Lawrence Artl III
CS 340 22EW6
Final Project
August 11, 2022

Austin Animal Center Database Access README

About this Project

The user interface was designed with simplicity and flow in mind. Within a web browser window the database of animals is displayed, along with various visualizations of the data. This includes a location map of the selected animals. The interface includes three buttons with pre-set filters that utilize the MongoDB index feature, as well as a fourth “reset” button to repopulate the table with all documents.

Motivation

Every database needs a way to add and remove, read, and update data; this project provides a way to do that in as simple a format as possible, while still maintaining data validity within the database. The CRUD module can be used from the backend in order to update records via command line.

Mongo provides a very easy to setup and use database system, both locally and on the cloud. With relatively simple commands for retrieving documents or adding to the database, Mongo was the perfect tool for this project. Additionally the Pymongo library includes an array of methods used to easily access a database, lending to the ease of getting the project up and running.

The dash framework was used to create the user interface; the UI includes interactive visualizations of the data in the table, as well as custom filtering options that users can select or reset via buttons in the UI.

Getting Started

Download the "mongoCRUD.py" file to a folder on your local machine. In that same folder, install the ModuleSixMilestone_AAC.py. In the file, be sure to import mongoCRUD, and from mongoCRUD import the CRUD class. When instantiating the mongoCRUD object, pass a connection string that connects to either a local database or a database on the MongoDB cloud (Atlas). Examples of each are provided in the ModuleSixMileston_AAC.py file. You can also follow this video here.

This project was originally built with Sublime Text 3, with Python enabled. See this tutorial here for setting that up. The tutorial includes instructions for installing Python as well. Once that is complete, follow the tutorial here to get started using MongoDB Atlas and Python to connect.

Usage

All methods take a JSON dictionary object as a parameter, save for "#UPDATE" which takes in two such objects. Error checking is done in the CRUD module's methods. Following is a brief example using pseudocode.

Code Example

```
create(self, data):

        if data is None or data == {}                    #empty object is passed

                print error message

        else

                results = self.collection.find_one(data)    #search for the data

                if results is None                          #data is not already in db

                        self.collection.insert_one(data)    #insert the data to the db

                        print success message

                else                                        #data is already in db

                        print error message
```

Tests

Testing can be done through backend access. To test methods, first create an object of CRUD in your testing file:

```
c = CRUD()
```

Note: This README has been adapted from several sample templates, including: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Call each method with the CRUD object while passing in the appropriate data object or string:

```
#CREATE
c.create({"name":"David"})
 or
d = {"_id": 5, "name":"Lawrence"}
c.create(d)
```

All methods in CRUD take in one dictionary object, save for #UPDATE, as shown below:

```
#UPDATE
e = {"_id": 6, "name":"Larry"}
c.update(d, e)
```

Where object 'e' is the updated object and object 'd' is the object to be updated.

The 'refresh' method is used with Dash to refresh the database. It does this by removing all members and re-inserting a newly created database. This was created so users can remove or add entries in succession, and then commit the changes at once instead of one at a time.

Screenshots

The following screenshots show a generic set of tests that produce the results on the right. These tests are labeled as to which method they are focused on at the time.

```
c = CRUD()

c.builddb()

#CREATE
d = {"_id": 5, "name":"Smoke", "Age": 15}
c.create(d)
c.create(d)


#READ
c.read(d)
c.read(d)
c.read(None)
c.read({"name":"Puck"})
c.read({"name":"Lawrence"})
c.read({"name":"lawrence"})
c.read({"name":"Larry"})
c.read({"name":"Lawrence"})


#UPDATE
e = {"name":"Puck"}
c.update(d, e)
c.read(d)
c.update({"name":"Lawrence"},{"name":"Larry"})
c.read({"name":"Larry"})


#DELETE
c.delete({"name":"Puck"})
c.read({"name":"Puck"})
c.delete({"name":"Puck"})
```

Figure1: Tests

```
SUCCESS - accessed database
SUCCESS - built a new database
SUCCESS - Smoke was successfully added.
ERROR - Data Smoke already in database.
SUCCESS - found Smoke
SUCCESS - found Smoke
ERROR - No data was passed in
ERROR - Puck was not found
SUCCESS - found Lawrence
ERROR - lawrence was not found
ERROR - Larry was not found
SUCCESS - found Lawrence
SUCCESS - updated Puck
ERROR - Smoke was not found
SUCCESS - updated Larry
SUCCESS - found Larry
SUCCESS - Puck deleted
ERROR - Puck was not found
ERROR - That data was not found
[Finished in 2.8s]
```

Figure 2: Results

Screenshots

The following screenshots showcase the web-UI in action with various options enabled as indicated

Note: This README has been adapted from several sample templates, including: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

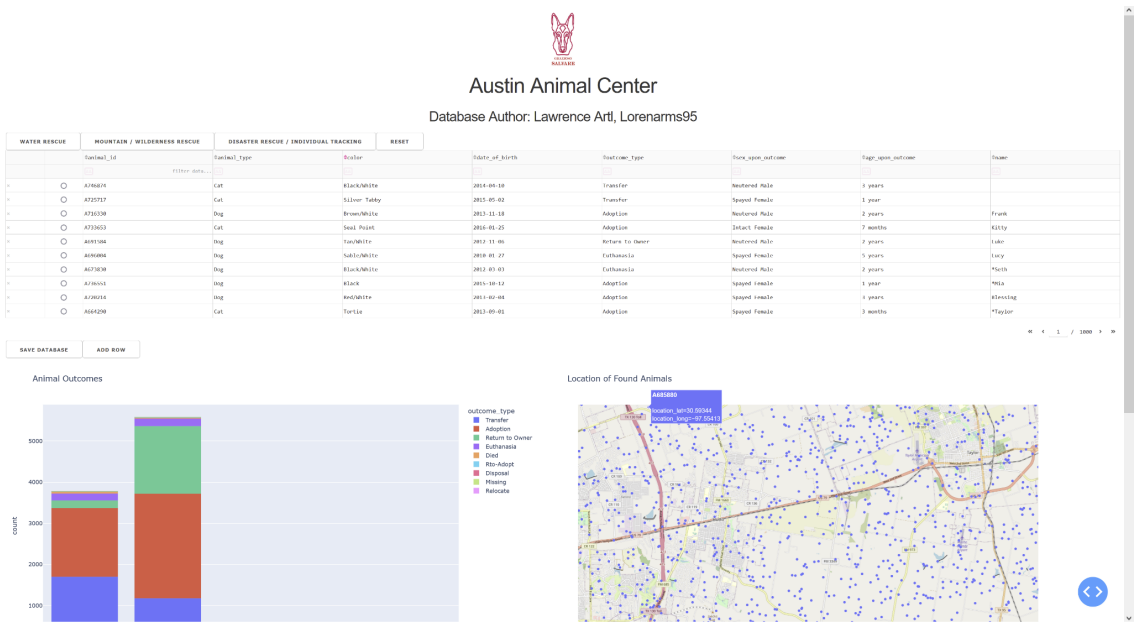

Figure 3: Main Page UI

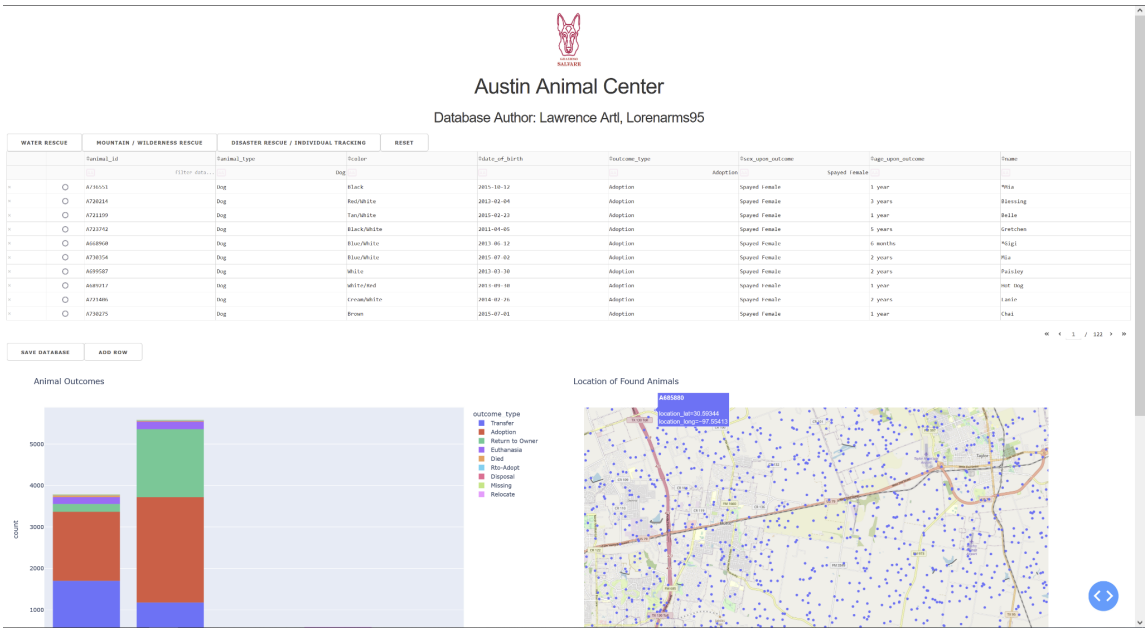

Figure 4: Custom Filtering

Note: This README has been adapted from several sample templates, including: [Make a README], [Best README Template], and [A Beginners Guide to Writing a Kickass README].

Figure 5: Data Visualization

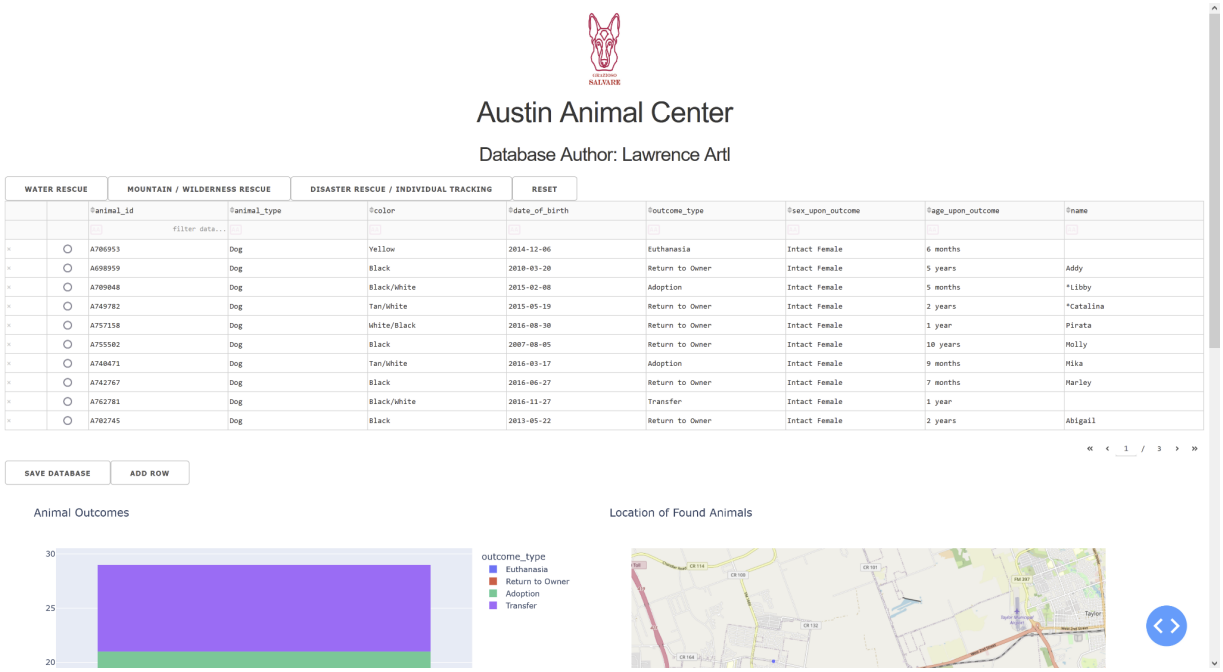

Figure 6: Water Rescue Filter Applied

Note: This README has been adapted from several sample templates, including: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

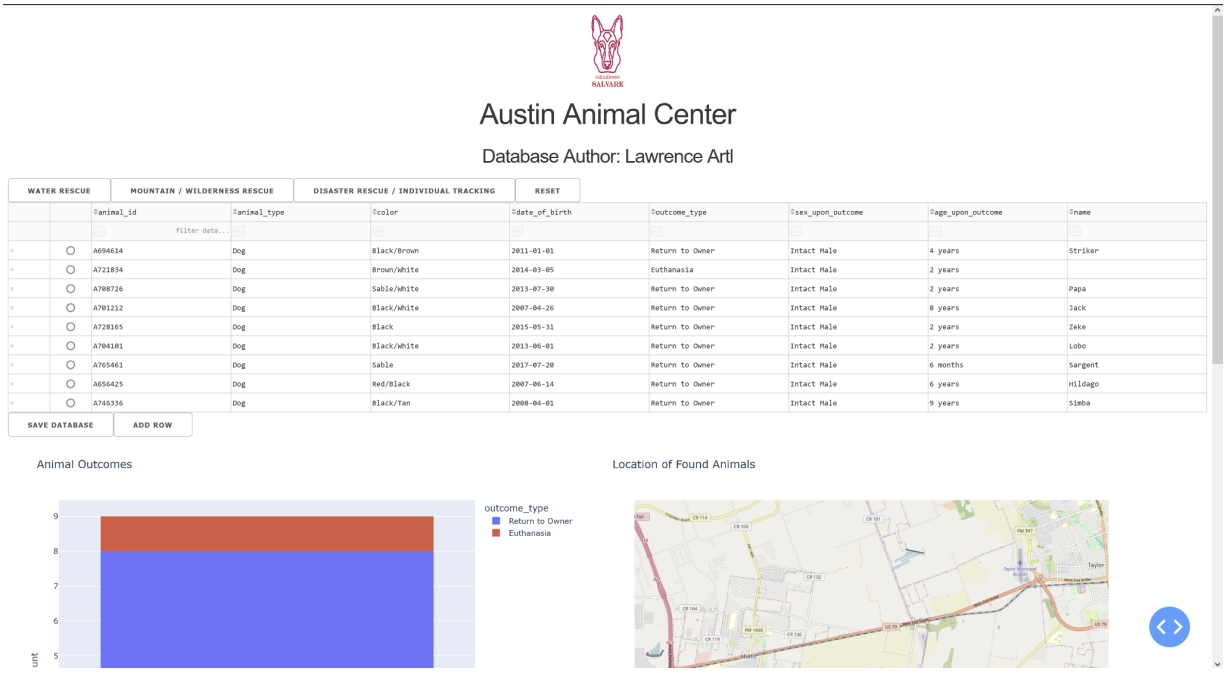

Figure 7: Mountain Rescue Filter Applied

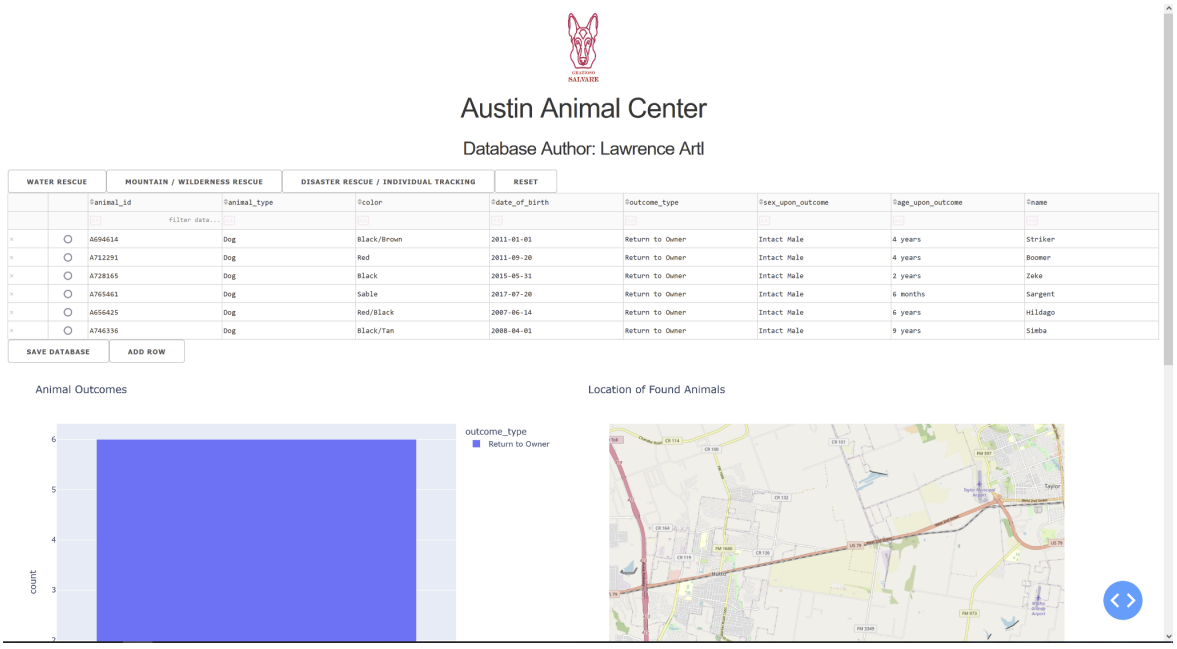

Figure 8: Disaster Rescue Filter Applied

Note: This README has been adapted from several sample templates, including: [Make a README], [Best README Template], and [A Beginners Guide to Writing a Kickass README].

Roadmap/Features

Future updates will include a more interactive way to use the database CRUD module, including:

- Drop down menu for most common searches instead of buttons
- More visualizations for ages, outcomes, and etc.
- Front-end UI for testing back-end

Contact

Lawrence Artl III: GitHub LinkedIn Email artllj.com

Resources

https://www.cdata.com/kb/tech/mongodb-python-dash.rst

https://www.youtube.com/watch?v=2pWwSm6X24o

https://www.youtube.com/watch?v=DWqEVpOfYxE

https://www.youtube.com/watch?v=IHQeDEn38BQ

https://www.youtube.com/watch?v=hSPmj7mK6ng

https://www.youtube.com/watch?v=XOFrvzWFM7Y

https://www.mongodb.com/docs/

https://dash.plotly.com/basic-callbacks

https://dash.plotly.com/interactive-graphing

https://pymongo.readthedocs.io/en/stable/index.html

Note: This README has been adapted from several sample templates, including: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.