## About the Project/Project Title

*This project allows for a user to access the Austin animal shelter database and then execute all CRUD functions from a web-based dashboard.*

*The project itself consists of MongoDB for data storage, A python class to facilitate the CRUD functions, and a Plotly Dash module to visualize the dashboard.*

***Tool selection***

*This project utilizes Python to execute all functions. Python has a large set of tools to be able to execute the CRUD functionality as well as execute other database commands.*

*Plotly Dash has been selected to make the visual dashboard. This is a Python framework that allows for a user to create a dashboard without the need for learning HTML, CSS or Javascript – meaning that both the CRUD module can be created using Python and Juypter notebook.*

*To create the project Jupyter notebook has been selected. This is an open-source IDE that can be operated within a web browser. It is easy to use and has little to no learning curve for new users.*

*For database storage this project uses MongoDB. This option was selected over SQL for its flexibility and ease of use. MongoDB also uses a dynamic schema that helps prevent errors during CRUD functions.*

***MongoDB*** *– this is a NoSQL database system that uses JSON-like documents. It offers drivers for all major programming languages as well, which is what we will be displaying in this project.*

***Download****:* [*https://www.mongodb.com/what-is-mongodb*](https://www.mongodb.com/what-is-mongodb)

***Python*** *– this is an object-oriented high level programming language that will later contain the python dash package. It will also be used to execute the CRUD functionality using a module.*

***Download****:* [*https://www.python.org/downloads/*](https://www.python.org/downloads/)

***Jupyter Notebook*** *– an IDE that can be used within a web browser.*

***Download****:* [*https://jupyter.org/*](https://jupyter.org/)

***Plotly Dash*** *– this is a python framework for creating interactive web applications.*

***Download****:* [*https://dash.plotly.com/*](https://dash.plotly.com/)

## Motivation

*My motivation for this project was to take my newly acquired knowledge of MongoDB and be able to translate it into a visual dashboard that is appealing and easy to use. This will give me needed experience with building end user applications that interact with a database using CRUD functions. By the end of this project, I hope to be able to be able to translate my new MongoDB knowledge to other applications to broaden my skill base.*

## Getting Started

***Note:*** *Detailed reproduction steps will be located within the usage and screenshot section, along with examples.*

1. *Start the mongo shell and import csv file (aac\_shelter\_outcomes.csv).*
2. *Create an admin account and a user account for the database. Save both usernames and passwords in a safe location.*
3. *Create a CRUD class for all functionalities within Jupyter Notebook.*
4. *Create a dashboard within Jupyter Notebook.*
5. *Run and open the dashboard within web browser.*

## Installation

*You will need a current version of Python and MongoDB as well as access to Jupyter notebook.*

***MongoDB***

[*https://www.mongodb.com/what-is-mongodb*](https://www.mongodb.com/what-is-mongodb)

***Python***

[*https://www.python.org/downloads/*](https://www.python.org/downloads/)

***Jupyter Notebook***

[*https://jupyter.org/*](https://jupyter.org/)

***Plotly Dash***

[*https://dash.plotly.com/*](https://dash.plotly.com/)

## Project break down

Grazioso Salvare has requested the following components to be implemented into the dashboard.

* Grazioso Salvare Logo with an anchor tag of www.snhu.edu.
* A unique identifier containing the creator’s name.
* A data table that dynamically responds to the following custom filters:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster Rescue or Individual Tracking
  + All
* A geolocation chart and Pie chart that dynamically responds to the above filtering options.

**Logo with click anchor**

A picture containing map

Description automatically generated

**Data table with radio selectors**

**Custom filters**

**Dynamic geolocation map**

**Dynamic pie chart**

The table will dynamically respond to the following filtering options.

Table

Description automatically generated

Water Rescue:

A picture containing map

Description automatically generated

Mountain or Wilderness Rescue:

Chart

Description automatically generated with medium confidence

Disaster Rescue or Individual Tracking

A picture containing graphical user interface

Description automatically generated

*The client also requested that the geolocation chart dynamically updates. This is facilitated by a tool tip popup stating the animals name if it is selected – if one exists.*

A picture containing diagram

Description automatically generated

A picture containing map

Description automatically generated

## Code Breakdown and recreation

*The following steps can be followed to recreate the project*

1. *Install the required software listed under the* ***Installation*** *section.*
2. *As an administrator load the CSV file into MongoDB.*
3. *Create a user account with read/write permissions for the database.*

Text

Description automatically generated

1. *Create CRUD classes and database initializer class within Jupyter notebook:*
   1. *CREATE*Graphical user interface, text, application, email

      Description automatically generated
   2. *READ*Graphical user interface, text, application, email

      Description automatically generated
   3. *UPDATE*Graphical user interface, text, application

      Description automatically generated
   4. *DELETE*

Graphical user interface, text, application, email

Description automatically generated

* 1. *Initializer class*Text

     Description automatically generated

1. *Create dash components and callbacks within Jupyter Notebook:*
   1. *Import CRUD module and assign login credentials.*Text

      Description automatically generated
   2. *Import custom logo and display with anchor.* Text

      Description automatically generated
   3. *Create interactive filters with requested labels*A picture containing text

      Description automatically generated
   4. *Add dashboard layout options and chart widgets.*Text

      Description automatically generated
   5. *Create callback for filtering buttons when selected using* requested paramters. Timeline

      Description automatically generated with medium confidence
   6. *Add callback for updating piechart data.* Graphical user interface, text, application, email

      Description automatically generated
   7. *Add callback for updating geolocation chart.* Text

      Description automatically generated
   8. *Run the dashboard module and confirm functionality.*

**Challenges:**

Most of the challenges regarding this project came from Python. In This language the spacing as well as indents do matter. When debugging Jupyter Notebook will often point to a specific line for a syntax error, however the real issue should be present at the line directly above it. This can cause headaches as often the line it’s pointing towards contains correct syntax. I recommend reviewing basic Python syntax guidelines or using an outside IDE with better debugging options if possible. I also found that this dataset was not clean. The filtering options were very clear, which is what I used, but I do know that some of the dog breeds are spelled differently throughout the database. To ensure that my filters were working I did use the db.collection.find command with the requested filter inside the mongo shell then matched it to my dashboard to ensure it was working properly. For a better experience I recommend adding to the filtering options in the requirements, or using a cleaner set of data.

## Contact

Your name: Michelle Dumas – [michelle.dumas@snhu.edu](mailto:michelle.dumas@snhu.edu)