Bryce Jensen
CS-340-Q7703
7-2 Project Two: README
2/21/24

About the Project

The Grazioso Salvare Canine Rescue Training Dashboard is a project developed by Global Rain to assist Grazioso Salvare, an innovative international rescue-animal training company. One goal of this project is to identify and categorize dogs that are capable of search-and-rescue training.

This project takes existing data from 5 non-profit animal shelters in the Austin, Texas region. This enable Grazioso Salvare to efficiently identify potential candidates for their training programs. Some key features of the project include filtering dogs based on their age, breed, and other characteristics that may indicate a suitability for different types of rescue missions such as water rescue, mountain rescue, disaster recovery, or scent tracking.

Motivation

This project was created to address the critical need to find dogs suitable for search-and-rescue training. The main motivation of this project is to improve the effectiveness and impact of Grazioso Salvare's training programs.

The decision to make the project open source is to allow collaboration and help the rescue animal training community. We are hoping to enable other similar organizations to adapt this software to their specific needs, enhancing the impact of canine rescue efforts around the world.

Getting Started

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

- Clone the Grazioso Salvare Canine Rescue Training Dashboard project from Github:
    - Open a command prompt.

    - `git clone https://github.com/global_rain/grazioso-salvare-canine-rescue-dashboard.git`

- Download and install and set up MongoDB (Community Edition)

- Import the data:

```
brystonjensen_snhu@78e20b13519a:/usr/local/datasets$ mongoimport --username="${MONGO_USER}" --password="${MONGO_PASS}" --port=${MONGO_PORT} --host=${MONGO_HOST} --db=AAC --collection=animals --authenticationDatabase=admin --type=csv --headerline --drop ./aac_shelter_outcomes.csv
2024-02-07T22:07:55.444+0000    connected to: mongodb://localhost:27017/
2024-02-07T22:07:55.444+0000    dropping: AAC.animals
2024-02-07T22:07:55.627+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
```

- Launch the MongoDB shell:

```
brystonjensen_snhu@78e20b13519a:/usr/local/datasets$ mongosh
Current Mongosh Log ID: 65d79ab2cf0aa597249abf04
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          7.0.2
Using Mongosh:          2.0.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

admin>
```

- Add a user to ensure user authentication to the database and collection:

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
admin> db.createUser(
... {
... user: "aacuser",
... pwd: "simplepass",
... roles: [ { role: "readWrite", db: "aac" } ]
... }
... )
{ ok: 1 }
admin> db.getUsers()
{
  users: [
    {
      _id: 'admin.aacuser',
      userId: new UUID("9ad0b099-5319-4f92-9a33-53f15402c8d5"),
      user: 'aacuser',
      db: 'admin',
      roles: [ { role: 'readWrite', db: 'aac' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    },
    {
      _id: 'admin.root',
      userId: new UUID("e530898f-0b7a-4bf0-ad2d-044f09c07e81"),
      user: 'root',
      db: 'admin',
      roles: [ { role: 'root', db: 'admin' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
admin>
```

- Run the `GraziosoSalvareDashboard.py` file.

- Open http://127.0.0.1:8050/.

Bryce Jensen
CS-340-Q7703
7-2 Project Two: README
2/21/24

Usage and Testing

Some things the Dashboard can do:

Display a pie chart of the data, show a bar graph of the data, sort and filter the data from any column, filter the data based on premade queries (buttons above the table), and display the location of animals on a map.

A Demo of the dashboard's features (file is also included in the project directory):

GraziosoSalvareDashboard.mp4

The project includes a CRUD class that can create, read, update, and delete documents from the database. The usage and testing examples for this class is included below and each can be used in the dashboard as necessary:

`CRUD()`

The constructor for the CRUD Class Initializes the MongoDB server.

Example Constructor:

```
crud = CRUD('aacuser', 'simplepass')
```

`create()`

Creates a new item in the database. Takes one argument for the data to be added. Must be in dictionary form.

Example Test:

```
newAnimalTestData = {
    'age_upon_outcome': '10 years',
    'animal_id': 'BGJ0715',
    'animal_type': 'Dog',
    'breed': 'Border Collie',
    'color': 'Black and White',
    'date_of_birth': '2013-11-19',
    'datetime': '2024-02-06 18:21:36',
    'name': 'Doug',
    'outcome_subtype': 'SCRP',
    'outcome_type': 'Adopted',
    'sex_upon_outcome': 'Neutered Male',
    'location_lat': 30.6525984560228,
    'location_long': -97.7419963476444,
    'age_upon_outcome_in_weeks': 533
}

created = crud.create(newAnimalTestData)
```

```
print(f"Item created: {created}")
```

`read()`

Finds and prints all documents of the search. Takes one argument as the search query. Can be as long as needed but must be in the correct Mongo format for queries.

Example Test:

```
read_count = crud.read(
    {"animal_id": "BGJ0715"})

print(f"There are {read_count} document(s) that fit the query.")
```

`update()`

Modifies a current document. Takes two arguments, one as the search to find the documents to be edited (must be in the correct Mongo format for queries), and the other is the data to be updated (must be in dictionary format).

Example Test:

```
updated_count = crud.update(
    {"animal_id": "BGJ0715"},
    {"outcome_type": "Adopted"})

print(crud.read({"animal_id": "BGJ0715"}))
print(f"There was {updated_count} document(s) changed.")
```

`delete()`

Deletes a current document. Takes one argument as the search query. Can be as long as needed but must be in the correct Mongo format for queries.

Example Test:

```
deleted_count = crud.delete(
    {'animal_id': 'BGJ0715'})
print(f"There was {deleted_count} document(s) deleted.")
```

Tools

MongoDB

MongoDB was used as the document database management software. Its documents made it easy to manage and index the data.

Bryce Jensen
CS-340-Q7703
7-2 Project Two: README
2/21/24

PyMongo

PyMongo was used throughout the project for interfacing with the Mongo database through Python. This is what allowed queries and filtering directly from the `GraziosoSalvareDashboard.py` file.

Plotly Dash

Plotly Dash is a Python framework for making interactive web pages. It only uses Python to build and format the site meaning HTML, CSS or JavaScript aren't necessary to know. This is what runs the webpage with the data table, charts, and maps.

Jupyter Notebook

Jupyter Notebook is a web-based application that can create and share documents with code, text and equations. While not necessary, any Python IDE can be used, this was a simple tool that was used as an easy way to create and run the python code.

Docker

I used Docker to run the virtual Linux server and MongoDB locally. BIG shoutout to user `moonlitaltar` on docker hub for putting together a container that ran it so well.

https://hub.docker.com/r/moonlitaltar/cs340

Challenges

The hardest part of this for me was getting it running locally even though I had some help from other students at SNHU. I'm not a fan of Apporto and have nothing but issues with it and I often learn a lot by downloading the software and running it myself from scratch. Running it locally is my preferred method in most classes. It also had a LOT of components that make this work and keeping track of them all to make it run correctly was a lot of work.

Bryce Jensen
CS-340-Q7703
7-2 Project Two: README
2/21/24

Contact

Bryce Jensen
bryston.jensen@snhu.edu