# CS 340 README Template

## Project 2 Dashboard

*This project is a dashboard application built with Dash, a Python framework for building analytical web applications. The dashboard is designed to interact with a MongoDB database and display data in a user-friendly way.*

## Motivation

*The project was conceived to support the noble cause of rescue operations. Well-trained search-and-rescue dogs are invaluable assets in life-threatening situations, and finding the right candidates for training is crucial. Grazioso Salvare's mission, assisted by this software, is to optimize the selection process of rescue dogs, thereby enhancing the efficiency and success rate of rescue missions.*

## Getting Started

1. ***Dash****: A Python framework for building web applications. It is used for creating the interactive dashboard.*
   * *Includes* ***dash****,* ***dash\_html\_components****,* ***dash\_core\_components****,* ***dash\_table****, and* ***jupyter\_dash*** *for Jupyter Notebook integration.*
2. ***Plotly****: An interactive graphing library for Python. It is used for creating the charts in the dashboard.*
3. ***Pandas and NumPy****: Libraries for data manipulation and numerical calculations.*
4. ***Matplotlib****: A plotting library for creating static, interactive, and animated visualizations in Python.*
5. ***Dash Leaflet****: For integrating Leaflet.js maps with Dash applications.*
6. ***MongoDB CRUD Module****: A custom module named* ***mongodb\_crud.py*** *for performing CRUD operations on a MongoDB database. This module is not a standard library and needs to be provided separately.*
7. ***Other Utilities****:*
   * ***os****: Standard library for OS-dependent functionality.*
   * ***base64****: Standard library for encoding binary data to Base64 strings, used here for embedding images in the dashboard.*

## Installation

*To install the necessary libraries, you can use pip, the Python package installer. Open your terminal or command prompt and run the following commands:  
 pip install dash jupyter-dash dash-leaflet plotly pandas numpy matplotlib  
Make sure to place the* ***mongodb\_crud.py*** *file in your working directory or an appropriate location where Python can import it*

## Usage

***Code Overview***

*The code is divided into several sections:*

* ***Imports and setup****: Importing necessary modules and setting up the database connection.*
* ***Dashboard layout****: Defining the layout of the dashboard, including the data table, radio buttons, and charts.*
* ***Callbacks****: Functions that update the components of the dashboard based on user interaction.*

*The dashboard interacts with a MongoDB database through the AnimalShelter class, which is defined in the mongodb\_crud module. The AnimalShelter class provides CRUD (Create, Read, Update, Delete) operations for the database.*

*The data table is populated with data from the MongoDB database. The data can be filtered based on the selection of the radio buttons. The filtered data is also used to update the pie chart and the geolocation chart.*

*The pie chart displays the distribution of preferred animals based on the filtered data. The geolocation chart displays the location of a selected data entry.*

### Code Example

radio\_button\_to\_breeds\_mapping = {

    'OPT1': ['Labrador Retriever Mix', 'Chesapeake Bay Retriever', 'Newfoundland'],

    'OPT2': ['German Shepherd', 'Alaskan Malamute', 'Old English Sheepdog', 'Siberian Husky', 'Rottweiler'],

    'OPT3': ['Doberman Pinscher', 'German Shepherd', 'Golden Retriever', 'Bloodhound', 'Rottweiler'],

}

radio\_button\_to\_age\_upon\_outcome\_mapping = {

    'OPT1': {'$gte': 26, '$lte': 156},  # 26 weeks to 156 weeks

    'OPT2': {'$gte': 26, '$lte': 156},  # 26 weeks to 156 weeks

    'OPT3': {'$gte': 20, '$lte': 300},  # 20 weeks to 300 weeks

}

radio\_button\_to\_sex\_upon\_outcome\_mapping = {

    'OPT1': ['Intact Female'],

    'OPT2': ['Intact Male'],

    'OPT3': ['Intact Male'],

}

@app.callback(

    [Output('datatable-id', 'data'),

     Output('datatable-id', 'columns')],

    [Input('filter-options', 'value')]

)

def update\_dashboard(filter\_type):

    try:

        # Check which button has been pressed and set the query

        if filter\_type == 'reset':

            query = {}  # Reset button pressed, so no filter

        else:

            # Map filter\_type to the preferred breeds

            preferred\_breeds = radio\_button\_to\_breeds\_mapping.get(filter\_type, [])

            age\_range = radio\_button\_to\_age\_upon\_outcome\_mapping.get(filter\_type, {})

            preferred\_sex = radio\_button\_to\_sex\_upon\_outcome\_mapping.get(filter\_type, [])

            query = {

                'breed': {'$in': preferred\_breeds},

                'age\_upon\_outcome\_in\_weeks': age\_range,

                'sex\_upon\_outcome' : {'$in': preferred\_sex}

            }

### Screenshots

*A screenshot of a computer

Description automatically generated.*

## Roadmap/Features (Optional)

***Features***

* *Interactive data table with sorting, filtering, and pagination.*
* *Radio buttons for filtering data based on specific criteria.*
* *Pie chart that displays the distribution of preferred animals.*
* *Geolocation chart that displays the location of a selected data entry.*

## Contact

Eric Buchanan