# CS 340 README Template

## About the Project/Project Title

The MongoDB CRUD application utilizing Python seeks to provide management of a collection using a data set provided by Austin Animal Shelter. This project has been extended in order to create a dashboard for Grazioso Salvare, who wishes to identify dogs for training using specific parameters.

## Motivation

The CRUD application was developed to make the process of managing the collection of animals more streamlined as well as allowing access to multiple users. By being able to create new animal objects and search for specific parameters to find a specific animal, the program allows for easy access to the objects within the collection as well as extending the collection.

For Grazioso Salvare’s dashboard, both a histogram and geolocation chart are displayed in order to display the distribution of needed breeds as well as locations of specific animals. Filtering options are available that narrow down animals’ sex upon outcome, breed, and age that fit the qualifications for certain types of rescue training. Currently, the dashboard filters Water Rescue, Mountain and Wildlife Rescue, and Disaster Rescue/Individual Tracking. There is also a Reset button that resets the filter.

Utilizing Python was a choice made due to Python’s simplicity of syntax and ease of reusability, as well as its effortless integration with MongoDB thanks to the readily available PyMongo tool. Python also compiles quickly and efficiently, also working well with Jupyter Notebook in order to create test cases to assess the program’s functionality.

## Getting Started

1. You will need to use a MongoDB client to start utilizing the CRUD application. Enter the Mongo terminal to continue.
2. You will need the ‘aac\_shelter\_outcomes.csv” file to import the collection. To do so, locate the directory that your .csv file is located; using the mongoimport command, be sure to use the parameters of “—type: csv” and “—file /[directory]/aac\_shelter\_ouctomes.csv”
3. Enter user credentials. Username: “aacuser”, password: “aacUserPassword”
4. You will need access to Python and the provided “AAC\_CRUD.py” for reading/writing and “AAC\_CRUD.ipynb” for testing.

## Installation

Before attempting to utilize the CRUD application, be sure to have access to [MongoDB](https://www.mongodb.com/) as well as the latest version of [Python](https://www.python.org/downloads/) and [Jupyter Notebook](https://jupyter.org/install).

Within the terminal, you’ll want to use “-m pip install pymongo” if pymongo is not already installed; Mongo will not be able to read the .py files if this is not done beforehand.

## Usage

### Code Example

Below are the definitions of the writing and reading functions within the AAC\_CRUD.py file.

* The *create* function is used to add an object to the collection; this will only occur if the data parsed to the function is not null, else an exception is raised. The function will return True if data was added and False if no data was added.
* The *read* function will use *search\_data* that is parsed to the function, and will use the .find() operation in order to print a list of animals that match the search query. If no *search\_data* is parsed to the function, it will return an empty list.

#create

def create(self, data):  
 if data is not None:  
 insert\_dict = self.database.animals.insert\_one(data) # data should be dictionary  
 if insert\_dict != 0: # return true if insert\_dict contains data  
 return True  
 else: # false otherwise  
 return False  
 else:  
 raise Exception("Nothing to save, because data parameter is empty")

#read

def read(self, search\_data):   
 search\_result = []  
 if search\_data is not None: # parameter is not empty  
 if search\_data == "all":  
 search\_result = self.database.animals.find()  
 else:  
 search\_result = self.database.animals.find(search\_data) # find search\_data in db "animals"   
 else:  
 raise Exception("Nothing to read, because data parameter is empty")  
 return search\_result

### Tests

For testing in Jupyter Notebook, the following lines should be entered in order to start:

import AAC\_CRUD

%run AAC\_CRUD

animals = AnimalShelter()

Afterwords, the *create*, *read, update,* and *delete* functions can be tested by creating a new object using applicable parameters based on the parameters provided in the aac\_shelter\_outcomes.csv and then reading, updating, or deleting an object by using specific parameters for querying.

### Screenshots

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

## Roadmap/Features

Successfully implemented in current update:

* *Grazioso Salvare dashboard*

Known issues:

* *Filters may fail to update the list, histogram, and/or map on certain platforms*

## Contact

Your name: Michael Duteau