# #README GRAZIOSO SOFTWARE AAC DATABASE

## About the Project/Project Title

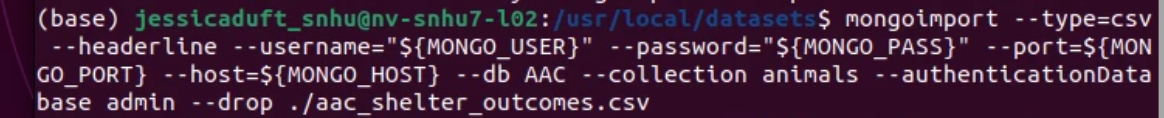
This application allows a user to access the AAC database, which is a database of animals in the AAC shelter. Users can create, read, update, and delete animals from the database as needed using an intuitive dashboard created with the DASH framework.

## Motivation

*This program was designed so that users can better manage the animals within their shelter so that all parties get the best possible outcomes possible. I’ve implanted a CRUD method that can be used with an intuitive and user-friendly dashboard to keep track of the data base as the user pleases.*

## Getting Started

*To get started, run the mongoimport command in a Linux terminal to import the aac\_shelter\_outcomes.csv file.*



*A screenshot of a computer

Description automatically generatedThen run the application using JuPyther, click the dashboard link at the bottom of the page:*

*This brings you to the main dashboard, which is intuitive and user friendly. See more about the dashboard and it’s functionality later in this #README file.*

## Installation

*You need to have a Linux operating system installed, Python should be downloaded and usable along with some shell such as JuPYter, MongoDB shall be installed in order to access the database. The Python driver for MONGODB, PyMONGO, is also used for this project. PyMongo is a user friendly, easy to use, and intuitive way to access and manipulate databases and collections. The AAC database will need constant manipulation, making PyMongo an excellent tool for meeting the program requirements. Finally all of this is brought together to create a final product using the DASH framework.*

## Usage

*User logs in with authenticated user name and password. The user profile created here, aacuser, has full read/write abilities and can only be used by an authorized user who has proper credentials.Once a user is authenticated, they can access the database and make changes to the database by using the code in the next section of this document or by using the intuitive dashboard documented later on.*

A screenshot of a computer screen

Description automatically generated

### Code Example

*The code allows users to create and read data within the database.*

*An example of the code is the following create method, which allows a user to create new data in the database. If item is successfully create a Boolean value will be returned, otherwise an error will be thrown.*

 def create(self, data):

        if data is not None:

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

            if insert!=0:

                return True

            else:

                return False

        else:

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

### Tests

*To test the function above, you would simply call the create method into a class that correlates to the database you are working with. You would then call the read method to view the object that you created or added into your database.*

*There is a full CRUD program implemented into this database so not only can users create new objects, they can also read , update and delete objects. To do so you would simply call the read, update or delete function followed by the read function to confirm changes. Examples of each are shown in the screenshots below.*

### Screenshots

Calls the create and read method

*A screenshot of a computer code

Description automatically generated*

## Calls the update and read method- Below you can see where the animal\_id was successfully changed from TEST123 to TEST\_UPDATE\_123

A screen shot of a computer program

Description automatically generated

## Here we call the delete function followed by the read function to confirm that our test animal with animal\_id TEST\_UPDATE\_123 is no longer in the database

A screenshot of a computer program

Description automatically generated**Functionality:**

## In addition to creating and testing the CRUD methods above , I’ve created a dashboard using the DASH framework that meets all of the client’s specified requirements. The client’s requirements and proof of functionality are listed below :

* *The client wants a company logo displayed on the dashboard with an anchor tag to* [*www.snhu.edu*](http://www.snhu.edu) *, see also the required unique identifier in the below screen shot.*

A screenshot of a computer

Description automatically generated

## *Interactive filter options (buttons, drop-downs) to filter the Austin Animal Center Outcomes data set by: Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and a RESET filter option. See screenshot below of the selectable filter buttons that appropriately filters the data selected by the user: In this screenshot, I’ve pre-selected the Disaster Rescue or Individual Tracking filter option*

A screenshot of a computer

Description automatically generated

## *A Data table which dynamically responds to filter options. See the screenshot below of the data table I created which responds to the radio button filter options shown at the top of the chart. The data can be filtered by breed, color, date of birth, and many other options. In the screenshot below, I have the data from above filtered by color in the screenshot below:*

## A screenshot of a computer Description automatically generated

## *A geolocation graph that shows the filtered data. In the screenshot below I’ve selected Zeke, one of the Disaster or Individual Tracking dog from the screenshot above. The geolocation chart shows the user where Zeke is currently located using latitude and longitude values entered by the user upon creation of the animal in the database.*

A screenshot of a computer

Description automatically generated

* *A second chart, of the developer’s choice, which displays filtered data. I chose to use a histogram graph. The graph will display the data that is currently shown in the data table. Using the same data from above, the Disaster or Individual Tracking filter option, see how the histogram sorts the 4 dogs from our chosen filter by breed:*

A screenshot of a computer screen

Description automatically generated

**Challenges :**

*When creating this dashboard, my first challenge was finding the appropriate port numbers. Port numbers can be notoriously difficult, but after troubleshooting the issue I was finally able to find the correct port number so that the DASH application would actually run. My next challenge was that PLOTLY was not installed in my lab environment. After some minor research I was able to get PLOTLY installed so that my data table, charts, and graphs would load appropriate data. My final challenges were mostly just minor syntax errors, and moving things around so that they looked good on the dashboard. These challenges are expected when developing any program or application.*

## Contact

Your name: Jessica Duft – jessica.duft@snhu.edu