# CS 340 README Animal Shelter

## Project Two: Animal Shelter

*This application serves as a gateway for users to access and interact with the animal database of CS 340 Austin Animal Center (AAC), offering powerful search and filtering capabilities. Users can employ various filters to search for animals based on criteria provided by Grazioso Salvare, the project requester. The primary goal of the application is to create an informative dashboard that not only facilitates efficient searches but also harnesses the capabilities of geolocation mapping, user engagement, and dynamic charts. By utilizing these features, users can identify and pinpoint animals suitable for search and rescue operations, enhancing the overall effectiveness of the process.*

## Motivation

*The motivation behind developing this program stemmed from the desire to bolster my proficiency in working with databases and effectively manipulating data. Python was selected as the preferred programming language due to its seamless integration with MongoDB. The decision was based on Python's inherent flexibility, ease of scripting, and its compatibility with MongoDB's structure. Furthermore, Python's swift compilation process, facilitated by tools like Jupyter Notebook, contributed to the choice, allowing for efficient development, and testing of the application's functionality.*

## Getting Started

1. ***Import Data into MongoDB:*** *Begin by entering your MongoDB environment and importing the "aac\_shelter\_outcome.csv" file. This CSV file contains the data you need for your application.*
2. ***Index Creation for Data Parsing:*** *It's advisable to establish two types of indexes, a simple one and a more complex version, to effectively retrieve data from within the documents. These indexes will enhance data retrieval efficiency.*
3. ***User Authentication Setup:*** *Establish user authentication by creating two essential accounts - an Admin account and an "aacuser" account. These accounts will provide secure access to your MongoDB database. This step ensures that only authorized users can interact with the data.*
4. ***Python Installation and Program Execution:*** *Ensure you have Python installed on your system. You'll need to run the program from a Jupyter Notebook or similar environment. This will allow you to execute the code and interact with the data using Python.*
5. ***Connecting to the Dash Application:*** *After running the program, you'll need to input the address of the Dash application. This is the web-based interface that allows users to interact with the data visually and intuitively. Connecting to the Dash application opens the user interface for exploring and analyzing the data.*

## Installation

1. ***Python and Jupyter Notebook:*** *You should have a current version of Python installed on your system, along with Jupyter Notebook to execute both* ***.py*** *and. ipynb files.*
2. ***Required Libraries:*** *Ensure that you have the necessary libraries installed. You can use the following commands to install them:*
   1. *A screen shot of a computer

      Description automatically generated*
3. ***MongoDB:*** *You need to have MongoDB installed and running on your system to access the database.*
4. ***Run the Provided Code:*** *Copy and paste the provided code into a Jupyter Notebook or a* ***.py*** *script and run it. The code sets up a Dash dashboard to interact with MongoDB, fetches data, and displays it using various components.*
5. ***Access to Required Files:*** *Make sure you have access to the necessary files, including the image 'Grazioso.png' mentioned in the code. Ensure that the image file is in the same directory as the code.*
6. ***Configure Database Access:*** *Replace the* ***username*** *and* ***password*** *values with your actual MongoDB credentials to connect to the database.*

## Usage

*When clicking the radio filters the histogram also updates   
A screenshot of a computer

Description automatically generated*

*A screenshot of a data

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

### Code Example

Running Dash: with the unique identifier  
A screenshot of a computer

Description automatically generated

Working Radio buttons: When clicking each one it filters the breeds and age of the animals that’s best suited for the need. Additionally, this updates the histogram as well

A screenshot of a computer

Description automatically generated

Changing options and switches Geolocation:  
A screenshot of a computer

Description automatically generated

**Errors:**

## The error I experienced, specifically the "InvalidComponentIdError," suggests that there was a problem with the component IDs I specified in my callback functions. Dash component IDs are used to identify which components trigger or receive updates from a callback. The error message indicates that one or more component IDs contained characters that are not allowed, such as dots (.) or curly braces ({}), in the context of the callback. Component IDs should follow specific naming conventions to ensure they are valid for JSON serialization and deserialization.

## To resolve this error, I needed to ensure that the component IDs specified in my callback functions adhere to the guidelines provided by Dash and JSON. I should check the callback definitions, the components involved, and their IDs to make sure they are correctly formatted and do not contain any disallowed characters. This will help my callbacks communicate data properly between the front-end and back-end of my Dash application.

## Contact

Your name: Blake Kemp.