# CS 340 README

## About the Project/Project Title

*Grazioso Salvare, a rescue-animal training company, is trying to find good candidate dogs that will aid them in their rescue missions. They have partnered with an agency that operates five animal shelters and they need a software application that will be able to identify and categorize dogs, based on the animal shelters current data.*

## Motivation

*The main motivation behind this project is to train sheltered dogs into search-and-rescue dogs. The dogs will have to meet certain requirements so that they can be considered for training.*

## Getting Started

*To initiate the program the user must install MongoDB, this can be done by simply following* [*these steps*](https://www.geeksforgeeks.org/guide-install-mongodb-python-windows/)*. The user should then import the CSV file that is being used and verify that they login with their credentials. This will give them access to the file, in which they be able to read and edit the data. The user will then be able to use the provided Jupyter Notebook files to make any required changes. The user will be able to create new fields, search the document by using a keyword, update any of the data, and also delete any data in the document. Test code has also been provided to help the user see how to use the CRUD operations. We also created a simplified Dash application, the ProjectTwoDashboard file, that will allow the user to easily search the data file. The user can run the file in Jupyter Notebook and will see that the connection to the database was successful and also a link to the Dash application. The dash application features an interactive data table, it allows the user to search by any of the data items and allows the user to filter results based on the rescue specification provided. A pie chart and map are also included to show the results.*

## Installation

*For this project we will be using the MongoDB database, Jupyter Notebook, Python, and Dash. The MongoDB database is the most popular database and is known for its real-time analytics, their horizontal scaling, and that it doesn’t require pre-defined schemas. It is the best place to have our data stored. Python is also the most popular programming language and can easily be incorporated with MongoDB. Jupyter Notebook will allow us to test and run our code easily and will allow us to use the CRUD functionality on the database. Dash is also a great software to use to create applications using Python. Dash allows software engineers to easily add interactive graphs and data tables using Python. The* [*Dash website*](https://dash.plotly.com/) *was a great tool that I used throughout my projects, it showed me how to properly use Dash, and how to use some of its many functions. All of these software’s can easily be installed by Googling and downloading them. Many of the issues I encountered were mainly due to user error, and I was able to overcome them by using the Dash website, YouTube, and Stack Overflow.*

## Usage

*The following is the code used to create, read, update, and delete the data in the database. Also shown is a way to test that the code is functioning correctly, a screenshot is also shown to better demonstrate the test code.*

### Code Example

*# Method that implements the C in CRUD (create).*

*def create(self, data):*

*if data is not None:*

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

*return True # Returns true if animal was successfully added*

*else:*

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

*#Method that implements the R in CRUD (read).*

*def read (self,data):*

*if data is not None:*

*variable = self.database.animals.find(data)*

*return variable # Returns the data that matches*

*else:*

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

*#Method that implements the U in CRUD (update).*

*def update (self,data,newData):*

*if data is not None and newData is not None:*

*result = self.database.animals.update\_many(data, {'$set': newData})*

*return result.modified\_count # Returns the numer of objects modified in the collection*

*else:*

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

*#Method that implements the D in CRUD(delete).*

*def delete (self,data):*

*if data is not None:*

*result = self.database.animals.delete\_many(data)*

*return result.deleted\_count # Returns the numer of objects modified in the collection*

*else:*

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

### Tests

*#Test the above code*

*from animalShelter import AnimalShelter*

*CRUD = AnimalShelter("aacuser", "aacuser")*

*# Tests that the create method is functioning correctly*

*created = CRUD.create({"animal\_type": "Dog"})*

*print (created)*

*# Tests that the read method is functioning correctly*

*record = CRUD.read({"animal\_id": "A664290"})*

*for r in record:*

*print(r)*

*# Tests that the update method is functioning correctly*

*updated = CRUD.update({"animal\_id": "A664290"}, {"animal\_id": "A6642901"})*

*print("Numer of documents updated:", updated)*

*# Tests that the delete method is functioning correctly*

*deleted = CRUD.delete({"animal\_id": "A6642901"})*

*print("Numer of documents deleted:", deleted)*

### Screenshots

Imported CSV file:

A screenshot of a computer screen

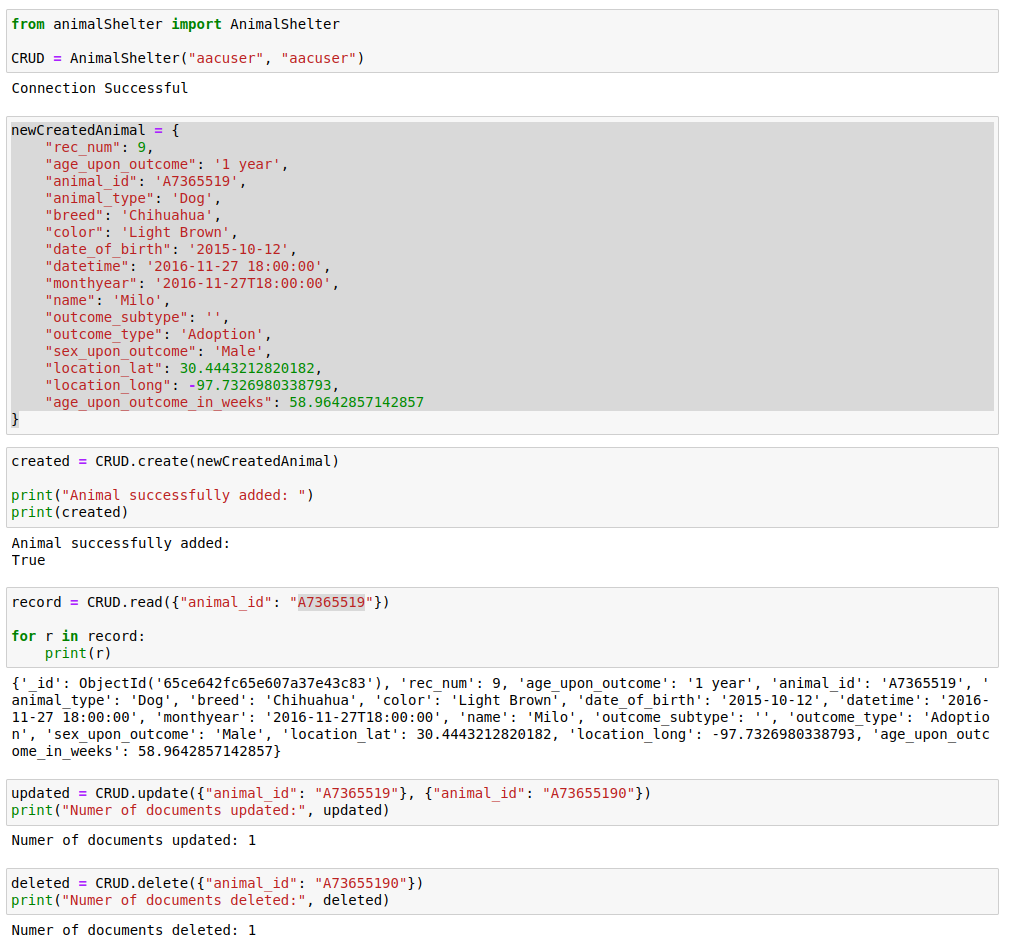
Description automatically generated

User Authentication:

A screenshot of a computer program

Description automatically generated

CRUD functionality test execution:

**

Starting base:

A screenshot of a computer

Description automatically generated

Water Rescue:

A screenshot of a computer

Description automatically generated

*Mountain or Wilderness Rescue:*

*A screenshot of a computer

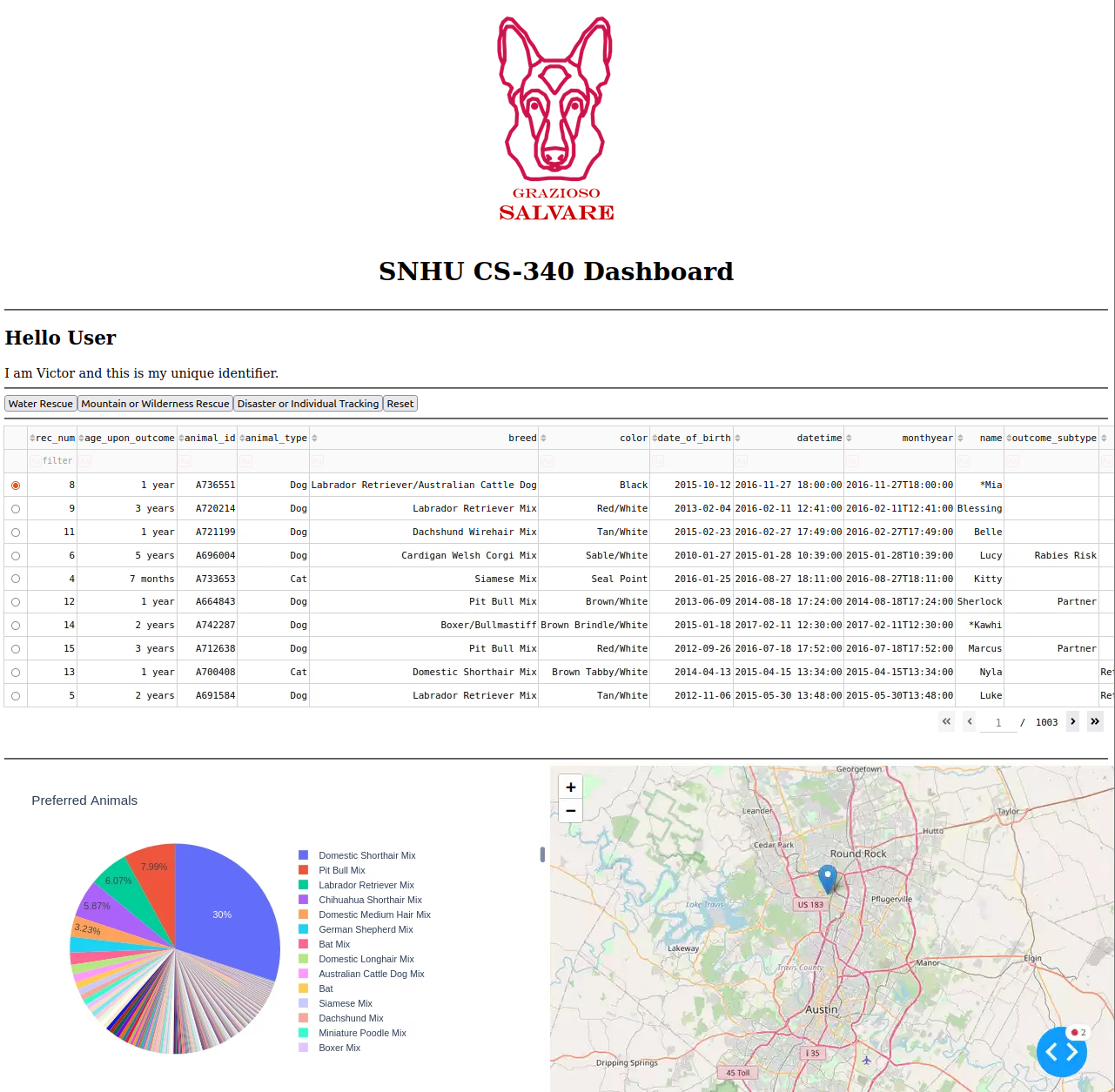
Description automatically generated*

*Disaster or Individual Tracking:*

*A screenshot of a computer

Description automatically generated*

*Reset:*

**

## Contact

Victor Sanchez: victor.sanchez3@snhu.edu