# CS 340 README Template

## About the Project/Project Title

This project is a database for an animal shelter using MongoDB. In this database we store information about the animal like name, type, breed, etc. The python component allows the user to read and write to the database as well as update and delete existing data.

## Motivation

This project helps users run and understand MongoDB and python scripts. MongoDB is accessed and controlled through the command line interface, this is important as most databases and servers have no GUI. Understanding python and how it interacts helps the user create scripts that assist in reading, creating, updating, and deleting entries in the database.

## Getting Started

To get started you must have MongoDB, and Python installed.

1. Load the “aac\_shelter\_outcomes.csv” file that is found in the /usr/local/datasets/ directory.

2. Create an administrator account in MongoDB. See [MongoDB Manual Enable Access Control](https://www.mongodb.com/docs/v4.2/tutorial/enable-authentication/)  for instructions.

3. Log into the admin account and create a new user named “aacuser” in the AAC database.

4. In Python configure the correct credentials and port for your database.

5. Update the testing file with data you want to enter.

6. Run the testing file.

7. Edit the project two file to include your log in information

MongoDB was used to store the data. Using MongoDB allows higher performance due to the data models compared to SQL databases. Accessing and changing data is more natural in MongoDB due to its BSON data format. The create, read, update, and delete functions use MongoDB syntax through pymongo.

Pymongo contains all the tools needed for interacting with MongoDB through Python. For create, the insert function was used to insert new documents into the collection. For read, the find function was used to query for the document in the collection based on the criteria given. The update function searches for the document that matches the search criteria and updates one of the values based on the key-value pair given. The delete function searches for the document based on the criteria given and if found, removes it from the collection.

The Dash framework is a data visualization and UI library. The Dash frame work was used to create the table to display the data to the user in an easily readable way. It is also used to create a chart to display the breakdown of all the different breeds of animals. Another framework that was used was Dash Leaflet. Dash Leaflet is a wrapper for a JavaScript library that allows for interactive maps.

One of the challenges I encountered was accessing the database, but that was due to a technical issue with the virtual environment I was using. Another challenge I encountered was that the dash table would be stuck on just “loading”. I did a few things to narrow down the problem and found it had something to do with the line *data=df.to\_dict('records')* which turns the panda dataframe that has all the documents into a dictionary. I ran Jupyter Notebook through the terminal, so I could get some output and got some error message when trying to check the output of the data variable, one which was particularly interesting: 'TypeError: Object of type ObjectId is not JSON serializable'. ObjectId seems to be unique data type that can’t be use in a JSON format. To fix this, I added  {"\_id": False} to the read method. {"\_id": False} omits the unique ID for each row.

## Installation

MongoDB - [Install MongoDB](https://www.mongodb.com/docs/manual/installation/)

Python - [Installing Python](https://realpython.com/installing-python/)

Jupyter Notebook - [Install Jupyter Notebook](https://jupyter.org/install)

Plotly Dash - [Plotly Dash](https://dash.plotly.com/installation)

Dash Leaflet - [Dash Leaflet](https://dash-leaflet.herokuapp.com/)

Pandas - [Pandas](https://pandas.pydata.org/getting_started.html)

## Usage

### Code Example

Importing data into MongoDB:Mongoimport –port “your port number” –db AAC –collection animals – type csv –headerline –file ./aac\_shelter\_outcomes.csv

Getting database access through pymongo:

self.client = MongoClient('mongodb://%s:%s@localhost:56223/AAC' % ('username', 'password'))

The create method:

def create(self, data):

if data is not None:

insert = self.database.animals.insert(data)

if insert!=0:

return True

else:

return False

else:

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

Putting the data in a table:

dt.DataTable(

id='datatable-id',

columns=[

{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns

],

data=df.to\_dict('records'),

editable = False,

filter\_action = "native",

sort\_action = "native",

sort\_mode = "multi",

column\_selectable = False,

row\_selectable = 'single',

row\_deletable = False,

selected\_columns = [],

selected\_rows = [],

page\_action = "native",

page\_current = 0,

page\_size = 10,

### Tests

For the testing, create a Json object to input into the database. To make sure you put in the data correctly you can read the data back using a key and value pair from the sample data and putting it into the read section.

animal\_data = [

{

'age\_upon\_outcome': '2 years',

'animal\_id': 'B214543',

'animal\_type': 'Bat',

'breed': 'Hoary Bat',

'color': 'Brown',

'date\_of\_birth': '2022-01-22',

'datetime': '2022-02-02 10:00:00',

'monthyear': '2022-02-02T10:00:00',

'name': 'Bruce',

'outcome\_subtype': '',

'outcome\_type': 'Adoption',

'sex\_upon\_outcome': 'Male',

'location\_lat': 30.5066578739455,

'location\_long': -97.3408780722188,

'age\_upon\_outcome\_in\_weeks': 156.767857142857

}

]

for i in animal\_data:

animal.create(i)

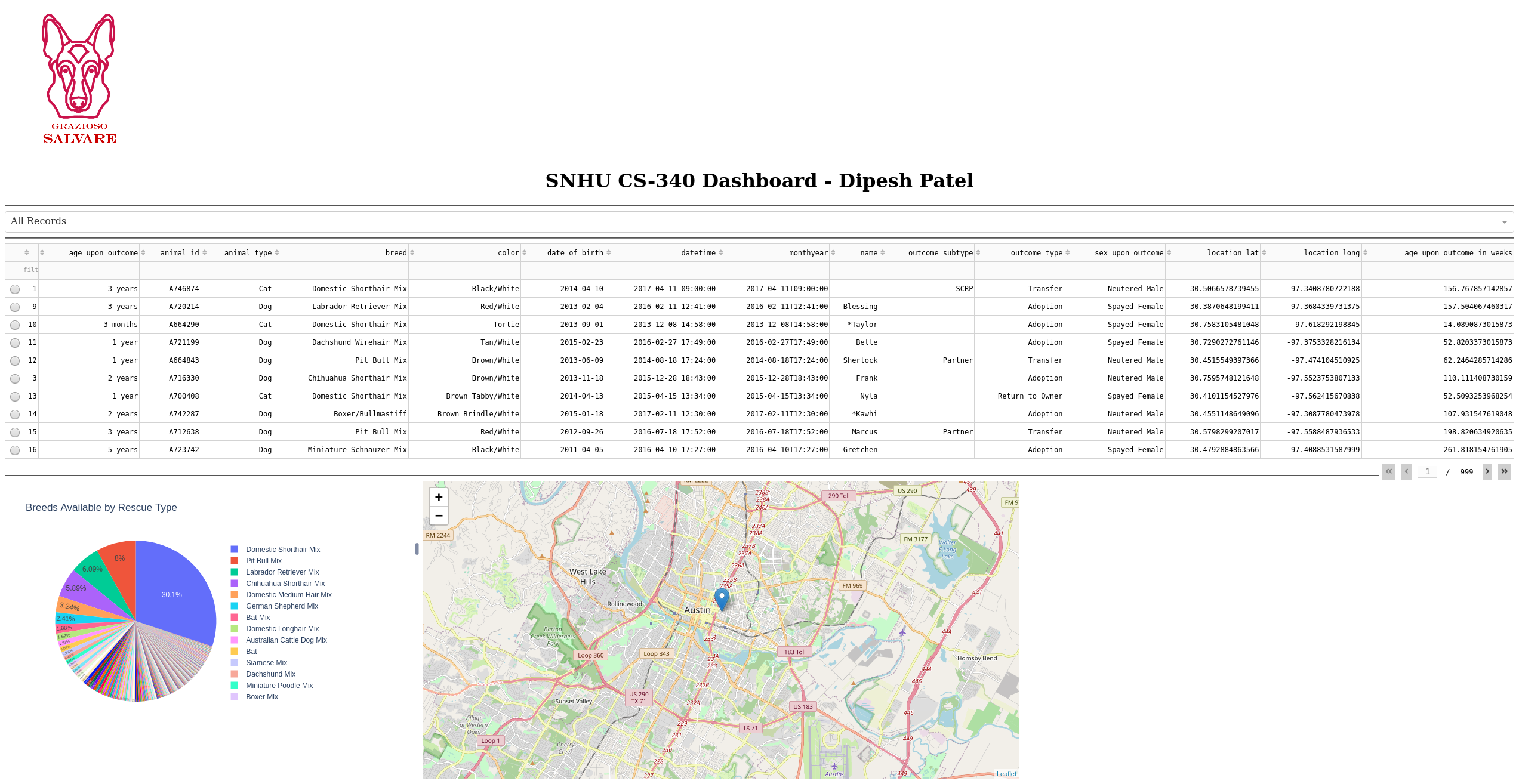
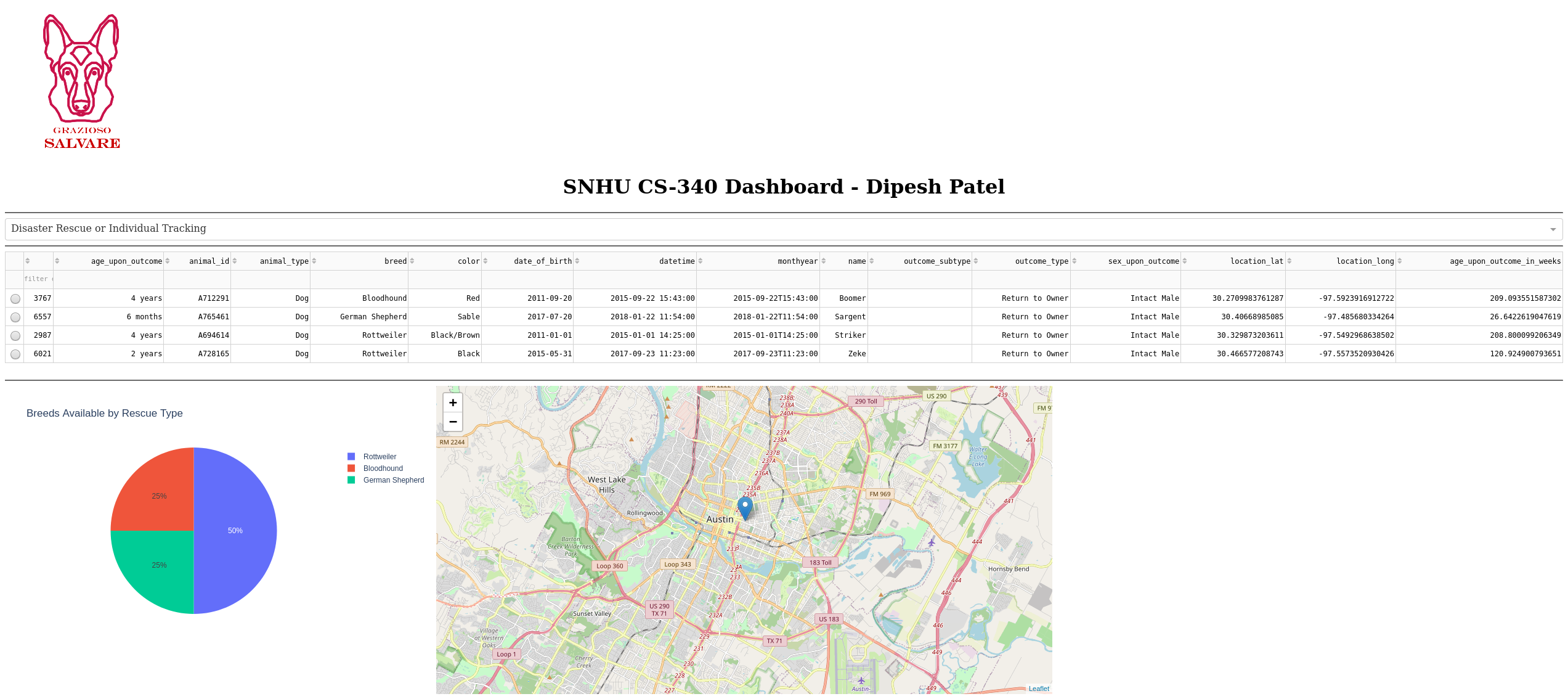
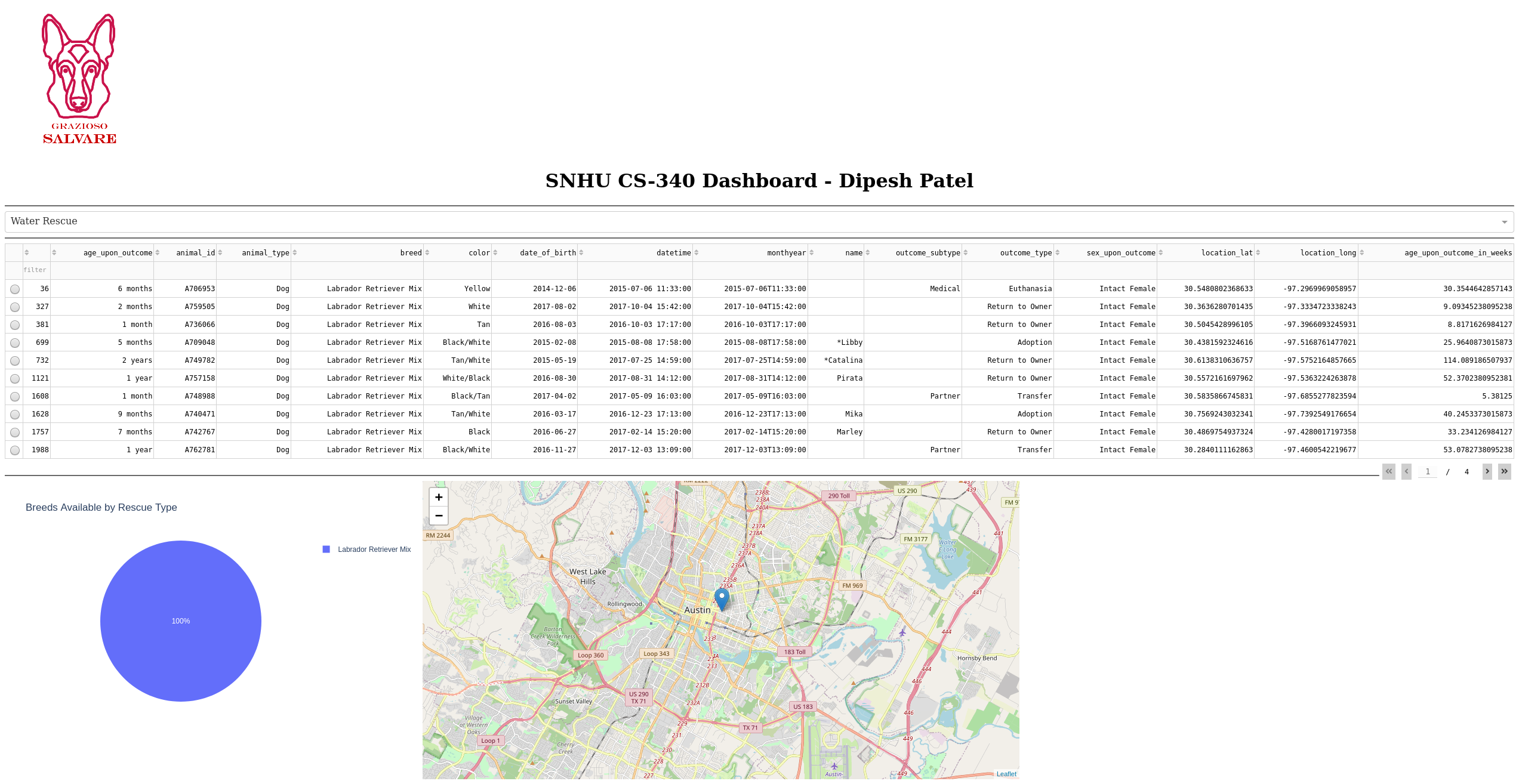
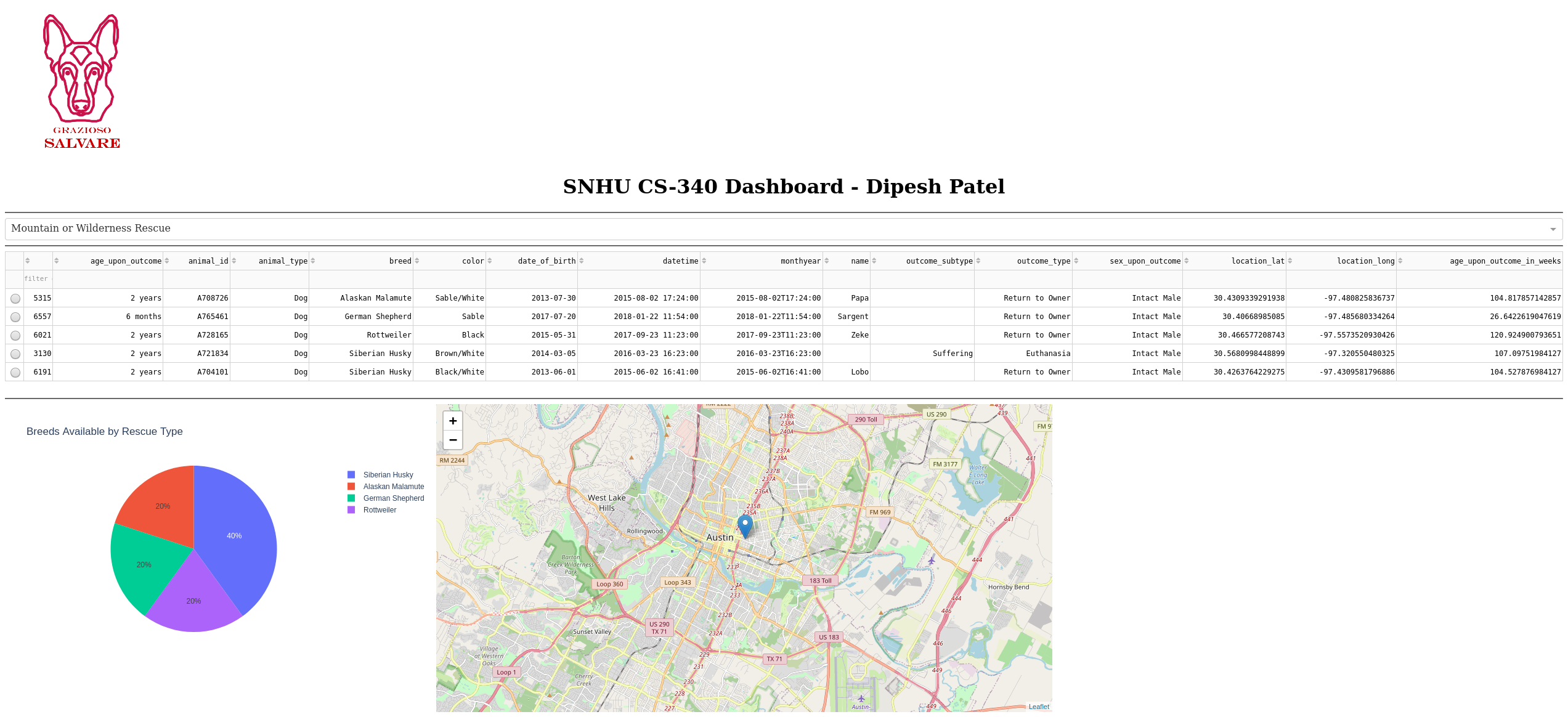
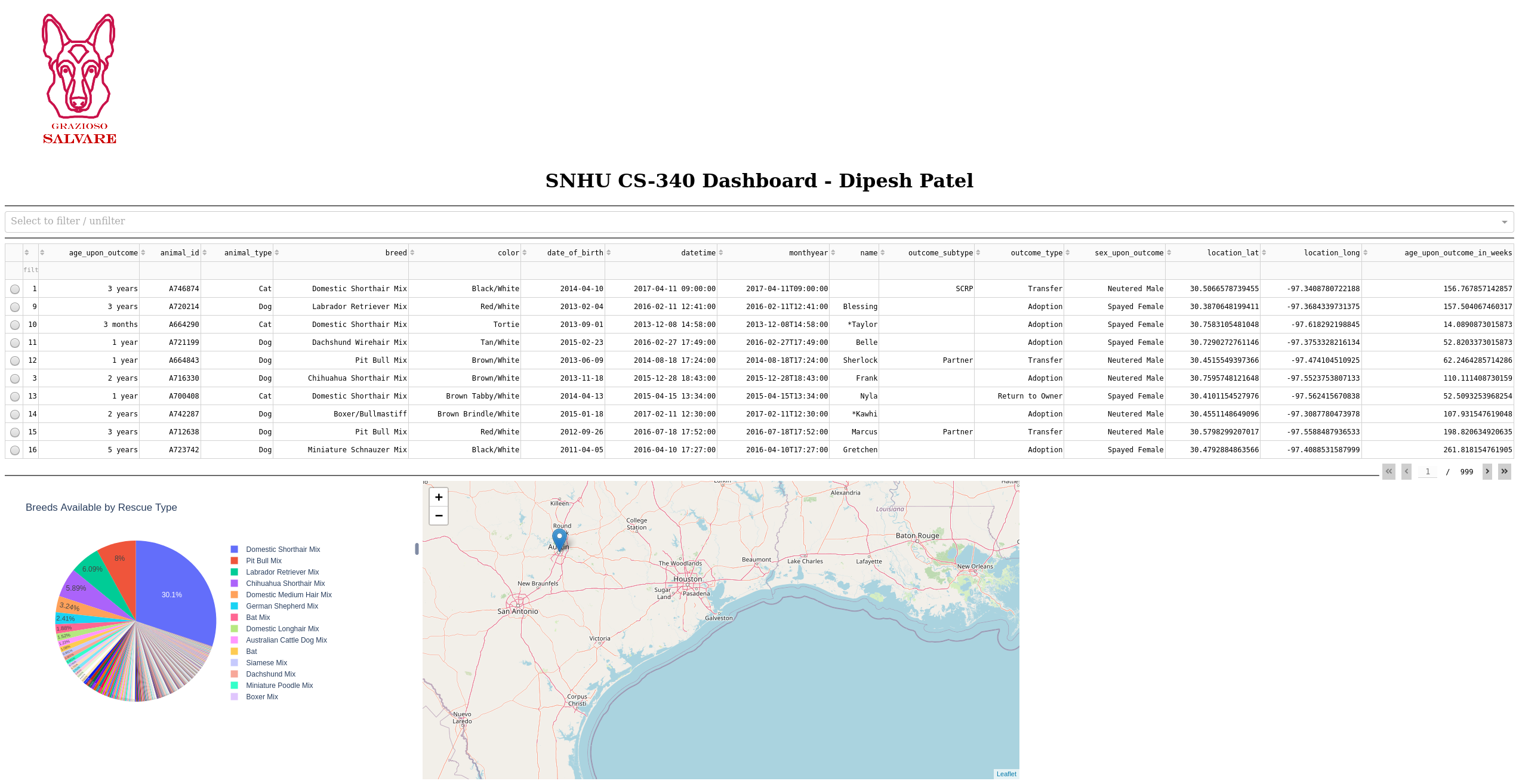
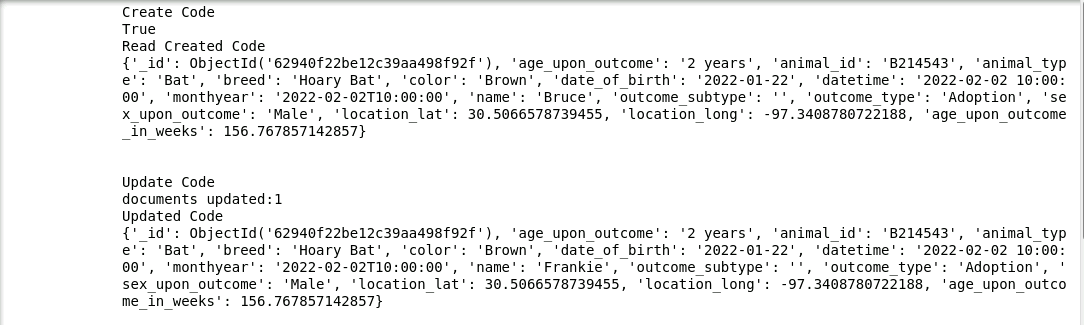
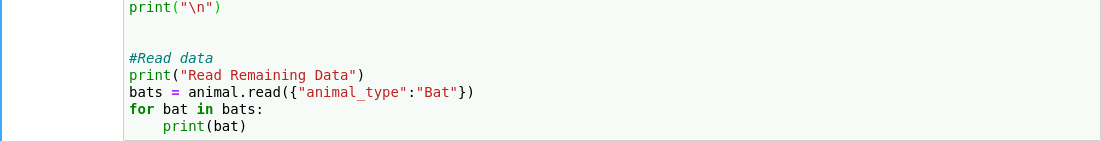
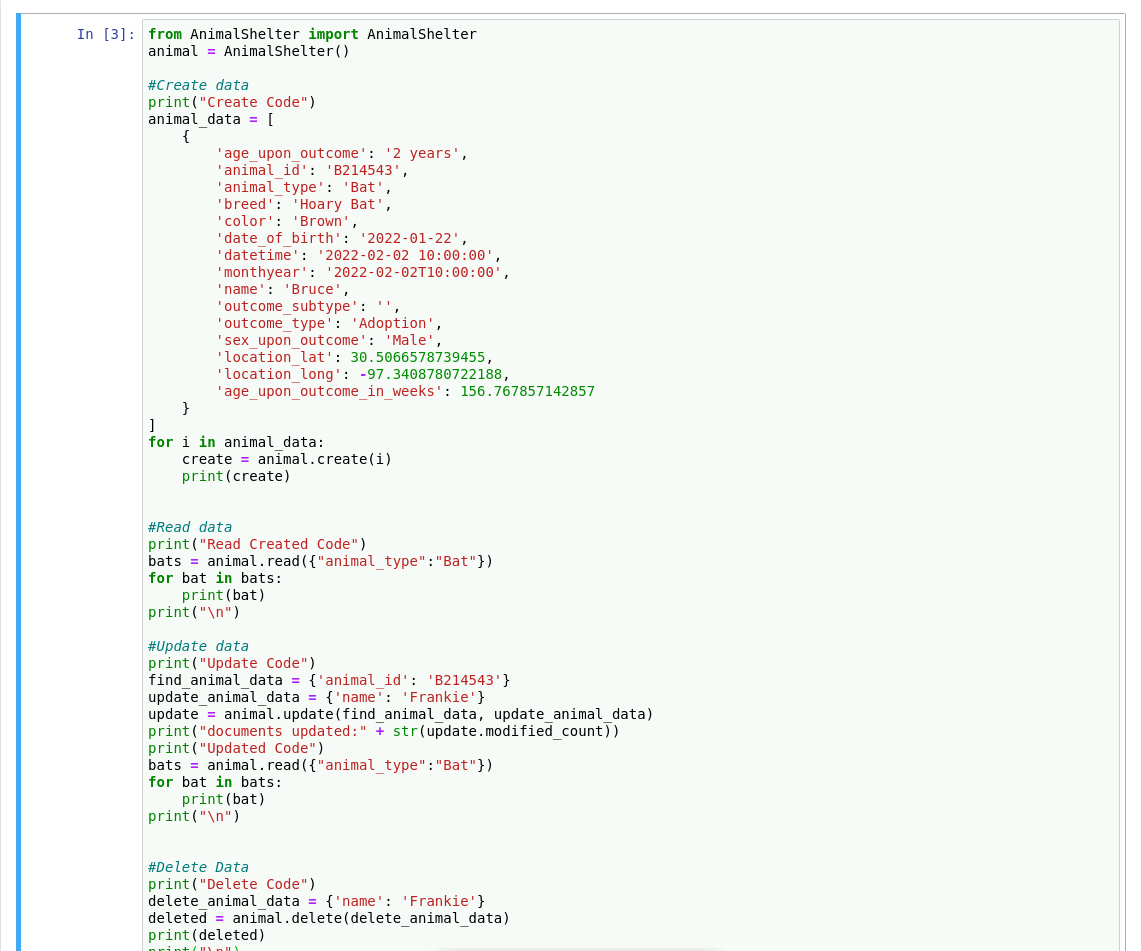
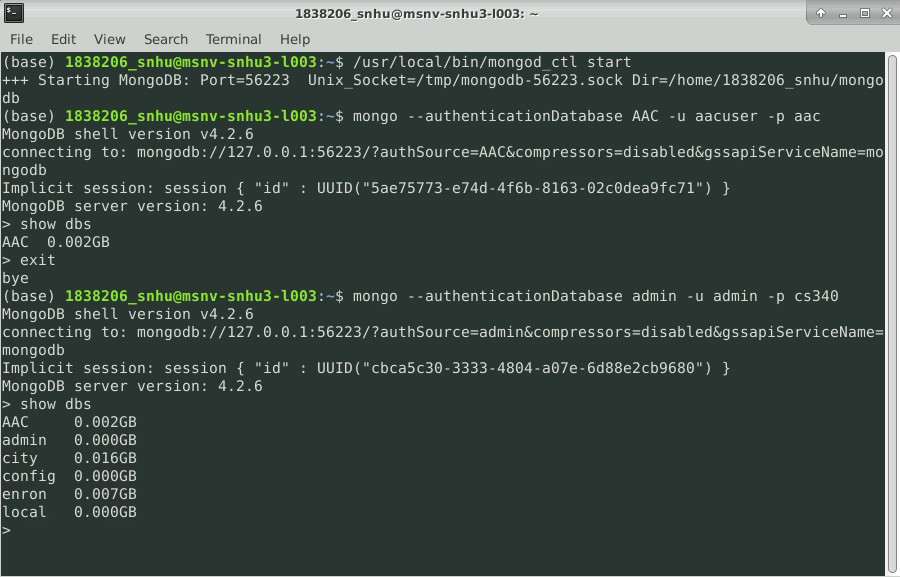
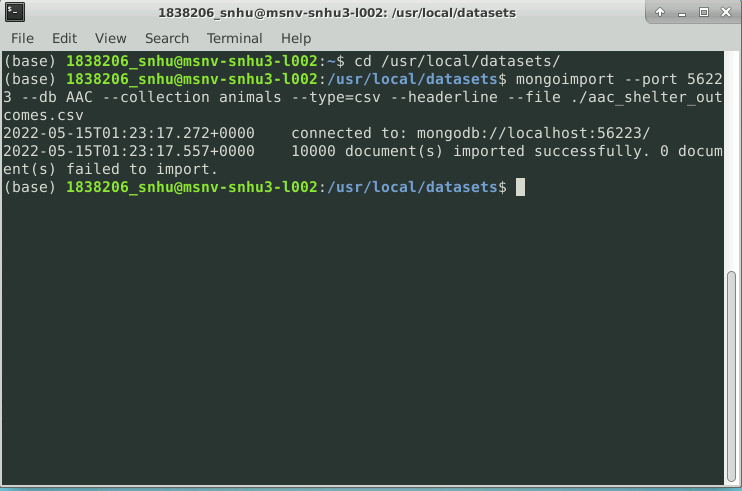
print("Read Code")

bats = animal.read({"animal\_type":"Bat"})

for bat in bats:

print(bat)

### Screenshots



<https://i.imgur.com/xH64mx7.gifv> (link to screencast)

## Roadmap/Features (Optional)

## Contact

Your name: Dipesh Patel