**Project Two: README**

Amanda Kerr

CS-340: Client/Server Development

Prof. Chao Ling

December 15, 2024

A dog with a black background

Description automatically generated

**Grazioso Salvare Animal Rescue Database README**

**About This Project**

The Animal Rescue Database is a custom software application produced by the Global Rain team for Grazioso Salvare, a specialized rescue-animal training organization. Grazioso Salvare identifies and trains dogs for search and rescue operations. With this software, we hope to streamline their process of finding suitable dogs by automating the identification of candidates from local shelters.

There are two primary motivations behind this project:

* First, to help Grazioso Salvare find their perfect candidates faster by utilizing shelter data.
* Second, to create a flexible and open-source database that other rescue organizations can customize to their needs.

**Functionality Requirements**

Grazioso Salvare requested the program to have the following:

* An interactive data table that allows for simple filtering by each column
* An interactive filter with the following presets:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster or Individual Tracking
  + Reset
* A graphical representation of the currently displayed data
* A map highlighting the location of the animal currently selected

**Installation**

The following are necessary to run the database:

* MongoDB (<https://www.mongodb.com/try/download/community>)
  + Ver. 6.0.13
  + MongoDB is a flexible, NoSQL database that is perfect for storing information about our animals. It can handle how we want to organize our data and grow with us as we add more shelters and animals.
* Python (<https://www.python.org/downloads/>)
  + Ver. 3.9
  + Python is easy to learn, powerful, and works well with MongoDB. It is open-source and can run on many computers, making it accessible to other rescue organizations. Python can also grow with us as we add more features and data, making it a great choice for the future.
  + Additional Dependencies:
    - PyMongo (<https://www.mongodb.com/docs/languages/python/pymongo-driver/current/get-started/download-and-install/>)
      * PyMongo is necessary for Python to interact with MongoDB.
    - Numpy (<https://numpy.org/install/>)
      * Numpy allows numerical computing within Python. Pandas and Plotly libraries may be utilizing Numpy for data manipulation.
    - Pandas (<https://pandas.pydata.org/docs/getting_started/install.html>)
      * Pandas is primarily used for data analysis. We are using it to manage data and filter data in our program. Pandas integrates seamlessly with Dash and Plotly.
* Jupyter Notebook (<https://jupyter.org/install>)
  + Ver. 6.4.8
  + NOTE: Anaconda offers a simple solution for viewing Jupyter Notebook (https://www.anaconda.com/download)
  + Jupyter Notebook is an excellent tool for showcasing how this software works. It helps to write and test the code quickly. It also allows for sharing work and including comments and explanations with the code.
* Dash (<https://dash.plotly.com/installation>)
  + Ver. 2.18.0
  + Dash is a simple framework for building web apps with Python. Dash allows us to use HTML to create visuals and turn the data into a filterable table.
* Plotly (<https://plotly.com/python/getting-started/>)
  + Ver. 5.24.1
  + Plotly is a library that supports creating interactive graphs in Python. This allows us to visualize our rescue data as a pie chart.

Setup:

1. Using your terminal:
   1. Install MongoDB
   2. Install Python and necessary dependencies
   3. Import the data using Mongo Import



NOTE: The above import statement uses the Austin Animal Center database and generic credentials; you may need to update it appropriately.

1. Ensure all program files are housed within the same folder.
2. Open the program file with Jupyter Notebook.

**Getting Started**

The database for this project is configured for secure access. You can test the functionality using the included Austin Animal Center data with the following credentials:

* USER = ‘aacuser’
* PASS = ‘SNHUcs340’
* HOST = (see your terminal for this information) \*
* PORT = (see your terminal for this information) \*
* DB = ‘AAC’
* COL = ‘animals’

\*NOTE: You can view the HOST and PORT details in your terminal by using “printenv”

If you choose to import your own dataset, you can start with any CSV file and import it via your terminal using Mongo Import.

**Usage**

The data table allows users to refine the data by entering a filter term in a column. The filter is case-sensitive, and the results will match exactly. The user can filter with multiple rows. When the user selects a cell, that cell is highlighted in red.

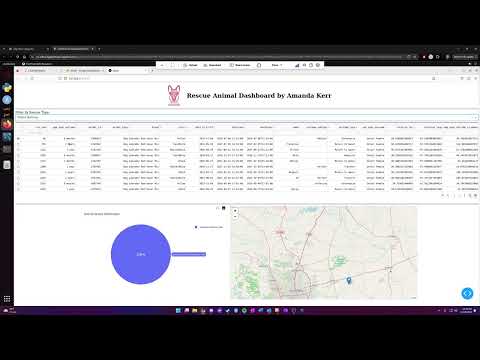
The program comes with premade filters in a dropdown list for Water Rescue, Mountain or Wilderness Rescue, and Disaster and Individual Rescue. Each rescue uses the following queries:

* Water:
  + Breeds: Labrador Retriever Mix, Chesapeake Bay Retriever, and Newfoundland
  + Sex: Intact Female
  + Age: 26 weeks to 156 weeks
* Mountain or Wilderness:
  + Breeds: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, and Rottweiler
  + Sex: Intact Male
  + Age: 26 weeks to 156 weeks
* Disaster or Individual:
  + Breeds: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, and Rottweiler
  + Sex: Intact Male
  + Age: 20 weeks to 300 weeks

The filter will only produce results matching these metrics exactly. This means that only purebred dogs will be produced from the search.

Additionally, the web app includes a graphical breakdown of the breeds found on the data table and filtered results. Information currently shows the percentage of specific breeds per query. This feature shows cat breeds if their breed data is included.

Lastly, a map showing the exact location of where the specific animal was reported as found. This is only populated if the latitude and longitude of the location are housed in the database.

[](https://www.youtube.com/embed/R1jenrJ1ESU?feature=oembed)The following video is a brief walkthrough of the tool and its current features:

**Code Preview**

The Animal Rescue Database relies on a simple Python file for CRUD operations. This file is AAC\_CRUD.py.

Class Creation:

A screen shot of a computer

Description automatically generated

This code allows the Jupyter Notebook file to log into the specified database with user credentials.

Create Method:

A screen shot of a computer code

Description automatically generated

This code lets the user add new information to the database. It first tries to add the information; if it cannot, it will show an error message. Currently, this method is not usable in the web-based dashboard but can still be utilized within the Jupyter Notebook.

Read Method:

A computer code with text

Description automatically generated with medium confidence

This code allows the user to search for specific information in the database. The user can pair any specific search terms as a library and use that as the data input. If there is an error, a message will show. Currently, this method does not produce the error message on the dashboard.

Update Method:

A close-up of a computer code

Description automatically generated

This method uses two pieces of information: the original data to find the document and the new data to be replaced. Users should be cautious with their search data to ensure only the correct document updates. Currently, this method is not usable in the web-based dashboard but can still be utilized within the Jupyter Notebook.

Delete Method:

A computer code with black text

Description automatically generated with medium confidence

The delete method takes the specified query and deletes all data that matches it. Users should be careful when using this functionality, as it will delete any results that match the query. Currently, this method is not usable in the web-based dashboard but can still be utilized within the Jupyter Notebook.

The Jupyter Notebook file, AKerr\_Project2\_Dashboard.ipynb, contains most of the visual functionality.

Unique Header:

A close-up of a computer code

Description automatically generated

This simple code includes the Grazioso Salvare logo and the program title in the header. The style code centers the content and places a small space between the image and the text.

Dropdown Menu HTML:

A screenshot of a computer screen

Description automatically generated

This code produces the dropdown menu for quick filtering. The primary label gives the user direction, while the subsequent labels let the user know what the filter should produce. The values listed work with the menu method to pair the dropdown selection with a specific query.

Dropdown Menu Functionality:

A computer screen shot of a code

Description automatically generated

A computer screen shot of a code

Description automatically generated

A computer screen shot of a program

Description automatically generated

This method is called when a user selects a specific filter from the dropdown menu. It begins by setting the data frame to None. The filter type listed is the value from the dropdown menu HTML. The filter type selection sets the query to the appropriate requirements and uses the Read method from the CRUD file. It then sets that read query to the data frame for displaying on the table.

As Grazioso Salvare set the query requirements for the current functionality, the queries are set to pull exact results for the dog breeds. It is possible to change the query to populate any result that includes the preferred breed by using code like the following:



If you use this in place of where the code uses “Labrador Retriever Mix,” the program will filter to include any dog whose breed includes the word “Labrador.” This means that it will populate with results where the breed of the dog may be something like “American Pit Bull/Labrador Retriever.”

Data Table HTML:

A screen shot of a computer program

Description automatically generated

This is the HTML that builds the visuals for the data table. As the code is commented, you will notice that editing the table and deleting rows are disabled. This ensures the user cannot tamper with the data within the program's base functionality.

Additionally, we can highlight the sort mode, which allows for multi-column sorting; the selected rows and columns, which set the default selected; and the page size, which limits the number of rows seen at a time. The developer chose the page size of 10 as an aesthetic choice. Filter action and sort action must remain “native” to enable filtering and sorting.

Graph Display:

A screen shot of a computer code

Description automatically generated

This code builds the graph shown on the web app. It uses Plotly’s px to create a pie chart from the data frame. It pulls the breed data currently displayed on the data table and breaks it into percentages. Several design choices were made based on developer aesthetic preferences and can be adjusted.

**Challenges**

The development team encountered two problems during our process.

The first issue involved the dropdown menu filtering. At first, the development team created variables as dictionaries for the queries. We then used if/else statements matching the values from the menu. The method should use the variable to generate the query if the value matches. This would allow for cleaner code and make changes easier for future developers. Unfortunately, these variables did not pass correctly and would not populate filtered data. This was resolved by directly building the query into the if/else statements.

The second issue is an ongoing issue from version 1.1. At launch, the data table selects row 0 as the information to display on the map. If the user uses one of the dropdown menu filters, the selected row remains row 0. When the user uses the column filter to refine the information on the data table, the selected row does not adjust and remains the original row selected. If this selected row vanishes from the table upon filtering, no row is selected, and subsequently, no data is available for the map.

**Resources**

The following resources were utilized to help in the development of this program:

* Austin Animal Center Outcomes data: <https://doi.org/10.26000/025.000001>
  + This provides information on the UI elements available with Dash.
* Dash Core Components: <https://dash.plotly.com/dash-core-components>
  + This provides a general overview of what is available with Dash and includes the information for creating and managing the dropdown menu.
* Dash DataTable (Dash/Plotly): <https://dash.plotly.com/datatable>
  + This is the overview page of using Dash’s DataTable functionality and includes subsequent pages with additional information for building out your data table.
* How to Make Pie Charts (Plotly): <https://plotly.com/python/pie-charts/>
  + This goes over the basics of creating Pie Charts in Python with Plotly. It also includes details on refining the visuals, adjusting the text size, and even changing the color.
* Setting Graph Size in Python (Plotly): <https://plotly.com/python/setting-graph-size/>
  + This explains adjusting height, width, and margins for graphs made with Plotly.

**Roadmap**

Planned future updates

* Resolve mapping errors
* Adjust data table filtering to allow mixed case and partial match

Version 1.2 (current)

* Added dropdown menu and filters
* Added graphic functionality
* Added Grazioso Salvare Logo

Version 1.1

* Implemented instantiation process
* Implemented filterable data table
* Ensured mapping functional

**Contact**

The primary developer is Amanda Kerr. They can be reached at:

* Email: amanda.kerr1@snhu.edu
* Discord: amarkerr