitive and skilled field is a time-consuming task. Aggregating data from animal shelters gives knowledgeable people much-needed data all in one place to not only rescue dogs from the shelter, but train these dogs to rescue others as well.

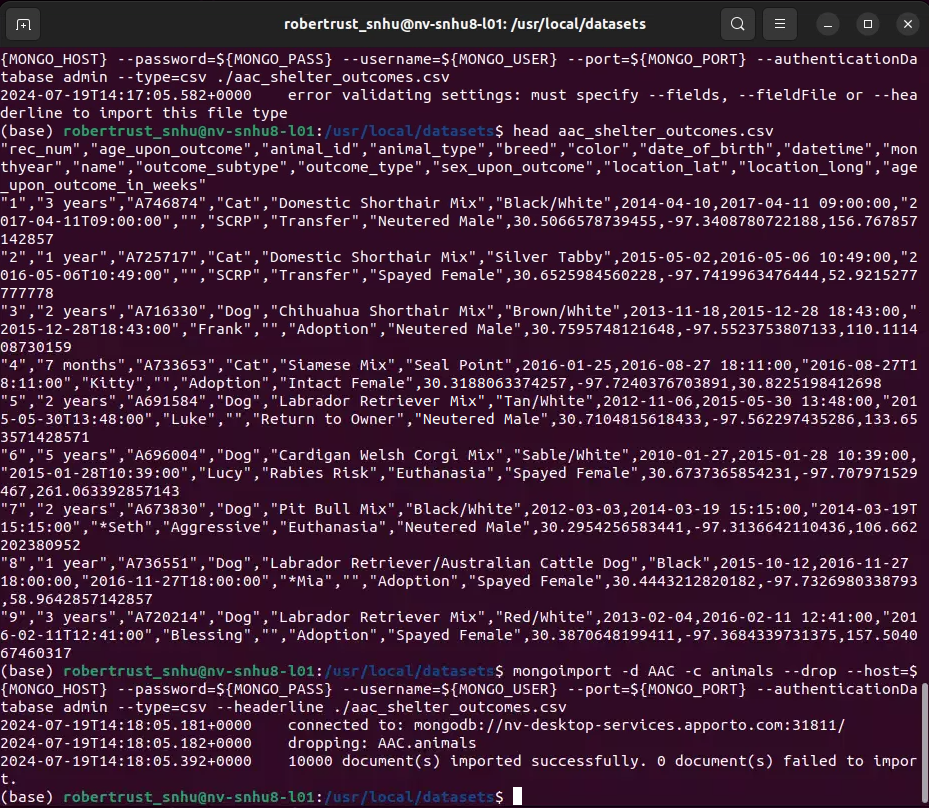
## Getting Started

To run this project locally, clone the GitHub repository: `git clone <https://github.com/example/aac-outcomes.git`>. Next, follow the installation instructions.

## Installation

*List the tools you need to use the software and how to install them.*

1. Ensure you have [MongoDB](https://www.mongodb.com/docs/manual/installation/) and [Python3](https://www.python.org/downloads/) (version 3.7+) installed.
2. Import the “aac\_shelter\_outcomes.csv” file using the mongoimport tool. Your output should look similar to the following:



1. Follow [these instructions](https://www.mongodb.com/docs/languages/python/pymongo-driver/current/get-started/download-and-install/) to get started with PyMongo. Note: You only need to follow the instructions on the page linked, not the rest of the tutorial.
2. If you don’t have a MongoDB user set up, run:

`mongosh`

`use admin`

`db.createUser({

user: <your\_username>,

pwd: <your\_password>,

roles: [{

role: “readWrite”,

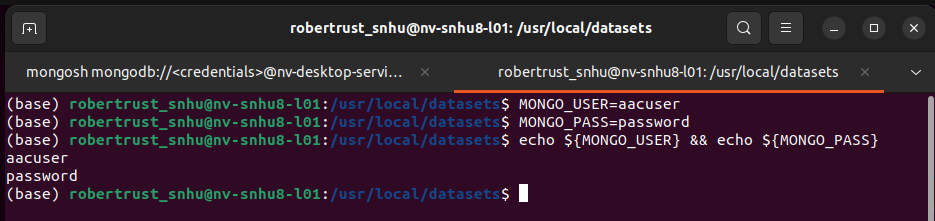
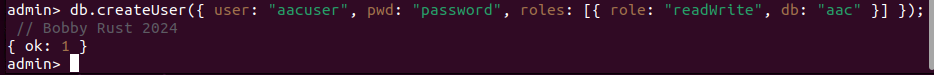
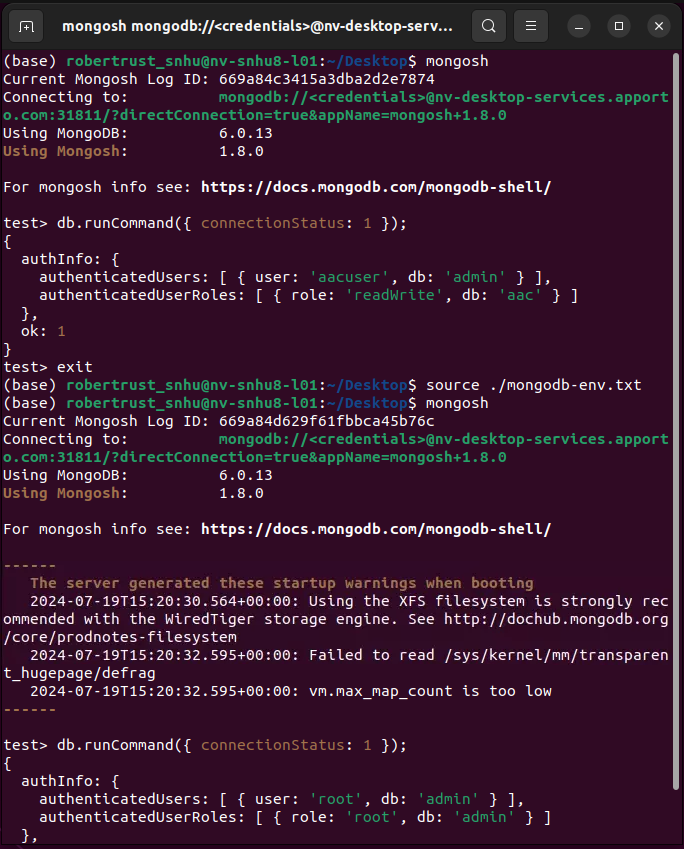
db: “AAC”

}]

})

`

Alternatively, you can use an administrator account.

1. In the python file, set `USER` to <your\_username>, `PASS` to <your\_password>, `HOST` to localhost (or your MongoDB connection URI), and `PORT` to 27017 (or whatever port you’re running MongoDB on).
2. You can now run “create\_and\_read\_test.ipynb” Jupyter Notebook file to view examples of how to use the API and to ensure you have everything up and running correctly.
3. If you have any issues, create a discussion post on the GitHub repository.

## Usage

This module contains CRUD functionality for interacting with the animals database. It has create, read, update, and delete methods. The create method takes a python dictionary and inserts the given document into the database. The read method takes a dictionary containing fields that will be used to return results that match the fields. The update method takes an operation field containing either the value “one” or “many” corresponding to the respective database operation, and the field(s) to update under the ‘match’ field, and a ‘new\_values’ field containing the new values. Similarly, the delete method takes an operation (‘op’), and a match field. If ‘one’ is passed in the ‘op’ field, the first document found matching the ‘match’ criteria will be deleted. If ‘many’ is passed, every document found matching the ‘match’ criteria will be deleted.

### Code Example

`from animal\_shelter import AnimalShelter

From pprint import pprint # pretty print dictionaries

# create a new instance of the AnimalShelter class

shelter = AnimalShelter()

# create a new animal dictionary

new\_animal = {

"age\_upon\_outcome": "1 year",

"animal\_type": "Cat",

"breed": "Domestic Shorthair Mix",

"color": "Black",

"date\_of\_birth": "2023-04-06",

"datetime": datetime.now(),

"monthyear": datetime.now(),

"name": "Felix",

"outcome\_subtype": "SCRP",

"outcome\_type": "Transfer",

"sex\_upon\_outcome": "Neutered Male",

"location\_lat": 100.0,

"location\_long": 100.0,

"age\_upon\_outcome\_in\_weeks": 64

}

# insert the new animal into the database

shelter.create(new\_animal)

# query the database for animals

search = {

"age\_upon\_outcome": "1 year",

"animal\_type": "Cat",

"breed": "Domestic Shorthair Mix",

"color": "Black",

"date\_of\_birth": "2023-04-06",

"name": "Felix",

"location\_lat": 100.0

}

# returns a list of all objects matching the query

results = shelter.read(search)

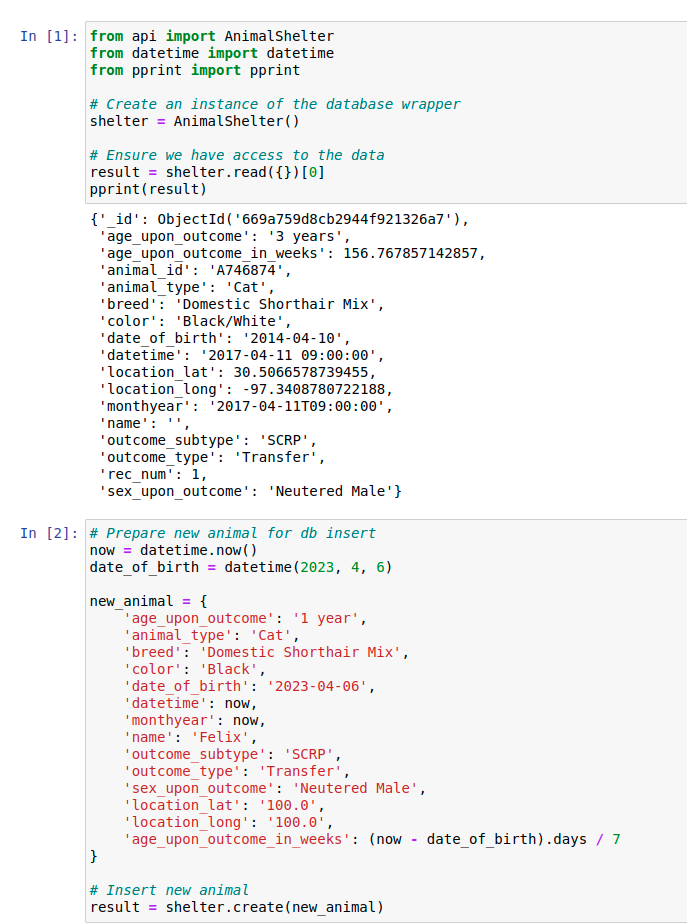
pprint(results)

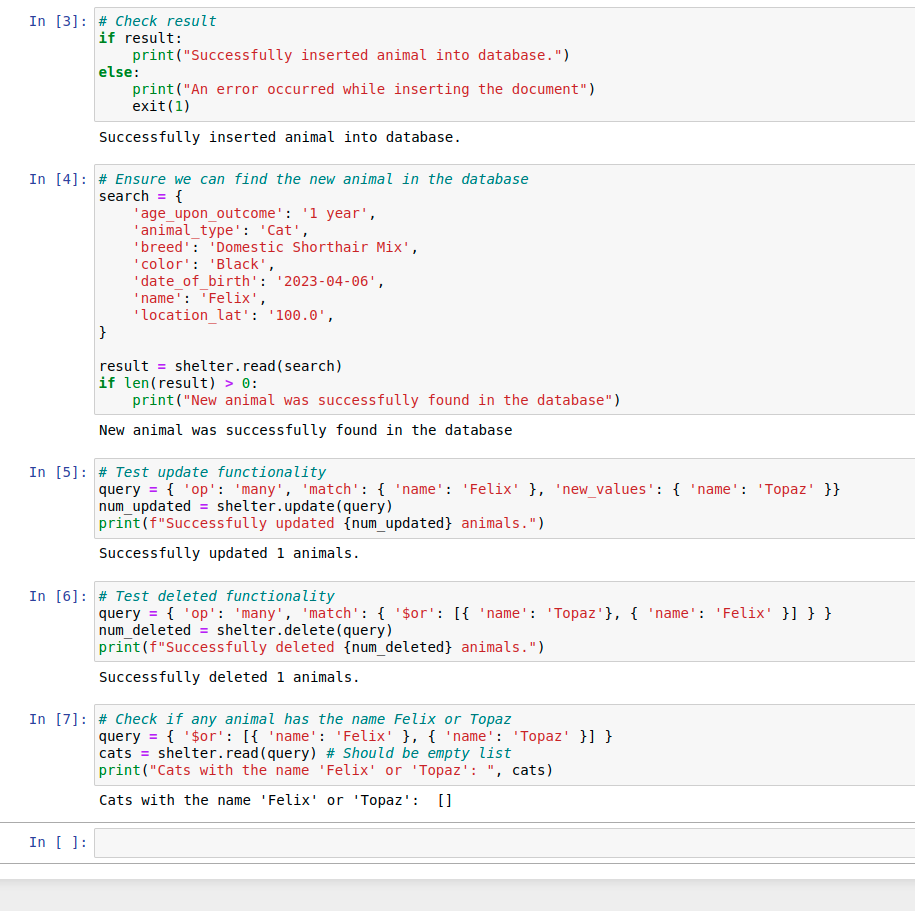
### Tests

*Describe and show how to run the tests with code examples.*

1. Open create\_and\_read\_test.ipynb in Jupyter Notebook.

2. Run the cells.

Screenshots  




**Frontend Dashboard**

**Required Functionality**

This web app uses information from an animal shelter database to search for animals that are fit to be trained as rescue dogs. Each button corresponds to a category of rescue animals. Different breeds of dogs have different strengths that make them more suited for certain types of rescues. By clicking a button, you can filter through all of the animals in the shelter to find the available animals that meet the criteria. The map shows where the currently selected animal is located. The pie chart shows the breeds of animals available in the shelter by their percentage. Each breed that makes up less than 1 percent of the total animals is grouped under the ‘Other’ category. The pie chart updates when the data is filtered for rescue animals to show the percentage of all shelter animals that are fit for the rescue type selected. **Tools Used**

The tools used to build this dashboard are the Dash framework and MongoDB. The Dash framework is a python framework for easily and quickly building data dashboard front ends. It provides many useful tools for easily showing data in different formats, such as bar charts and pie charts. It allows the developer to write front-end code in Python which lends itself to quick development. MongoDB is a non-relational database management system. These types of databases are good for large amounts of unstructured data and are easy to use as they are less fussy than relational databases such as SQL. MongoDB provides flexibility in data. The official MongoDB driver, PyMongo, is used to allow Python to interact with the MongoDB database, which is hosted locally.

**Dash:** https://dash.plotly.com/

**MongoDB:** <https://www.mongodb.com/>

**PyMongo:** https://pymongo.readthedocs.io/en/stable/

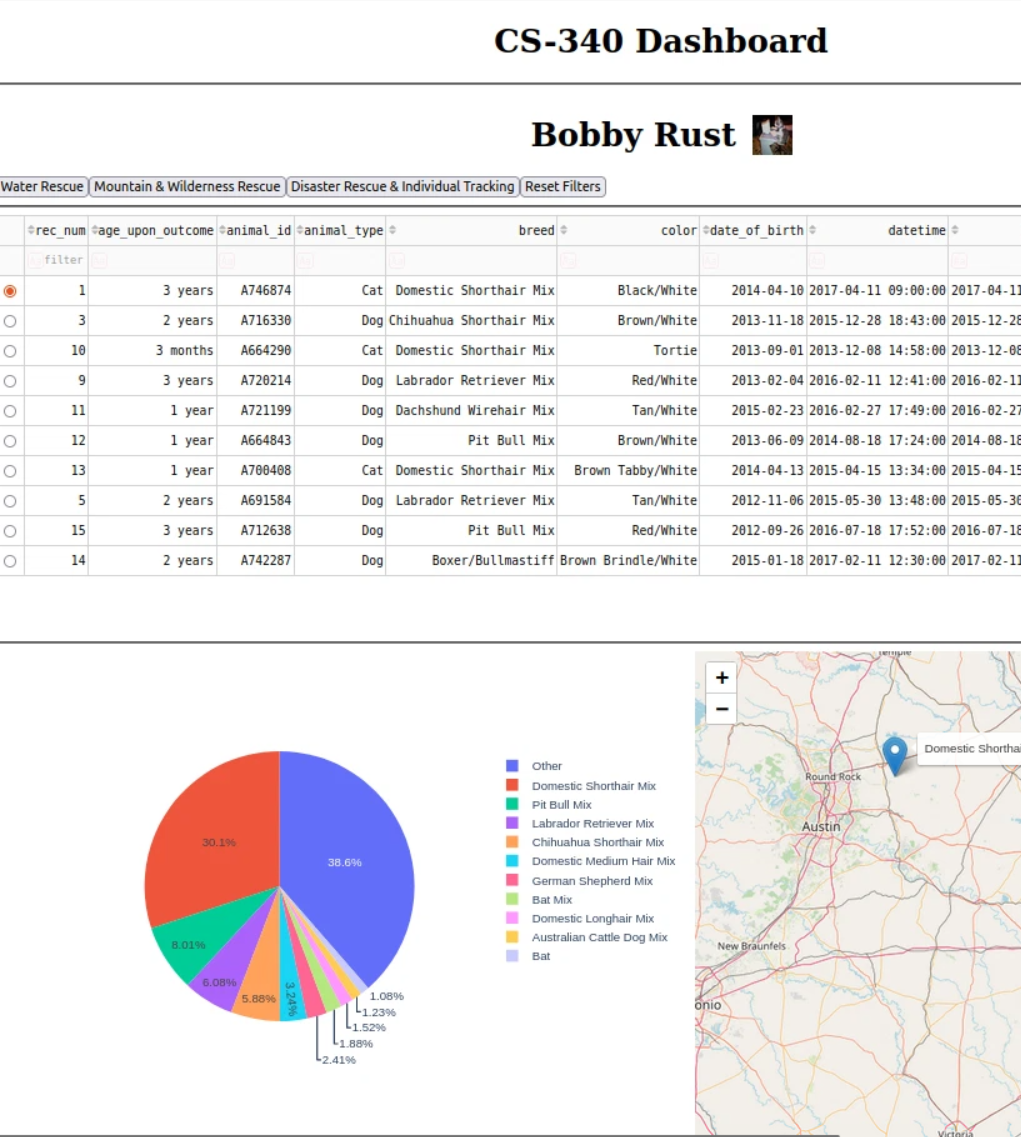
**Development Process**

The Dash framework uses React under the hood to transpile Python code into React, which then transpiles to JavaScript. A model-view controller (MVC) architecture was used to connect the database to the frontend. Dash provides an app.Layout function in which the HTML for the page is created. The classnames for the HTML elements can be connected to callback functions which is a way for Dash to allow for state updates for an interactive webpage. A separate API was created which provides an interface to the database to perform CRUD operations. That API file is imported into the Dash application code and is used to retrieve the data for Dash to display.

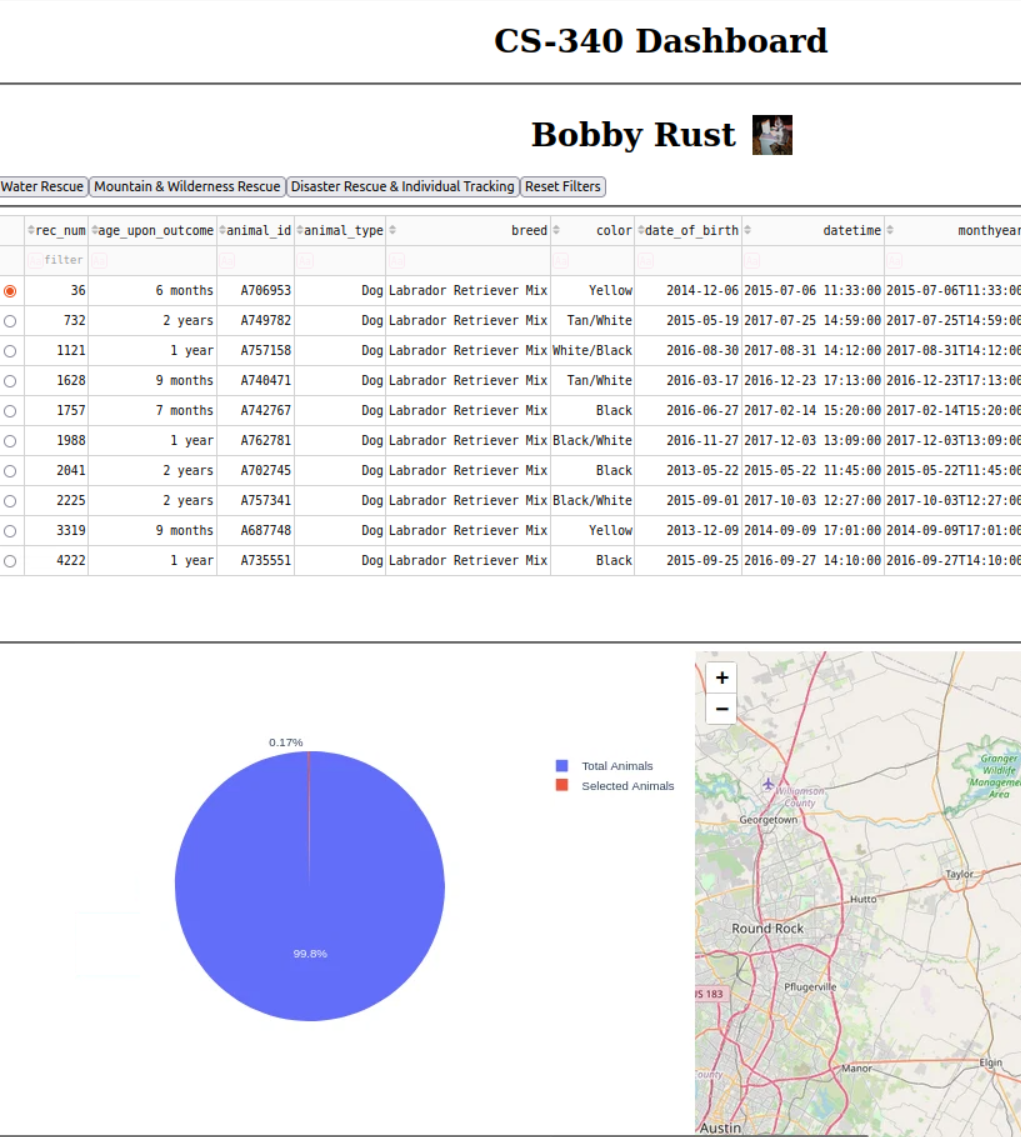
**Overcoming Development Challenges**

There were a few challenges faced when developing this application. Most of these were due to this being my first time using the Dash framework, so I had to get accustomed to it. I did not understand how to connect the callback functions to the HTML elements which provide state updates due to the layer of abstraction between the developer and the React code. The Dash documentation proved useful for finding examples of Dash apps and documentation for the different functions Dash offers. Along with this, googling error messages is useful as other developers had run into the same or very similar issues to what I experienced and provided solutions.

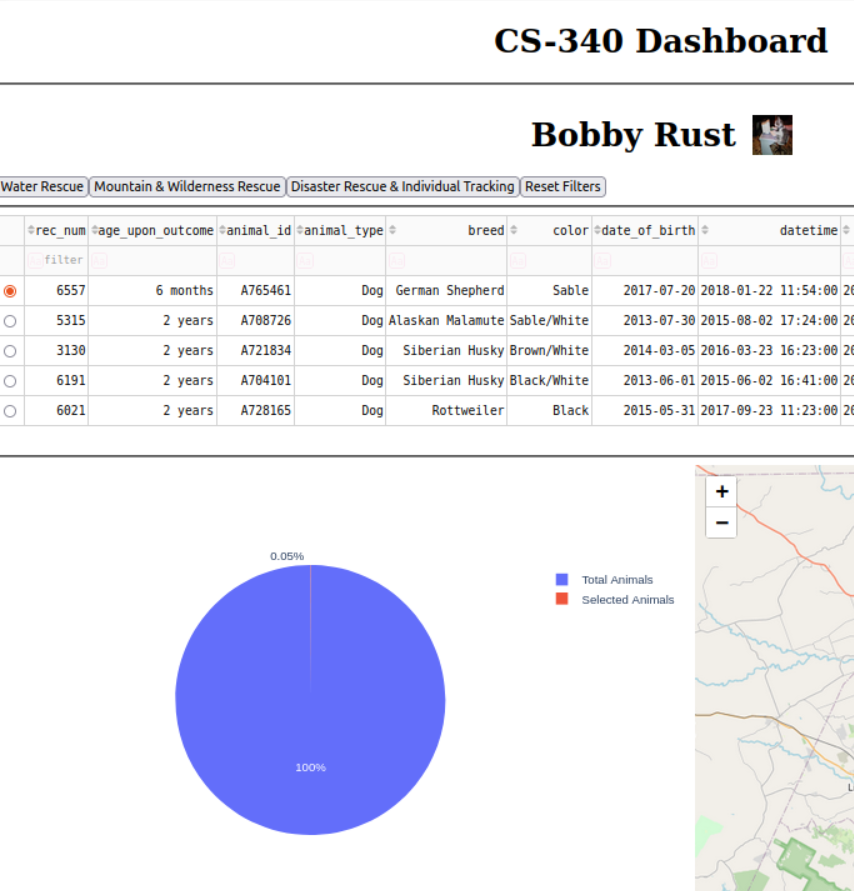
**No Filters (Reset)**



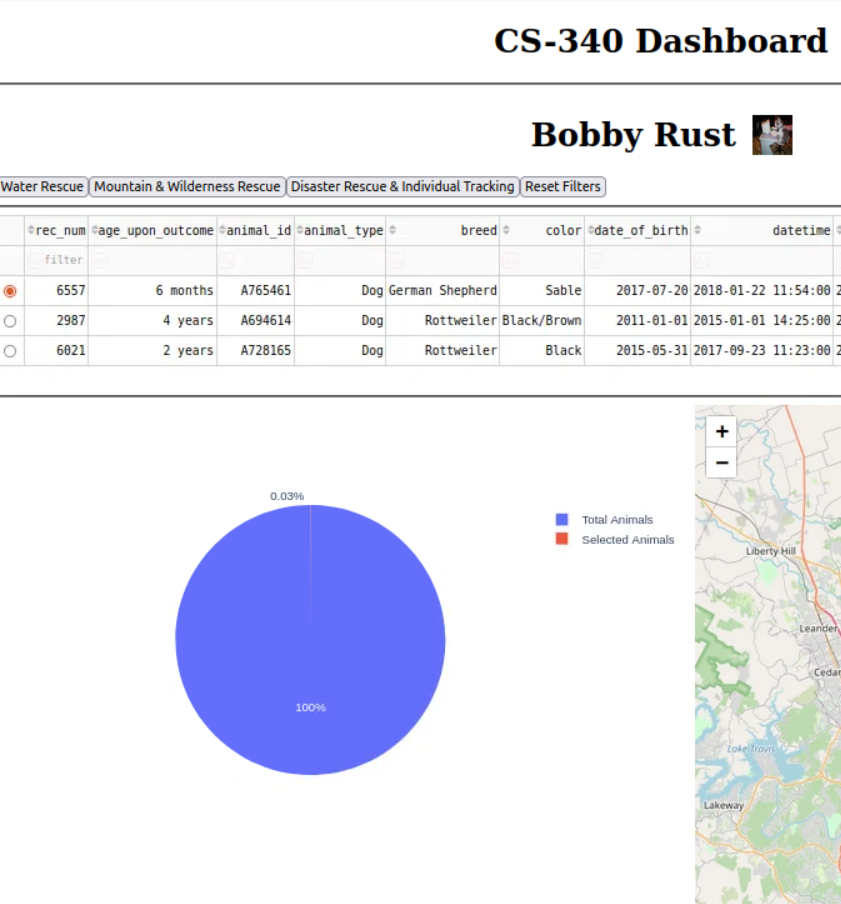
**Water Rescue**



**Mountain & Wilderness**



**Disaster Rescue & Individual Tracking**



## Contact

Name: Bobby Rust

GitHub: https://github.com/bobby-rust