# CS 340 Grazioso Salvare README

Jessica Ayer

Date: October 16th, 2022

CS340-T1192 22EW1

Instructor Dr. Tad Kellogg

## A picture containing text Description automatically generated

## Author:

Jessica Ayer

Jessica.ayer@snhu.edu

## About the Project/Project Title

Grazioso Salvare: Austin Animal Center Dashboard

## Grazioso Salvare, a rescue animal training company has partnered with Austin Animal center to create a database of animals available for adoption. This application utilizes CRUD functionality and is secured with user authentication. It allows Grazioso Salvare to filter the animals available for adoption that meet their criteria for different types of rescue animals. It utilizes Read queries to a MongoDB database to filter for the desired results. The base language Python and the user interface was created with Dash, plotly and leaflet. An interactive dashboard allows the user to further filter the results, sort them by any field, or remove animals from the list. A geolocation map shows a marker for the first animal listed, and a pi-chart gives a visual representation of the varying breeds listed on the page.

## Motivation

## This application is built for the use of Grazioso Salvare to Identify candidates for search and rescue dogs in the Austin Area animal shelters. It maintains a database of available animals and has user friendly controls to quickly access animals that fit Grazioso Salvare’s various rescue animal requirements. This includes water rescue, mountain or wilderness rescue, and disaster or individual tracking dogs. Click the logo in the application to link to Grazioso Salvare’s website if you would like to learn more about their work.

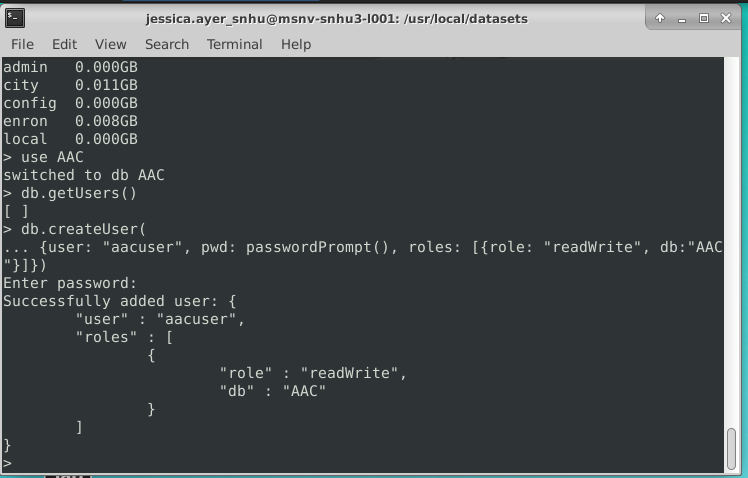
Graphical user interface, application

Description automatically generated

## Getting Started

This application will be open source and made available on GitHub. The link will be made available after completion. MongoDB any Python must be installed onto the user’s device. See installation for links to needed tools. Once appropriate tools are installed follow these easy steps.

1. Create a database. MongoDB database was used for this project because it interacts well with Python (the most used language and language of choice for this project) through the PyMongo library. MongoDB’s schemeless capabilities is particularly helpful for this application. Often data on stray animals is incomplete. With MongoDB there is the flexibility to work around missing data with ease. [Create A MongoDB Database | MongoDB](https://www.mongodb.com/basics/create-database)
2. Create user accounts. For MongoDB see the following resource [Enable Access Control — MongoDB Manual](https://www.mongodb.com/docs/v4.2/tutorial/enable-authentication/)



1. Go to GitHub for source code [JessAyer/CS340 (github.com)](https://github.com/JessAyer/CS340)
2. Download the animal\_shelter.py to your local device.
3. Download ProjectTwoDashboard.ipynb to your local device.
4. Start MongoDB with authentication.
5. Run ProjectTwoDashboard.ipynb in Python IDE (Jupyter Notebook was used to create this application)

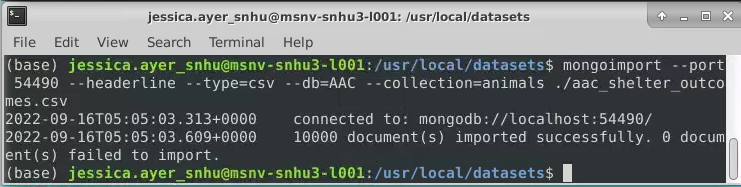
## Installation

1. MongoDB server available at [MongoDB: The Developer Data Platform | MongoDB | MongoDB](https://www.mongodb.com/)
2. Python available at [Welcome to Python.org](https://www.python.org/)
3. Jupyter Notebook available at [Project Jupyter | Home](https://jupyter.org/)

To use this a pre-existing data set you can import a CVS file into MongoDb via the mongo shell with the following command. (Be sure to change the port number to the port that MongoDb is utilizing on your device and the file name to your CVS file.) A sample database CVS file is available in the GitHub repository. [CS340/aac\_shelter\_outcomes.csv at main · JessAyer/CS340 (github.com)](https://github.com/JessAyer/CS340/blob/main/aac_shelter_outcomes.csv)

mongoimport –port= ##### --db= AAC –file=“file name here”.cvs

See images bellow for example:



## To Reproduce

if you would rather reproduce this application than use the provided files additional tools will be needed. Feel free to brose the provided py and ipynb files for guidance and utilize the following resources to make your own customizations.

1. See above for creating Database.
2. Create a CRUD module with python. See [CS340/animal\_shelter.py at main · JessAyer/CS340 (github.com)](https://github.com/JessAyer/CS340/blob/main/animal_shelter.py) for an example. Remember to import the database library.



1. Dash Framework for Python was used to create the user interface. See the [CS340/ProjectTwoDashboard.ipynb at main · JessAyer/CS340 (github.com)](https://github.com/JessAyer/CS340/blob/main/ProjectTwoDashboard.ipynb) for an example. This project utilizes the callbacks, Data Table, Radio Items, and Pie Chart but there are tutorials available for you to make your own customizations including alternative buttons/dropdowns and visual charts. [Introduction | Dash for Python Documentation | Plotly](https://dash.plotly.com/introduction)
2. Dash Leaflet was utilized for the geolocation map. For instructions visit [Dash (dash-leaflet.herokuapp.com)](https://dash-leaflet.herokuapp.com/)

Text

Description automatically generated

## Usage

This application uses MongoDB for server-side backend programing. The Python language is used to make database queries and Dash is used for the user interface. For a brief overview on how this application works there are examples of the code bellow and a description of their functionality.

### Code Example

This application is set to default with user authentication. For the application to work you must me logged into MongoDB with an account that has appropriate access. To turn start MongoDB with authentication, see the images bellow.

Text

Description automatically generated

The application utilizes a read\_all query from the CRUD module to filter search results:

Text

Description automatically generated

To add addition filter options, add a new radio button and search query:

Text

Description automatically generated

Text

Description automatically generated

**Challenges**

This was my first experience working with databases and building a full-stack application, so each step presented new challenges. Developing the callback for the radio buttons interactivity was by far the most challenging part for me. I honestly didn’t believe I would meet the deadline for this project as I struggled with the radio buttons up until the eve before the project was due. The tutorial showed several ways to build the radio buttons, however I could only get one format to work. Getting them to appear on the dashboard was only part of the challenge though. Building the callbacks and configuring them to each of the graph’s nearly caused panic a few times when changes made the graphs completely disappear. Eventually in a moment that felt like magic, after studying the resources and provided callbacks in the templates, and a lot of manipulating lines of code to see what would happen, I found the error and was able to get the desired functionality.

### Tests/Screenshots

Starting state:

Application

Description automatically generated with medium confidence

Water Rescue:

A picture containing graphical user interface

Description automatically generated

Mountain of Wilderness Rescue:

A picture containing application

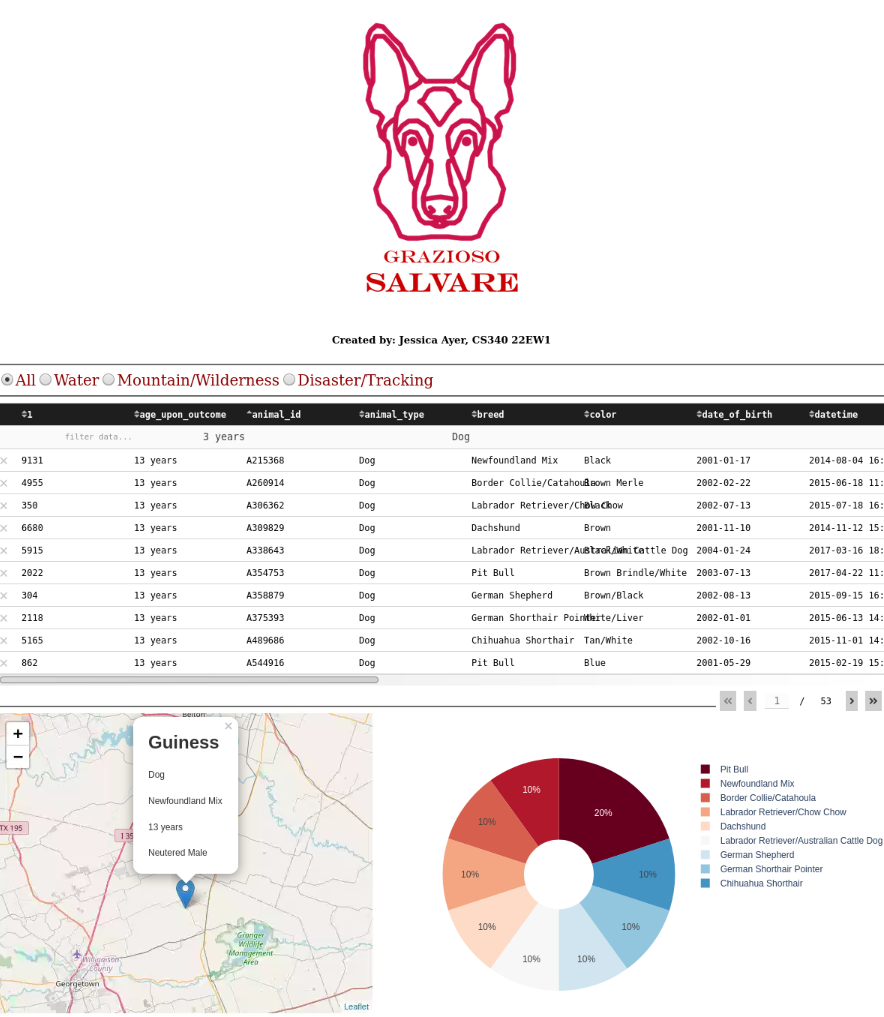
Description automatically generated

Disaster or Individual Tracking:

A picture containing application

Description automatically generated

Custom Filters (Dog, 3 years, sorted by ID):



Reset (All with filter ordered by date of birth):

A picture containing graphical user interface

Description automatically generated