# CS 340 Project Two

## About the Project / Search & Rescue Dog Dashboard

This dashboard is a web application designed for Grazioso Salvare to identify good dog candidates for search and rescue training. This data is compiled from local animal shelters in Austin, TX and the surrounding areas.

Using Pythons dash framework this application is built and interacts with the CRUD Python Module which is the middle layer for the stacked development. Basically, it’s the glue between the base level and the client level. Raw data is the base level which is maintained by the MongodDB tool.

## Motivation

The client wishes to identify specific traits in a large set of canine data easily. These traits are separated into three categories: water, mountain and wilderness, disaster and individual rescue.

MongoDB is desired because of its natural interconnectivity across all platforms such as Python and the Dash framework. MongoDB is also free and open source which can support large scale projects with rapid development timelines. PyMongo is needed to communicate with Python. MongoDB is great because it has a vast variety of tools that allows for fast querying through large data sets with indexing.

* <https://docs.mongodb.com/manual/introduction/> is the location of the MongodDB manual

Python is the most flexible and powerful programming language used in various applications to this day. It pairs very well with data analysis and communicates very good with MongoDB. Using PyMongo driver you can have a powerful connection to the MongoDB server. Embedded in the CRUD.py module is the connection. This allows inserting, modifying, querying and removing documents within the database.

* <https://docs.python.org> is the Python manual
* <https://pymongo.readthedocs.io/en/stable/> is the PyMongo manual

Dash is the Python framework which designs web applications. It is also free and open source like the other software we used. So, you don’t have to use other languages such as HTML or JavaScript the Python Dash framework allows development of the full stack using only Python. This then allows for good communication between all three levels of stack development.

* <https://dash.plotly.com/> is the location for Dash manual.

## Getting Started

1. *Install MongodDB, Python, PyMongo distribution and the project\_one.py to get started.*

*(View Installation section below)*

1. *Initiate the Mongo server with no authorization.*

*(/usr/local/bin/mongod\_ctl start\_noauth)*

1. Graphical user interface, text, application

   Description automatically generated
2. *Now start mongo by typing “mongo”*
3. Graphical user interface, text, application

   Description automatically generated
4. *Enter “use admin” which will enter into the admin database and enter “db.createUser()” this will create the admin account.*
5. Graphical user interface, text, application

   Description automatically generated
6. *The following steps will exit mongo and restart with authentication.*
   1. *Exit*
   2. */usr/local/bin/mongod\_ctl stop*
   3. */usr/local/bin/mongod\_ctl start*
7. Graphical user interface, text, application

   Description automatically generated
8. *Now start mongo (mongo -authenticationDatabase “admin” -u “username” -p “password”)*
9. *A screenshot of a computer

   Description automatically generated*

## Installation

* [*https://www.mongodb.com/try/download/community*](https://www.mongodb.com/try/download/community) *- MongoDB Community Server*
* [*https://www.python.org/downloads/*](https://www.python.org/downloads/) *- Python*
* [*https://pypi.org/project/pymongo/*](https://pypi.org/project/pymongo/) *- PyMongo*
* [*https://dash.plotly.com/installation*](https://dash.plotly.com/installation) *- Dash*
* [*https://plotly.com/python/getting-started/*](https://plotly.com/python/getting-started/) *- Plotly*

## Usage

The goal for project one is for a client/web application to access and query the database on the server side.

This is the create function and allows for the creation of a new document.

Graphical user interface, text

Description automatically generated

This is the read function which allows users to query existing documents within the AAC database.

Graphical user interface, text

Description automatically generated

The delete function locates and deletes it from the database.

Graphical user interface, text

Description automatically generated

The update function is used by the client-side application to locate and update an existing certain data.

Graphical user interface, text

Description automatically generated

For the create function test a set of document data is wrote as json and converted to a python dictionary which is assigned to a test variable.

Graphical user interface, text

Description automatically generated

A key or value pair is used for the read function test.

Graphical user interface, text

Description automatically generated

The update function test sends two variables as arguments to the function. The first one will query the database and the second is the updated data.

Graphical user interface, text

Description automatically generated

The delete function test the key function as the only argument and the specific document will be deleted once found.

Graphical user interface, text

Description automatically generated

## Contact

Jerry Fugate