# CS 340 README – Project Two

## About the Project/Project Title

*Grazioso Salvare Search-and-Rescue Candidates*

*Grazioso Salvare identifies dogs that are good candidates for search-and-rescue training. When trained, these dogs are able to find and help to rescue humans or other animals, often in life-threatening conditions. To help identify dogs for training, Grazioso Salvare has reached an agreement with a nonprofit agency that operates five animal shelters in the region around Austin, Texas. This non-profit agency will provide Grazioso Salvare with data from their shelters.*

*Grazioso Salvare has noted that there are specific types and breeds of dogs to train. For instance, search-and-rescue training is generally more effective for dogs that are no more than two years old. Additionally, certain breeds of dogs are proficient at different types of rescue, such as water rescue, mountain or wilderness rescue, locating humans after a disaster, or finding a specific human by tracking their scent.*

## Motivation

*This is a software application that can work with existing data from the animal shelters to identify and categorize available dogs. Global Rain has been contracted for a full stack development of this application that will include a database and a client-facing web application dashboard. Grazioso Salvare will use this dashboard to interact with and visualize data from a MongoDB database. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and training time.*

## Getting Started

*To get a local copy of this software application follow these simple steps:*

1. *Go to the GitHub repository*
2. *Download repository files locally*
3. *Run JupyterDash and import the files into JupyterDash app*

## Installation

*To do this, simply download Jupyter Notebook and follow these steps to get to the console:*

1. *Download MongoDB Server*
2. *Download Jupyter Notebook*
3. *Create Jupyter Dash app using Jupyter Notebook*
4. *Import the GitHub repo files for CRUD functionality*
5. *Run Jupyter Dash app through your choice of software*

## Usage

*MongoDB*

*MongoDB makes it much easier for developers to store structured and unstructured data. It can make use of most modern programming languages due to the format it uses so the developers don’t need to think about normalizing data. It can also handle high volume, is highly available, and scalable. MongoDB allows for developers to immediately start building an application without spending time configuring a database. Most importantly, it is free and-open source.*

*Python*

*Python is a high-level, modern programming language and is used in this application due to its usability with MongoDB through PyMongo driver, this allows for great interconnectivity. The connection between these two is embedded through the animal\_shelter.py file that implements CRUD to create, read, update, and delete files within the database.*

*Dash*

*Since we are designing a web application that uses Python, it makes sense to use the Python framework for this job. This allows for continual development of the full-stack using Python without having to implement additional languages. Just as PyMongo allows for interconnectivity, Dash continues to follow suit of this interconnectivity and creates seamless communication for full-stack development.*

### Code Example

*What this application can do:*

1. *Text

   Description automatically generatedCreate new singular indexes*
2. Text

   Description automatically generated*Create compound indexes*
3. Text

   Description automatically generated*Create Logins for Admins and Users and each with their own rights*

*Implementation of CRUD (Create Read Update Delete) methods. The purpose of the CRUD Python module is to allow the user to work with the data in a database. These four operations are fundamental operations in any database.*

*![Graphical user interface, text, application, email

Description automatically generated]()Create method*

*![Graphical user interface, text, application

Description automatically generated]()Read method*

*![Graphical user interface, application

Description automatically generated]()Update method*

*Delete method*

*![A picture containing background pattern

Description automatically generated]()*

*The layout of this application is set up with centered headers and logo. The table is set up below the headers and logo with the charts on the same level below the table. Below you will find the code for the interactive radio buttons, the data table style, and chart layout.*

*![Text

Description automatically generated with medium confidence]()Radio button code*

*![Graphical user interface, text, application

Description automatically generated]()Data table style*

*![Text

Description automatically generated]()Chart layout*

*The pie chart and geolocation map code can be seen below. The pie chart is created from the page that the user is currently on in the data table. The data that the pie chart utilizes is the breed. The geolocation map takes the latitude and longitude info from each animal on the page then creates markers for them. If the cursor is hovering over the marker, the animal breed is given, but if the user clicks on the marker then the animal’s name pops up.*

![Text

Description automatically generated]()

### Tests

*Four tests below show how the application can add an animal into the database, how to find the animal using a key, how to update an animal’s info, and how to delete an animal from the database.*

* *The first test inserts data into the database using the create method while the second text find the inserted data to verify it exists using the read method. This was verified in the mongoDB console that the animal added through the test now exists in the database. The test also output a print statement confirming the test passed.*
* *The third test tested the update functionality of the CRUD method. This test took two arguments which consisted of finding the document with the first argument and then updating the document with the second argument. This test passed by printing the statement “Animal Updated” and this was also confirmed with the mongoDB console.*
* *The final test tested the delete functionality of the CRUD method. This test took one argument which found the selected information and deleted all documents which contained the selected information. The data can be made more specific so that it targets only one document, or it can be kept general so that it deletes multiples. The test passed and it was confirmed through the mongoDB console.*

*![Graphical user interface, text, email

Description automatically generated]()*

## Screenshots

![Text

Description automatically generated]()Import execution

*Text

Description automatically generatedUser authentication execution*

## Functionality

*A picture containing table

Description automatically generatedOnce the database is set up and the application is running, the dashboard of this application will open to this main page. By default, the radio button, ‘Reset’, is chosen so that all the data is compiled without any restrictions.*

*If Water Rescue is chosen, the table and charts will update to its filter settings.*

*Graphical user interface

Description automatically generated with low confidence*

*If Mountain or Wilderness Rescue is chosen, the table and charts will update to its filter settings.*

*![A picture containing application

Description automatically generated]()*

*If Disaster or Individual Tracking Rescue is chosen, the table and charts will update to its filter settings.*

*![A picture containing graphical user interface

Description automatically generated]()*

## Challenges

*There were a few challenges that came along with this project when it came to the creation of the dashboard. Most of my issues came with the geolocation chart because it would not generate when I tried coding it for selected row. I eventually figured out how to have the chart pull all information from the current table page and generate a marker that way. Each page shows markers related to the animals on that page.*

## Roadmap/Features

*This is a functional web application that can be utilized immediately but it still requires revisions, such as generating a marker for the selected row. Before the final version release, this will be fixed and functional. If there are any questions, comments, concerns, or if you would like to contribute to this application, please contact me via e-mail.*

## Contact

Vitalie Cucuta

[vitalie.cucuta@snhu.edu](mailto:vitalie.cucuta@snhu.edu)

## Resources

Dash Core Components - <https://dash.plotly.com/dash-core-components>

Dash DataTable - <https://dash.plotly.com/datatable>

Dash Leaflet - <https://dash-leaflet.herokuapp.com/>

Dash Pie Charts - <https://plotly.com/python/pie-charts/>

MongoDB Manual - <https://docs.mongodb.com/manual/>