# Rescue Breeds Dashboard README – Noelle Bishop

## About the Project – Rescue Breeds Dashboard

This project is about creating an interactive dashboard that queries an animal shelter database for breeds relevant to different types of rescue operations, like water rescue. It uses a reusable Python CRUD class to query a MongoDB database with specific filters. Although the IPYNB file in use is tailored to the client’s needs, thanks to the underlying Python code, it can be easily customized for any kind of dashboard.

## Motivation

This project exists to create an intuitive, informative dashboard for quickly finding appropriate shelter animals for different rescue operations. Detailed filtering options, table and chart views, and a geolocation map all serve to deliver details on available animals like their location, breed, and age. This project also works as proof-of-concept for the Python CRUD class it queries the database with.

## Getting Started

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

1. Set up your MongoDB database.
2. Download MongoDB\_CRUD.py.
3. Download MongoDB\_CRUD\_Dashboard.ipynb and place it in an appropriate directory for running with Jupyter Notebook.
   1. Place MongoDB\_CRUD.py in the same directory as MongoDB\_CRUD\_Dashboard.ipynb.
   2. Create a folder titled “assets” in the same IPYNB directory.
   3. Download “GS-logo.png” and place it in the assets folder.
4. Change the credentials in MongoDB\_CRUD\_Dashboard.ipynb to match the credentials of the user you want to test functionality as.
5. Run the IPYNB file and test the dashboard.

**Challenges**

This project built off Project One and previous weeks’ work in a logical way, so its continued development felt straightforward. Figuring out how to include a pie chart and place it side-by-side with the geolocation chart was the biggest challenge, since button pressing logic was handled in last week’s work. I also needed help adding a link to the logo image. I found guidance from various sources (cited in code) and from looking at how other parts of the code were structured as flex displays.

## Installation

To run this project, you will need…

* A working non-relational database that uses MongoDB.
  + [Download link with instructions](https://www.mongodb.com/try/download/community-kubernetes-operator)
* The Python file MongoDB\_CRUD (see Project One README for further instructions)
* An IDE that supports editing .py files, if needed.
  + [Download link (for Visual Studio Code) with instructions](https://code.visualstudio.com/download)
* Jupyter Notebook, for using and testing the code with the database.
  + [Link with installation instructions](https://jupyter.org/install)

**Justifications/Explanations of Tools**

* MongoDB is the base of this project. It is the Model from which the View displays data and the Controller interacts. For large amounts of varied data, such as animals in a shelter, its allowance of unstructured data is helpful for daily operations, especially given that any entries needing attention can be attended to with CRUD operations.
* The CRUD Python class is the “glue” of this project. It is responsible for Creating, Reading, Updating, and Deleting documents in the database without the need for command line operations in the Mongo shell. In other words, it is the Controller because it retrieves the Model data used in the View.
* The IPYNB file is the “client” of the project. It is the View which displays database information based on how the user interacts with the interface. In other words, it’s the client requesting data from the server. The button press functionality is part of the Controller.

## Usage

The IPYNB and Python files work together to create an interactive dashboard and query a database. The screenshots included below only show code from the IPYNB file, as the Python CRUD file has its own detailed README.

### Code Example

The code uses HTML elements, imported from the dash library, to create an interactive dashboard. The main functionality is pressing radio buttons and viewing the subsequent results, and the code for this is included below. A callback makes the buttons respond to clicks based on their IDs while on\_click() uses callback context to determine which button was most recently pressed. Individual button scenarios are handled by if/elif statements and use multi-dict queries with $in, $gte, and $lte operators.

A computer screen shot of text

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

### Tests

The code can be tested by verifying that filtering options display accurate qualifications. The filters for each type of rescue are included in both a table and in code below.

A table with different breeds

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

Based on the output in Figure 2 (see Screenshots), one can verify that clicking the Water Rescue button successfully retrieves dogs who are one of the three breed types, intact females, and between 26 and 156 weeks old.

### Screenshots

A screenshot of a computer

AI-generated content may be incorrect.

Figure – Default view or Reset (All Animals)

A screenshot of a computer

AI-generated content may be incorrect.

Figure - Water Rescue

A screenshot of a computer

AI-generated content may be incorrect.

Figure - Mountain or Wilderness Rescue

A screenshot of a computer

AI-generated content may be incorrect.

Figure - Disaster Rescue or Individual Tracking

**Roadmap/Features**

Additional query filtering may be added so that deceased animals do not turn up in results, at least for Grazioso Salvare’s purposes.

The filtering options may also be optimized for breeds with slightly different classifications in the database. For example, the client asked for Chesapeake Bay Retrievers as a water rescue breed, but the database classifies these animals as “Chesa Bay Retr Mix”. This change was made in code to accommodate the mismatch, and other such changes could be made to ensure the fullest available results are shown.

## Contact

Noelle Bishop