# CS 340 README Template

## About the Project/Project Title

An international rescue-animal training company, Grazioso Salvare, is seeking software to help identify dogs in the Austin, Texas-area animal shelters that would be good candidates for training. The project is to create a CRUD Python module to interact with a .csv file loaded into MongoDB of existing shelter information to be able to search and select candidates based on particular qualities of an animal.

## Motivation

Grazioso Salvare is searching for shelter animals that would be fit for rescue-mission training to help aid in the rescue of people or animals in the wilderness. The client has identified qualities within dogs that would make them more apt for water, mountain, or wilderness rescue, as well as qualities that would best locate humans after disasters or a particular human by scent. This project will make it easier for the client to obtain these animals for training, cut down on the shelter population, and generate a larger population of highly trained rescue animals to save lives.

## Getting Started

## Installation

To get a local copy up and running you must have installed MongoDB, Python, and download the aac\_shelter\_outcomes.csv to import into MongoDB to create the database with the proper user authentication with a username and password to read or write data. You must then download animal\_shelter.py to perform CRUD functions within MongoDB through Python.

* Importing database to MongoDB

Text

Description automatically generated

## Usage

### Code Example – animal\_shelter.py

* The below code is displaying the CRUD functions of the program. Below you can see the Python code connections with MongoDB with proper authentication creating an AnimalShelter class. Once the MongoClient is initialized we can then perform CRUD operations.

Text

Description automatically generated

* Graphical user interface, text, application, email

  Description automatically generatedThe create method reads data to be inserted into the database, checking to see if there is data that meets criteria, if so it’s added to the database, if not, an exception is raised returning false.

Graphical user interface, text, application

Description automatically generated

* The read method passes a search query, if a match to the query is found in the database the result is returned, if not an exception is raised as there is no match.

Graphical user interface, text, application

Description automatically generated

* The update method passes a search query and a variable containing the newValues to be updated. It then checks that both query and newValues have contain a value. If both query and newValues have data and a match is found to the query, the data corresponding to the query is then updated by newValues. If query or newValues is empty, an exception is thrown as there would be no data to search and/or have information to update.

Graphical user interface, text, application

Description automatically generated

* The delete method passes a search query to find a match within the database. If the search query is not empty and a match is found, the corresponding data is deleted from the database. If delQuery is empty, an exception is thrown that there is no parameter to search by.

### Tests

The below demonstrates test to be run in animal\_shelter.py:

* In order for all tests to be facilitated, an instance of the AnimalShelter class with proper authentication must be created defined as testObject.



* To test the create method a data set is created with the qualities defined by the contents of the .csv file that is then passed to the create method, returning “True” if passed.

Text

Description automatically generatedText

Description automatically generated

* To test the read method, a search query is passed by animal ID that is then passed to the read method, resulting in a returned cursor if passed.



Text

Description automatically generated

* To test the update method, a search query as well as a value to update is passed. In this instance, we will be searching by animal\_id and updating the animals name, that is then passed to the update method, returning a result in JSON format if passed.

A picture containing chart

Description automatically generated

A picture containing text

Description automatically generated

* To test the delete method, a search query is passed by animal\_id that is passed to the delete method, returning a result in JSON format if passed.

Text

Description automatically generated

### Screenshots

Below are the results of running the tests in test-script.ipynb

Graphical user interface, text, application, email

Description automatically generated

**Program Dashboard**

* As stated, the above code handles all CRUD functionality within MongoDB. To make this more usable, we developed a web application dashboard to sort through the different qualities that would make Grazioso Salvare choose a dog for training. The Dash app defaults to a menu displaying all animals within the .csv file generated. Below the Grazioso Salvare logo is a dropdown menu featuring the options “Water,” “Mountain or Wilderness,” “Disaster or Individual Tracking,” and “Reset.” Based on the option the user selects, the function will pass the predetermined attributes and parse the .csv file and return dogs that meet those qualities, with reset resetting the list to default.

**Why MongoDB, Python, and Dash?**

* MongoDB allows the user to quickly and easily create different users with varying authority, load different databases, parse said databases for general, or specific queries and more. Python is also a very user-friendly programming language, where one with a good general understanding of computers could figure out how to modify the program/functions could easily do so by utilizing documentation or other online resources. Working with both Python and MongoDB allows for the use of PyMongo, which is a tool to fully interface Python with MongoDB to execute CRUD functions within the database. Utilizing both MongoDB and Python are ideal as Grazioso Salvare wants to make this program available to any shelter/organization that wishes to use it. Dash is a Python framework that allows for the creation of user-interfaces. The model-view-controller architecture is used to allow the controller to pull data, from this case the database, and then control and decide how the data is displayed to the user, in this case, passed to the Dash.

**Screencast**

* To better convey how the Dash app works, included is a screencast to the basic navigation of the dashboard.
* <https://www.youtube.com/watch?v=T1pZVcEFuBE>

**Development**

* This project was developed very incrementally. We began by ensuring the database was uploaded to MongoDB and performing basic queries to ensure information was being parsed correctly. We then next created user authentication for both an Admin and aacuser1 to be able to execute CRUD functions in the database. Next we created the CRUD Python application and ran the above tests to ensure all CRUD functions were successful. Lastly, we created the Dash application tying the whole project together.
* Very few issues came up during the development. Issues mostly consisted of not having MongoDB running while trying to run the Python applications (always ensure it’s up!).

## Contact

Jeremy Woods