# CS 340 README

## About the Project/Project Title

This project serves to assist the client, Grazioso Salvare, more easily organize and analyze the animals stored in a MongoDB database with the purpose of finding animals that may be suitable for rescue tasks and maintain a record of all animals kept by the organization. In this iteration, the application produces a graphical user interface displaying a chart that can organize entries stored in the animal database, here referred to as AAC, into categories such as age, breed, animal type, or location.

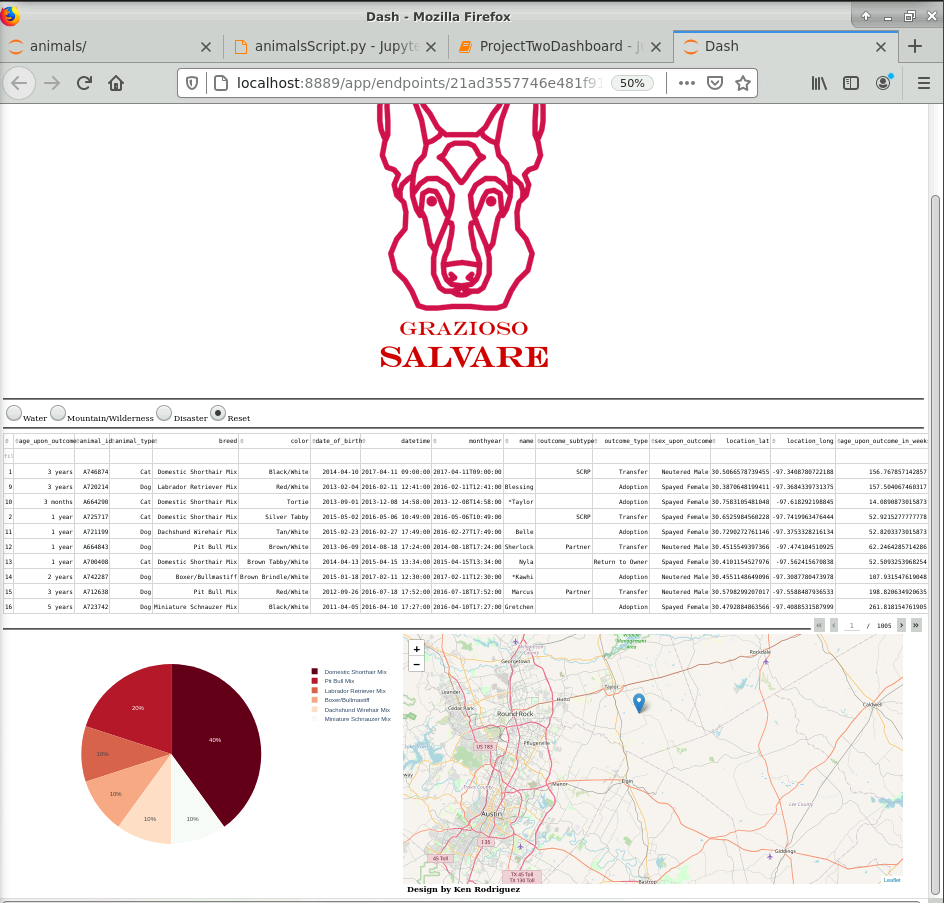
To interact with the database’s contents, radio buttons displayed near the top of the page execute different MongoDB *read* queries, organizing the animals displayed into different categories such as Water Rescue animal candidates or Disaster Rescue animal candidates, then displaying them in the database chart and producing a pie chart and geolocation result from the query’s results. This provides the user with easy to read, easy to navigate, and actionable data.

## Motivation

This project was created with the intention of studying CRUD operations in a MongoDB database in a practical context. For example, the read operation is utilized to provide the application’s users with organization methods to navigate the database, organize the database, and a couple of charts to further navigate the data stored in the form of a pie graph and geolocation. This exercise has been an informative and extensive look at how a complete application may function and look on a low level.

**Client Requirements**

* Dashboard Branding
  + The Grazioso Salvare logo must be displayed at the top of the page
  + The author must use a unique identifier (*Design by Ken Rodriguez*)
* Dashboard Widgets
  + Interact filter options (radio buttons) to filter the Austin Animal Center Outcomes dataset by:
    - Water rescue
    - Mountain/Wilderness rescue
    - Disaster rescue
    - Reset
  + A data table that dynamically responds to filtering
  + A geolocation chart AND a second chart (pie chart) that dynamically respond to filtering options



*The client’s requirements are displayed in this image: the Grazioso Salvare logo at the top of the page and the “Design by” unique identifier at the bottom of the page for branding requirements, then a datatable, pie chart, and geolocation chart that can be interfaced with using organizational radio buttons at the top of the chart. More examples of this interaction are available later in this document.*

## Getting Started

*This is an example of how you may give instructions on setting up your project locally: “To get a local copy up and running, follow these simple example steps.”*

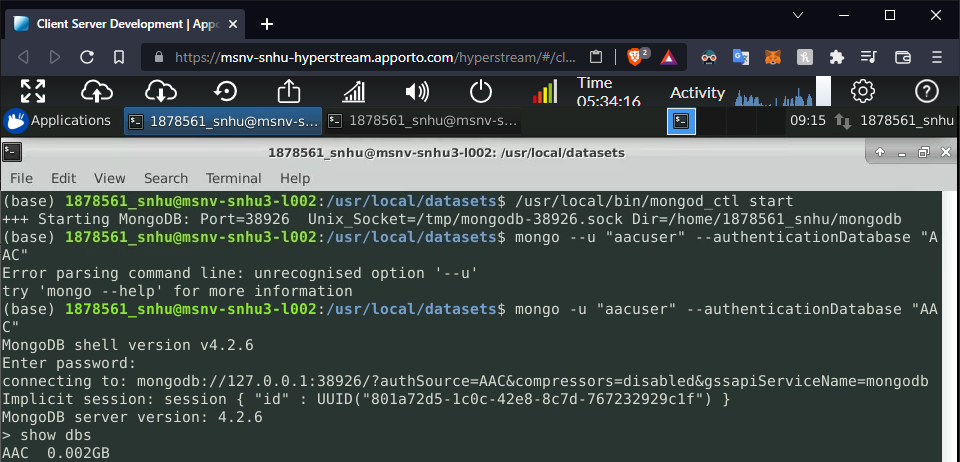
To use the tools included with this project most effectively, first install MongoDB and an IDE that can execute Python scripts; the included scripts were developed and work well with Jupyter Notebook.

1. First, create a database using MongoDB named *AAC*.
2. Create an administrator account and user account to access that database. For this instance, we will assume this account is called *aacuser*.
   1. The user account should have read/write permissions.

Text

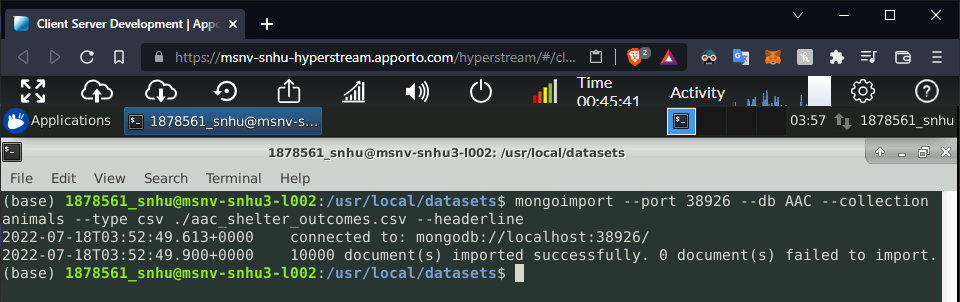
Description automatically generated

*Here, the command* db.createUser *is used to create a new user, “myUserAdmin”. This is an administrative account; after specifying that a password should be created, the relevant roles “userAdminAnyDatabase” in the “admin” database and “readWriteAnyDatabase” in the “admin” username establish these permissions. Afterwards, the user is prompted to type in a secret password, and the account is successfully created. A user account could be added similarly, using the “readWriteAnyDatabase” permissions in the “AAC” database.*

**

*Here,* mongo *is executed to launch MongoDB database with the* -u *parameter (user login) with the username “aacuser” using the* –authenticationDatabase *authentication database* “AAC”*. The* show dbs *command is then executed to verify that the user aacuser can only access the* AAC *database.*

1. Import the database data from the document *aac\_shelter\_outcomes.csv* using the *mongoimport* command.
   1. First, use your instance’s port number with the *–port* field; here the port number is 38926, yours will very likely be different.
   2. Next, specify the database destination with the *–db* field, *AAC*.
   3. Choose the *animals* collection using *–collection*.
   4. Specify the filetype with *–type csv*, then specify the CSV file’s location; here, *./aac\_shelter\_outcomes.csv* is used.
   5. The .csv file has a headerline to define different fields in database entries, so the *–headers* modifier is used to skip the first line and ensure the data is properly implemented.



*Example of the mongoimport command being used. The instance’s port number is first used, then the database to be used is specified. Next, the collection to which the database entries will be added is specified as animals followed by the filetype, csv. Finally, the file name is added followed by the headerline qualifier to ensure the first line is skipped.*

1. Add the included Python script and test script to a Jupyter Notebook note.
2. Launch MongoDB in the terminal if it’s not already running.
3. Modify the included Jupyter Notebook test script in the section under *Data Manipulation / Model* to match the MongoDB instance’s login; in this case, the username is “aacuser” and the password is “password”.
4. Run the Jupyter Notebook test script. If everything runs successfully, the charts should launch in the default “Reset” state and allow organization using the radio buttons or header columns.

## Installation

*List the tools you need to use the software and how to install them.*

The tools necessary to use the included scripts are MongoDB, Python, and Jupyter Notebooks.

* MongoDB is a noSQL database server software that comes in community and enterprise editions, detailed instructions can be found on the MongoDB website:
  + <https://www.mongodb.com/docs/manual/installation/>
* Python is a scripting language with several powerful libraries for a multitude of purposes; for this purpose, it is used to access and modify the AAC database with MongoDB. Windows executables, OSX packages, and Linux tarballs can be installed on the Python website:
  + <https://www.python.org/downloads/>
* Jupyter Notebook can be installed with Python pip commands in the commandline. Installation instructions can be found on the Jupyter website under *Jupyter Notebook*:
  + <https://jupyter.org/install>

## Usage

In its current state, the application offers organization methods using the radio buttons at the top of the database contents chart, right below the Grazioso Salvare logo. This offers four options, allowing users to primarily sort animals into the categories of Water Rescue candidates, Mountain/Wilderness Rescue candidates, Disaster Rescue candidates, and reset to the default criteria of all animals in the database. The data can be further organized by clicking the header titles, modifying the data displayed in the included pie charts.

* 1. *Default Reset*

A picture containing graphical user interface

Description automatically generated

* 1. *Water radio button*

A picture containing graphical user interface

Description automatically generated

* 1. *Mountain/Wilderness radio button*

Graphical user interface, application

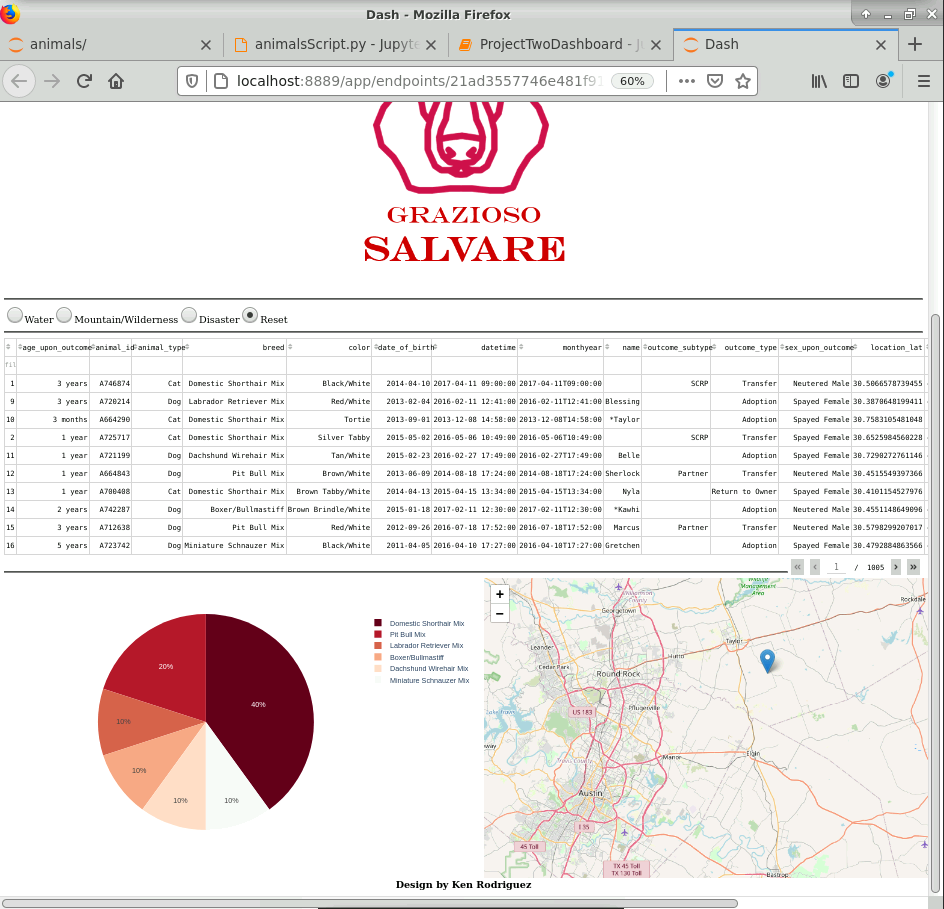
Description automatically generated

* 1. *Disaster radio button*

Graphical user interface

Description automatically generated

* 1. *Reset radio button*



### Creation

To create this application, after creating and importing the MongoDB database’s contents, a Python script was written to allow applications to interface with that database using CRUD (Create, Read, Update, and Delete) commands. After the script was written, a Python graphical user interface was written to meet requirements, namely the brand logo and the design credit. Then, the first Python script’s Read CRUD command was used to create a datatable. Entries from that datatable are used to create a Dash framework pie chart displaying the distribution of animal breeds and a Dash framework geolocation map displaying animal locations by their respective latitude and longitude.

### Challenges

During development, one of the most notable challenges I had faced was implementing the Dash framework items in the context of the application; for example, writing queries in Mongo that would work with the radio buttons to change the data displayed in the pie chart. Initially, the script had been using the Read command I had written instead of the ReadAll command, producing incomplete results or results that did not produce any results at all. After unsuccessfully playing with the datatable and charts and realizing the primary issue was implementing the wrong read command, the project was a matter of building on earlier iterations of my application.

## Contact

Your name: Ken Rodriguez