Project Two Read Me

Author: David Robey

2/14/2023

**Table of Contents:**

1. Purpose
2. Functionality
3. Tools Used
4. Development Steps
5. Challenges Encountered

**Purpose:**

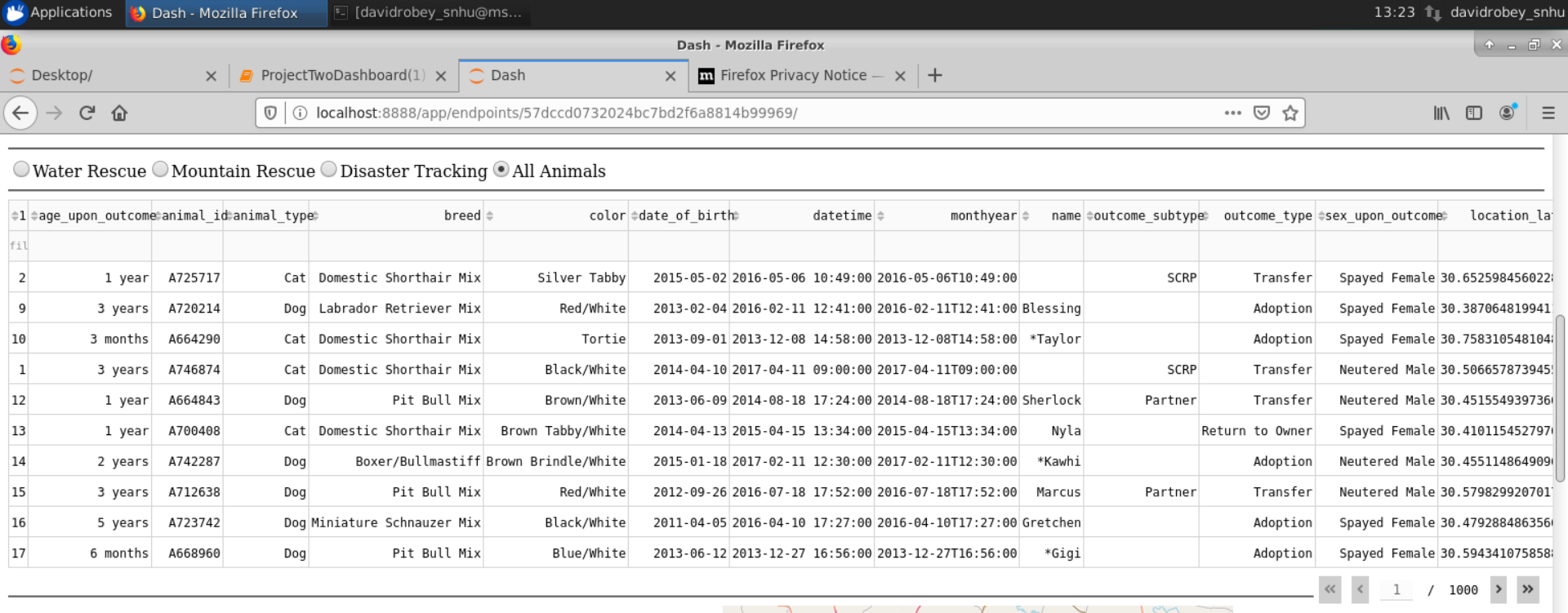
The purpose of this project was to create a full stack web application using MongoDB for the base, Python for the middleware, and dash framework with python for the client-side. The company Global Rain tasked me with helping Grazioso Salvare to create a web-based application. This application will help in searching and training dogs for disaster reliefs such as water rescue, mountain rescue, and disaster tracking. They Austin Animal Shelter has more than just dogs, as well. This application will allow them to search through candidates for their rescue relief with multiple search options and features.

**Functionality:**

As you can see below, the company image is loaded at a size that reduced by 25%. This allows the user to be able to see the whole image without having to scroll down as far (highlighted in red). There also are the radio buttons, that start with all “All Animals.” This is also the reset. (highlighted in blue).A picture containing text

Description automatically generated

Scrolling down there is a table that has an item size of 10. You can continue to the next page or previous page (highlighted in red) which will display the next set of animals. There is a search bar where you can type in your criteria at the top which will update the dashboard to display those criteria, if the animals exist within the database (highlighted in blue).



Scrolling down furth on the page we will fill find a pie graph located on the left (highlighted in red) and a geolocational chart (highlighted in blue). The pie graph will update with what is currently present on the animal within the data table. Examples of this will change will be provided with context as to what was done to produce the change. The pie graph will display the percentage of what breed is currently being viewed within the data table. The geolocation chart will update with the location of the animal at the top of the page.

Graphical user interface, application

Description automatically generated

Below an example of this change in both the geolocation map and the pie graph are seen. This occurred when using the water rescue button located at the top of the datatable.

Graphical user interface, application

Description automatically generated

The radio buttons are easy to update in the future. Below is a look at the code used to get the results for each of the buttons.

Water rescue: requires the animal to be a Labrador retriever mix, Chesapeake bay retriever, or a Newfoundland with an intact female sex as well as age between 26 and 156 weeks.



Mountain rescue: requires the animal be a German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, or a Rottweiler with an intact male sex as well as an between 26 and 156 weeks.

Graphical user interface, text, application

Description automatically generated

Disaster relief: requires the animal to be a Doberman pinscher, German Shepherd, Golden Retriever, Bloodhound, or Rottweiler with an intact male sex, and an age between 20 and 300 weeks.

Text

Description automatically generated with medium confidence

Its possible to change the criteria by adding or removing more breeds from the search. You can also add extra query variables, or change the range within each of the radio buttons.

**Tools Used:**

*MongoDB:* MongoDB was used as a base layer. This allowed me to have noSQL database that was able to utilize queries to create, read, update, and delete documents (CRUD). MongoDB was used to add the database, collection, and documents. It was also used to create admin, and aacuser accounts for authentication and CRUD purposes. The aacuser account does not have access to all other databases, only admin does.

*CRUD.py:* CRUD.py was python file created for the purpose of Creating, Reading, Updating, and Deleting documents. It was implemented for being able to query the database. It can also be allowed later to add and delete documents (animals) from the database.

*Dash framework:* The Dash framework was used as the client-side layer. This is what displayed the data to the client. It utilized app callback to then use python functions in order to call CRUD functions, create a geolocational map, as well as an updatable pie chart. Call back functions were provided what is to be outputted with what inputs. As well as styling for the client-side view. The allowed for there to be rows and columns that updated upon queries.

*Helpful Recourse links used*

<https://dash.plotly.com/dash-core-components> - used for radio items and introduction to graphs in dash

<https://dash.plotly.com/datatable> - used for creating dash table, as well as page numbers, app callbacks functions

<https://plotly.com/python/pie-charts/> - used for creating the updating pie chart based off of the animal breed

**Development Steps:**

The development of this program began with importing the Austin Animal Center CSV file into mongoDB with the name of the database being “aac” and the collection being named “animals.” Two authenticated users were created and tested. One of the users was an admin user that had the ability to gain full access to all databases and create more users within mongoDB. The second user created was aacuser that had privileges to the aac database with respect to create, read, update, and delete for documents within the database within the mongoDB shell. MongoDB would be used as the backend/database for this fullstack application.

From there CRUD.py was developed to allow the user to access the create, read, update, and delete features within a python program. This was tested within a .ipynb file that allowed basic reading in a JSON format to output to the user. This was the middleware for the application that would allow the user to communicate to the database.

Finally, the development of the web-application concluded with the development of the user interface with the Dash framework. The .ipynb file logged into the “aacuser” account, loaded, the middleware file (CRUD.py), and would receive all the documents within the database and display them 10 at a time through pages. This dash framework also allowed the ability to search the file through columns with queries that matched users results. The development of a geolocation chart was developed at the request of the client which would take the coordinates and display them below the loaded documents. Radio buttons were used within the dash framework to allow the user to search for animals that fit a particular set of traits that were requested by the client. Finally, a pie graph was created to display the current page of search results and chart the percentage of what breeds are on the current page.

**Challenges Encountered:**

I encountered a lot of challenges using the jupyter notebook app. There wasn’t a lot of information when something didn’t run properly with syntax errors. I also had an instance where it wasn’t displaying the client-side view. I had to restart a new .ipynb and copy what I had done and it started working without error even though I didn’t omit anything from the copy paste. During development of the geolocation map and graph portion I couldn’t get either to display. I ended up deleting both instances and retyped them out and they started displaying then. I have no idea where these issues were coming from but I eventually got it to work. I was kind of lost at first with the pie chart I used the online source provided in the recourses, but it only did so much for understanding and I had to realize that I needed to keep it simple and use what was provided in the comments and fill in the blank with the correct function call as well as the correct variables. This took a lot of playing around to get both the map and the graph to display while having the proper display.