# CS 340 README

## About the Project/Project Title

This project focuses on creating software for an international rescue-animal training company, Grazioso Salvare. As lead developer on this project, I must help build an application that works with existing data from the animal shelters to identify and categorize dogs within the shelters they partner with. This includes building a database that will have CRUD functionality, currently with Create and Read functionalities, as well as building a web application that can interactively filter through it. Overall, this project helps to display the functionalities and understanding of CRUD, Dash, and its interaction with Jupyter Notebook, MongoDB, and Python.

## Motivation

This project exists to not only create an adequate software application for Grazioso Salvare, but to fully display the capabilities as a developer my knowledge on CRUD and understanding how to implement a full-stack application. This project focuses on working with existing data from animal shelters and utilizing a database and client-facing web application dashboard to display this information. MongoDB, a database program, and Dash, a low-code framework, was utilized in this project as well to understand its capabilities and functionalities alongside creating Python code.

## Getting Started

To get a local copy up and running on your system from GitHub, follow these steps:

1. Ensure you have all the tools needed downloaded onto your system, which are listed below.
2. Open your terminal shell on your computer. This will vary based on what operating system you have.
3. Ensure you have MongoDB installed, and open this within your shell.
4. Once MongoDB is opened, import the Austin Animal Center CSV file with the import tool required. Using mongoimport is important here, and any additional information regarding the correct commands to use can be found here: <https://www.mongodb.com/docs/database-tools/mongoimport/>
   1. Ensure to use the database name “AAC” and collection name “animals” when importing the file.
5. Create a simple and compound index to help with queries to look for breeds. Additional help with this can be found here: <https://www.mongodb.com/docs/manual/indexes/>
6. Now to user authentication. This part is particularly important for security measures and accounts set in place for both administrators and users.
   1. Create an administrator account via the mongo shell. Help with this can be found here: <https://www.mongodb.com/docs/v4.2/tutorial/enable-authentication/>
   2. Enable user authentication for the database by using the following commands:
      1. sudo /bin/cp /etc/mongod\_withauth.conf /etc/mongod.conf
      2. sudo /bin/systemctl restart mongod.service
   3. To ensure that the user authentication has been enabled, use this command in Linux to start mongo:
      1. Mongo --port xxxxx --authenticationDatabase “admin” -u “admin” -p
      2. This will prompt you to give your username and password.
   4. Create a user account to database AAC inside the mongo shell. Use the mongodb docs for additional help: <https://www.mongodb.com/docs/v4.2/tutorial/enable-authentication/>
7. From here, you will now create the C, R, U, and D portions for the Python module, which are the Create, Read, Update, and delete sections for the CRUD module.
   1. First, you must import MongoClient and ObjectId.
   2. Then, create a class called “AnimalShelter”
   3. From here, you need to initialize the MongoClient to access the database and collection.
   4. Then comes the creating the CRUD portions.
      1. Create a function for Create that takes in the parameters self and data
      2. Then create an if-else statement that if data is not none, then an error occurred.
      3. The Read portion is created similarly, however, you must keep in mind utilizing the find() method over the insert() method for create.
      4. The Update portion is again, similar to Create and Read, however, you must include ‘newValues’ in the parameters to take in the new updates made to the data created in the Create method.
      5. Lastly, the delete method is straightforward and must delete whatever data chosen.
8. Now, we move onto making the web application dashboard with Python. This dashboard allows users to navigate the data provided more easily.
   1. You must create a data table using Dash to enable an unfiltered view of the Austin Animal Center Outcomes data set. The CRUD Python module will come in handy here.
   2. Next you will have to implement filter functionalities for the four queries required: water rescue, mountain rescue, disaster rescue, and finally resetting back to the original data.
      1. This will also include creating either radio buttons or a drop-down menu to filter through these options via the application’s dashboard.
   3. You must modify the data table created so that once a choice is “pressed” or filtered, the data table will respond in turn by displaying data from that particular query.
   4. Next, you must create a geolocation chart and a second chart, such as a pie chart or graph. More assistance with this can be found here: <https://plotly.com/python/pie-charts/> and here: <https://plotly.com/python/maps/>
      1. Both the map and chart of your choice must be able to filter through the AAC data as well when a specific button is pressed.
   5. Do not forget to include and import the image (logo) needed for the dashboard. To properly import this image, be sure to include ‘import base64’ to the beginning of your jupyter project to ensure it properly loads.

## Installation

The tools needed to use the software application are as follows:

* **Terminal**: This will allow you to test and see the database on your computer. Your computer should already have a terminal installed.
* **Jupyter Notebooks**: This application is necessary to run the IPYNB files provided in the project. You can download Jupyter Notebooks here: <https://jupyter.org/install>
* **MongoDB**: This software is necessary to run the script within the terminal, and is important to understand how a NoSQL database works. It is best to download the Community Server version, since it is free to use.

Download MongoDB here: <https://www.mongodb.com/try/download/community>

* **Python**: This is the main programming language required for the Python files included in the project. Download the latest Python version for your operating system here: <https://www.python.org/downloads/>
* **Dash:** Dash is a low-code framework to help developers build data apps, including data tables, charts, and maps within their applications. More information on how dash works, and its implementations can be found here: <https://plotly.com/dash/>
* **Additional Resources:**
  + This resource from Medium helped break down how to interact with Python data table using a pie chart: <https://medium.com/analytics-vidhya/a-guide-to-interactive-data-visualizations-with-python-plotly-c1b949eeb1f>
  + This resource helped with understanding any design changes with color to implement: <https://plotly.com/python/discrete-color/>
  + Lastly, a resource from GeeksforGeeks helped to break down the difference between loc and iloc for the geolocation function and how to implement it: <https://www.geeksforgeeks.org/difference-between-loc-and-iloc-in-pandas-dataframe/>

## Usage

In this section I have included code examples, tests, and screenshots of the processes of how the code should look and run.

### Code Examples

### CRUD Module

*# Complete this create method to implement the C in CRUD.*

*def create(self, data):*

*if data is not None:*

*self.database.animals.insert\_one(data) # data should be dictionary*

*return True*

*else:*

*raise Exception("Nothing to save, because data parameter is empty")*

**Web Application Dashboard**

*app.layout = html.Div([*

*# Included unique identifier*

*html.Center(html.B(html.H1('Rosario Robinson CS-340 Dashboard'))),*

*html.Hr(),*

*# added customer logo and resized it*

*html.Img(id='customer-image',src='data:image/png;base64,{}'.format(encoded\_image.decode()),*

*style={'width': '50%',*

*'height': '50%'},*

*alt='customer image'),*

### Tests

The tests done for this code are in a test script file within Jupyter Notebook and require a username and password, as well as data to insert into the database itself. I have included an example of my code to display how the test will be run. The test script creates a new object or data to be insert that you will create, then will go through the process of the CRUD python file previously created. Ensuring that the data created will properly be read, updated, and deleted within the AAC database and the animals collection. I have also included code from the web application dashboard that include creating functionality for radio buttons.

***CRUD module***

*from animal\_shelter\_CRUD import AnimalShelter*

*obj1 = AnimalShelter("aacuser", "mango")*

*data = { "name": "Jax", "breed": "German Shepherd" }*

*if obj1.create(data):*

*print("record added")*

*else:*

*print("error adding record")*

*record = obj1.read({"name" : "Jax"})*

*for k in record:*

*print(k)*

***Button functionality***

*# button function to click through and display information required*

*def on\_click(bt1,bt2,bt3,bt4):*

*# start case*

*# button one displays for Water Rescues*

*if (bt1):*

*df = pd.DataFrame.from\_records(shelter.read({"animal\_type":"Dog","breed":*

*{"$in":["Labrador Retriever Mix","Chesapeake Bay Retriever","Newfoundland"]},*

*"sex\_upon\_outcome":"Intact Female",*

*"age\_upon\_outcome\_in\_weeks":{"$gte":26},*

*"age\_upon\_outcome\_in\_weeks":{"$lte":156}}))*

### Screenshots

The screenshots provided below give a brief overview of how each section of the project was executed, from importing and preparing MongoDB, to creating and running the CRUD python module, to implementing and executing the web application dashboard.

**Inserting CSV file via MongoDB import tool**

Graphical user interface, text

Description automatically generated

**Ensure authentication to the database and collection**

**Admin**

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

**AAC User**

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

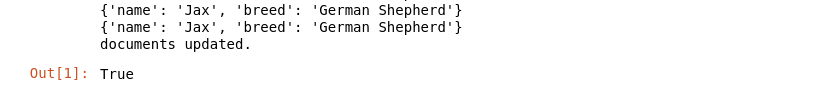
**CRUD Execution**

Graphical user interface, text, application, email

Description automatically generated

A picture containing text, newspaper, screenshot

Description automatically generated



**Python CRUD file**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Dashboard First Ran**

A picture containing text

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Chart

Description automatically generated

Map

Description automatically generated

**Executions of dashboard**

**Water Rescue**

A picture containing text

Description automatically generated

Table

Description automatically generated

A picture containing chart

Description automatically generated

Map

Description automatically generated

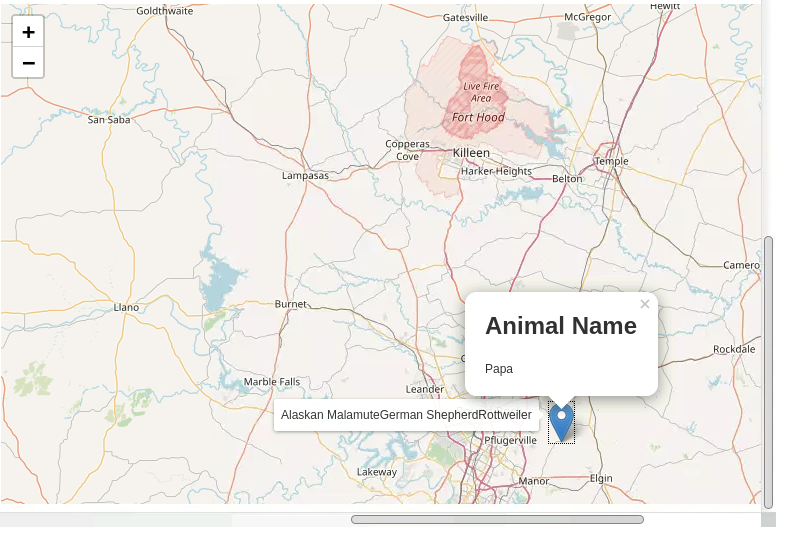
**Mountain Rescue**

Text

Description automatically generated

Chart, pie chart

Description automatically generated



**Disaster Rescue**

Text

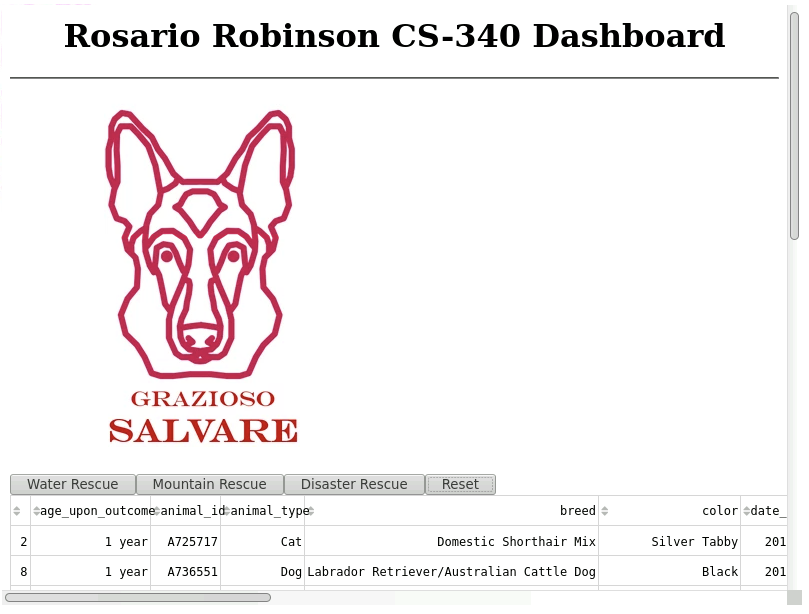
Description automatically generated

A picture containing chart

Description automatically generatedMap

Description automatically generated

**Reset**



Chart, pie chart

Description automatically generated

Map

Description automatically generated

## Contact

Your name: Rosario Robinson