

**Developer**: Joaquin Esguerra Jr

**Course**: CS 340 Client/Server Development

**Project Two**: Grazioso Salvare - README

**Instructor**: Dr. Kellogg

**Date**: February 25, 2024

**Agenda:**

1. Project Purpose
2. Required Functionality
3. Tools Used to achieve functionality.
4. Development Steps to complete the project.
5. Challenges Encountered/overcome.
6. **Project Purpose:**

The project purpose was to develop a full stack web application utilizing the MongoDB within the terminal for the base, a Python IDE for the middleware, and Dash framework with the Python IDE for the client-side. Global Rain, a software engineering company that specializes in custom software design and development assigned my team to work this project for Grazioso Salvare. Grazioso Salvare is an innovative international rescue-animal training company that needs an application to help in searching and training dogs for disaster relief specific to disaster tracking, water/mountain rescue. The application will provide the client with features and functionality to search for candidates for their rescue relief along with additional features and options.

1. **Required Functionality:**

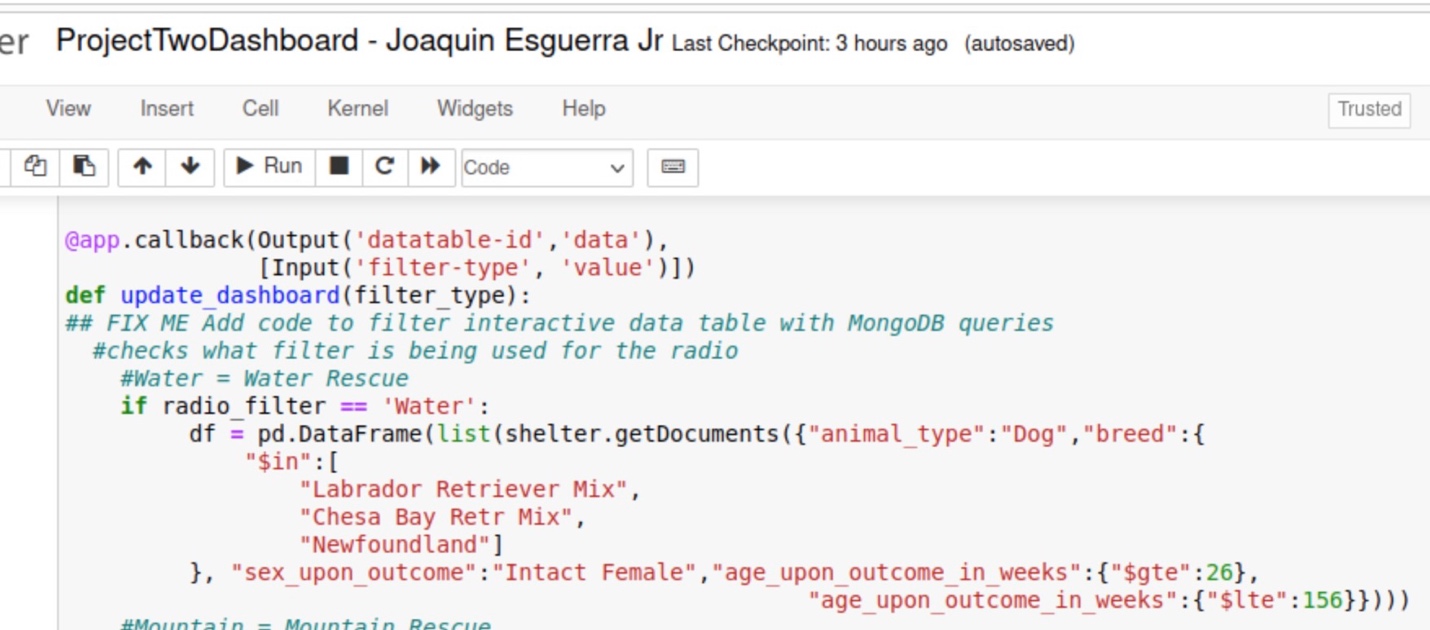
Beginning with the main Dash screen, Grazioso Salvare’s logo will be loaded at a size that is reduced by 25%. This will display the company’s logo at the top of the screen eliminating users from having to scroll down to see the entirety of the logo. At the base of the screen the required “radio buttons” will be listed and include the following buttons, “All Animals”, “Disaster Tracking”, “Mountain Rescue”, and “Water Rescue”.

Below the company logo there will be an animal listing chart. User can scroll through the list and move between pages of animals by using the arrow markers. Also, on the page there will be a search bar where users can enter in specific criteria so that the page is sorted based on the information entered. Once the criteria is entered the page will update based on the users preferences.

Below the animal chart the users screen will display a pie graph and a geolocational chart. Based on the animal selected from the chart the pie graph will update with present information from the data table. Changes will be outputted with this feature once the context is adjusted. The pie graph basically represents a percentage of the selected the is selected withing the animal chart. The animal’s location will be updated and show on the geographical chart. By using the water rescue button, we can display examples of both pie graph and geographical map as they process through updates.

Looking at future updates to the system, radio buttons can easily be adjusted to meet future needs. I’ve included screen shots of the required code to output the expected results for the radio buttons.

**Water rescue**: Requirements include the animal being a Labrador retriever mix, Chesapeake Bay retriever, or a Newfoundland. Water rescue also requires the animal to have a sex of an intact female along with meeting age requirements between ages 26 to 156 weeks.



**Mountain rescue:** Requirements include the animal being a German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, or a Rottweiler. Mountain Rescue also requires that the sex of the animal be an intact male along with meeting age requirements between ages 26 to 156 weeks.



**Disaster Relief**: Requirements include the animal being a Doberman pinscher, German Shepherd, Golden Retriever, Bloodhound, or Rottweiler. Disaster relief also requires that the sex of the animal be an intact male along with meeting age requirements between ages 20 to 300 weeks.

A screenshot of a computer

Description automatically generated

By adding or removing breeds of animals from the search the criteria can be changed. Within each radio button, extra variables can be queried along with changing the range of the search.

1. **Tools Used to achieve functionality.**

***MongoDB:*** MongoDB was utilized as the foundation for the project. This enable the noSQL database for the CRUD module that utilized queries to create, read, update, and delete documents. MongoDB was also utilized when adding the database, collection, and documents. MongoDB was also utilized when creating the admin, and aacuser account for both the CRUD and authentication purposes. The account for the aacuser only has access to the admin databases.

***animal\_shelter.py****:* animal\_shelter.py was developed as a python file with the purpose of creating, reading, updating, and deleting documents within the database set. Its primary purpose was implemented to query the database. An additional feature of the py file can allow later updates to delete or add additional animals to and from the database.

***Dash framework:*** Moving to the client-side layer, the Dash framework was developed. The dash framework worked to display all the data on the client side. App callback was utilized for the python functions which worked to call the CRUD functions from the animal\_shelter.py, create the geolocational map, and update the pie chart. The call back functions were provided with the functions that would be outputted based on input given along with the styling on the client-side which then outputted columns and rows updated upon the queries.

**Resources:**

*Dash core components*. Plotly. (n.d.-a). https://dash.plotly.com/dash-core-components

*Dash datatable*. Plotly. (n.d.-b). https://dash.plotly.com/datatable

*Pie*. Pie charts in Python. (n.d.). https://plotly.com/python/pie-charts/

1. **Development Steps to complete the project.**

To begin development of the program the team first had to import the Austin Animal Shelter csv file within the terminal and MongoDB. The call name of the database with the MongoDB is “AAC” and the collection is “ANIMALS”. Once the “CSV” file was imported two users who were authenticated were created and tests were run. The first user was the “admin” whom had total access to the databases with the ability to create more users within MongoDB. Once the “admin” user was created we created the “aacuser” that had access to the AAC database for the sole purpose of creating, reading, updating, and deleting documents within the databases shell. For this full stack development, MongoDB would be used as the backend database. After completing the initial development steps with importing the csv and authenticated users within the database the CRUD module was developed as the animal\_shelter.py. file. The animal\_shelter.py file allowed users access to create, read, update, and delete the python program features. Tests were then ran with a .ipynb file that worked to read in JSON format which was outputted to the user. The .ipynb file works as the middleware for the application to communicate to and from the database. We wrapped up development of the web application with the dash framework which developed the user interface. Our .ipynb file logged into the account of the “aacuser” user, loaded the user, then the middleware animal\_shelter.py file, which then outputted the dash framework to receive all the documents displaying the data ten items at a time through multiple pages. The framework also allowed searches to be conducted based on user input which queried the columns in rows to align with user search criteria. Per the request of the client, the geographical chart was developed to output a display for the coordinates based on selected animals. This chart will be displayed below the loaded docs. Within the dash framework “radio” buttons were developed to allow users to search animals based on specific traits based on client requests. Lastly the pie graph was developed to output a display chart that represented percentages of breeds based on the current page results.

1. **Challenges Encountered/Overcome:**

Using the jupyter notebook presented multiple challenges for the development team. We encountered errors with accessing the MongoDB along with syntax errors and errors with modules not being found which effected the required output. This specifically caused problems when working with the dash framework on the client-side. We attempted to create new python and .ipynb files by transferring code already developed into new files, but still ran into output issues. For some reason the dashboard wouldn’t communicate with our animal\_shelter.py file which resulted in being unable to output the client-side screens. Septically the logo, data rows/columns, and the pie/geographical chart. After acquiring some advice through instructions and resources we were able to move further into development, but still faced challenges with delivering the expected outputs although our code was correct. We imagine it’s something very simple were missing in the development steps and will continue to troubleshoot as we move forward with development.