# **CS 340 README**

## **About the Project**

The ProjectOne Python program is designed to simplify interactions for users executing CRUD operations for a MongoDB database. The Jupyter Notebook file implements this .py file along with other frameworks and libraries in order to produce a dashboard that displays a table, geolocation chart, and pie chart that display data based on certain types of rescue missions that the user can select using radio buttons at the top of the dashboard. The image that I incorporated in the header was given by the client and includes a link to their homepage.

## **Motivation**

This project was created as part of a project for a class I am taking to complete my Bachelors in Computer Science. The Python file helped guide my learning on creating a program that directly interacts with a MongoDB database. The Jupyter Notebook file provided me an opportunity to learn more about User Interfaces and incorporating data from a MongoDB database into the UI.

**Client Requirements**

The client for this project is in the business of training rescue dogs. For this specific project, the client wanted a program for finding dogs from five different animal shelters in the Austin, Texas area that would be ideal candidates for their different training programs. They provided the dataset from the shelters as well as the dog qualities that each training program needed. For the interface, they asked that the dataset be displayed as a table with radio buttons to specify the candidates that met their criteria for each rescue program. They also wanted a geolocation chart that allowed individual animal’s locations to be displayed and a pie chart that specified the breed breakdown of each dataset. The client also wanted their logo to be displayed in the header of the dashboard with a link to their homepage. I have attached a recording of the finished product that displays all of this functionality being implemented.

## **Getting Started**

In order to run this project on your local machine, you will need to update the Python file and change the connection variables in the constructor method – USER, PASS, HOST, PORT, DB, and COL. USER and PASS will verify the credentials of the user. HOST and PORT will vary depending on your local machine. DB and COL define the database and collection the program will access. Ensure that all of the frameworks and libraries that the Jupyter Notebook file implements are up-to-date. The .ipynb file includes hardcoded username and password credentials for accessing the database. These will need to be changed to access your specific database or you can implement a login feature with a callback that takes user input directly from the dashboard to access a specific database.

## **Installation**

1. Clone repository to your local machine
2. Install and run MongoDB
3. Install Python driver and packages and run the Python file
4. Install and run Jupyter Notebook
5. Upload the .py, .ipynb, and .png files to the same location with Jupyter Notebook
6. Run the .ipynb file and follow the link to the dashboard

**Tools**

* Pymongo was chosen as the Python Driver for MongoDB because it is recommended by MongoDB and due to its simplistic API and easy integration with MongoDB.
* Dash Framework was chosen because of its simple integration with Python to create a Web-based application, and it allows for a developer-friendly implementation of the View and Controller sections of an MVC design pattern in a single program.
* Python, Plotly, and Pandas were all chosen because of the ease-of-integration with each other and simple data manipulation functions.

**Developer Process**

In order to complete this project, I broke it up into different sections and tackled each one independently. The Python file came first, and allowed me to focus on creating a solid base for the CRUD operations to effectively interact with the MongoDB database. After the Python file, I began working on the .ipynb file in order to implement the view and controller aspects of the design pattern. I broke this up into two sections itself – the view and the controller. While the Dash framework provides an easy integration of both in one file, I found it easier during development to focus on the view first before implementing my controller logic. Building the frame before filling it up felt to be a smart design choice, especially due to Dash’s use of callbacks in the controller logic to be connected to the specific view id’s. So I built the layout with html and then designed the table, geolocation map, and pie chart. Then I implemented the controller logic to connect them together.

**Challenges**

The biggest challenge during the development of this project was getting accustomed to the Dash framework and how it implemented callbacks to interact between the view and controller sections of the program. Once I had a firm understanding of this interaction, the project went fairly well without many major hiccups. Understanding the syntax of the other implemented libraries took a bit to familiarize myself with, but overall, the development process went pretty smooth given the amount of new material I had to learn.

## **Contact**

Brady Weber