# CS 340 Project 2 README

## About the Project/Animal Shelter Web Application

This project revolves around collaborating with an innovative international rescue-animal training organization. The primary objective is simplifying the identification and selection process for dogs with search-and-rescue operations ability. These specially trained dogs play a crucial role in locating and rescuing both humans and animals in hazardous situations. The proposed software application is designed to leverage existing data from animal shelters, facilitating the efficient identification and categorization of dogs suitable for training.

The comprehensive full-stack application incorporates the development of a robust database and an intuitive client-facing web application dashboard. This integrated system empowers users, enabling them to seamlessly access and manage the database. The goal is to enhance the efficiency and effectiveness of the rescue-animal training process by leveraging technology to identify and prepare canines for life-saving missions.

## Motivation

The motivation behind this project is to streamline the process of handling database operations in Python applications. Often, developers need a reliable and efficient way to perform CRUD operations, and this module intends to serve that purpose. By providing a clean and easy-to-use interface, it aims to enhance the development workflow for projects that involve database interactions.

## Getting Started

To get a local copy up and running, follow these simple example steps:

1. **Linux terminal:** The Linux shell is used to connect to the MongoDB shell and run commands.
2. **Database Setup:** Ensure you have a database installed and configured with user authentication set up.
3. **Python Environment:** Create a virtual environment and install the necessary dependencies.
4. **Configuration:** Use Mongo import tools to include your database credentials and connection details.

## Installation

The tools used in this project and their installation instructions are as follows:

* **MongoDB:** Chosen as the database for storing the animal shelter data.
* **Python:** The core language for development.
* **Jupyter Notebook:** A toolkit for Python.
* **Jupyter Dash:** Used for its easy integration with Jupyter notebooks and its ability to create interactive web applications.
* **Dash Leaflet:** Chosen for its compatibility with Dash and its features for displaying maps.
* **Plotly Express**: Selected for its simplicity in creating interactive charts.

## MongoDB

## MongoDB was used as the model component of the development because of its ability to store JSON-like documents, which aligns well with the Python data structures. This made it easy to interface with Python, as data could be easily read from and written to the database using Python's dictionary-like objects.

## Dash Framework

## The Dash framework was used to provide the view and controller structure for the web application. It allowed for the creation of interactive components such as the data table, charts, and map, and facilitated the communication between these components.

## Usage

### Imported Animal Shelter Dataset A screenshot of a computer Description automatically generated

**User connection to Mongo**

A screenshot of a computer

Description automatically generated

**User Authentication**

A screenshot of a computer screen

Description automatically generated

**Python CRUD module**A screenshot of a computer program

Description automatically generatedA screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

### Tests

In this testing script:

* We import the **CRUD** class from our CRUD Python module.
* We instantiate the **CRUD** class for the MongoDB database.
* We define four test functions, **create**, **read**, **update**, and **delete** to test the **CRUD** functionality, respectively.
* In the **create** function, we create a new animal record and then read it from the database to verify its insertion.
* In the **read** function, we read all data from the database to ensure that data exists.
* In the **update** function, we update a document matching the search criteria with the provided data.
* Lastly, the **delete** function deletes a document matching the search criteria.
* We run the **main** test function to verify the functionality of our CRUD module. It first creates a document, then reads it, updates it, and finally deletes it.

To execute this script in Jupyter Notebook, follow these steps:

1. Open Jupyter Notebook.
2. Create a new notebook file.
3. Copy the code into cells in the notebook.
4. Run each cell to execute the code.
5. Observe the output of each cell to ensure the tests pass successfully.

### Python Testing ScriptA screenshot of a computer Description automatically generatedA screenshot of a computer program Description automatically generatedA screenshot of a computer program Description automatically generated

**Python Controller**

**Steps to implement the Dashboard**

1. Set up the Jupyter Dash environment.
2. Import necessary libraries and modules.
3. Connect to the MongoDB database.
4. Create the layout for the dashboard, including the data table, charts, and map.
5. Implement callbacks to update the components based on user interactions.
6. Test the dashboard to ensure functionality.

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**Interactive Dashboard**

**A close up of a logo

Description automatically generated**

**A screenshot of a dashboard

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a dashboard

Description automatically generated**

**A screenshot of a map

Description automatically generatedA screenshot of a map

Description automatically generatedA screenshot of a map

Description automatically generated**

**Challenges**

Many challenges arose while developing this interactive dashboard. They include:

* Importing the CRUD module
* Instantiating the CRUD module and connecting to the database
* Populating the Data table with an interactive filter
* Populating the pie chart and map with updated data
* Ensuring that the callbacks update the components correctly based on user interactions.

## Contact: Tinisha Cain-Beckford

**References**

*Austin Animal Center. (2020).*Austin animal center outcomes [Data set]. City of Austin, Texas Open Data Portal. <https://doi.org/10.26000/025.000001>

*Basic pie chart — Matplotlib 3.3.4 documentation*. (n.d.). Matplotlib.org. <https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html>

*Dash Leaflet*. (n.d.). Www.dash-Leaflet.com. <https://www.dash-leaflet.com/>

Guo, D. (n.d.). *Make a README*. Make a README. <https://www.makeareadme.com>

*How to Write Beautiful and Meaningful README.md*. (2020, January 19). Silentlad. <https://silentlad.com/how-to-write-beautiful-and-meaningful-readme.md>‌

*RadioItems | Dash for Python Documentation | Plotly*. (n.d.). Dash.plotly.com. <https://dash.plotly.com/dash-core-components/radioitems>

*mongoimport — MongoDB Database Tools*. (n.d.). Www.mongodb.com. <https://www.mongodb.com/docs/database-tools/mongoimport/>

‌