# CS 340 README Template

## About the Project/Project Title

This project is related to managing a CRUD database via Python code. These methods will ultimately support a user interface and dashboard related to the CRUD database.

## Motivation

This project exists to provide automated ways to interact with a Mongo database. Common interactions done manually quickly become unmanageable as an application scales, this code will provide an easy way to scale these interactions.

## Getting Started

Step 1: Launch Mongosh

Step 2: Switch to proper collection

Step 3: Instantiate AnimalClass class object

Step 4: class.add(record) to add a record.

Step 5: class.read(query) to read for a record.

## Installation

Tool List:

* Mongo Database
* Mongosh
* Python3
* Jupyter Notebook

**How to install:**

Windows:

* Install MongoDB:
* <https://www.mongodb.com/docs/manual/installation/>
* Install Python:
* <https://www.python.org/downloads/windows/>
* Install Mongosh:
  + <https://www.mongodb.com/docs/mongodb-shell/install/>
* Install Jupyter:
  + <https://jupyter.org/install>

**Steps taken to complete project:**

To complete this project, I started by setting up a MongoDB database to store and manage the data related to the rescue animals. Next, I built a Python class utilizing the PyMongo library to enable interactions with the MongoDB instance. This class encapsulated a suite of CRUD operations, providing the necessary functionality to create, read, update, and delete records as required by the application's logic. To further meet project needs, I built a set of queries capable of filtering the rescue dogs based criteria outlined by our project stakeholders. I also used the Dash framework to develop an interactive web dashboard that allows users to seamlessly browse through data using these queries. Throughout the development cycle, I engaged in iterative testing to ensure my dashboard and queries were sound, that my dashboard layout was user friendly and that all project requirements were met.

**Tools used and rationale:**

MongoDB was chosen for its scalability, flexible schema design, and powerful query capabilities, making it a great choice for managing the data sets associated with animals and their attributes. The choice to use Python for the server-side logic was driven by its widespread use, extensive library ecosystem, and its ease of integration with MongoDB through the PyMongo driver. On the front end, the Dash framework gives us the ability to create highly interactive web applications purely in Python, without the having to directly manipulate HTML. Dash's simplicity combined with its Plotly integrations allow for sophisticated data visualization that are appealing for end users. These three tools combine nicely with a web-server infrastructure to provide an interactive dashboard that is appealing, accurate and quick to respond to end users.

## Usage:

Import DB Screenshot:

A screenshot of a computer

Description automatically generated

**User creating and auth**:

Creation -

A screenshot of a computer program

Description automatically generated

Auth -

A screenshot of a computer program

Description automatically generated

**Instantiate a Class Object for the Animal Shelter Database:**

A white background with red and blue text

Description automatically generated

**Create a record:**

### Code Example

def create(self, data):

        if data is not None:

            # protect db insert call

            try:

               self.database.animals.insert\_one(data)

               return True

            except Exception as e:

                print(f"An error occurred: {e}")

                return False

### Tests

A screenshot of a computer

Description automatically generated

### Screenshots:

Screen shot showing that the record we created now exists in the database.

A screenshot of a computer

Description automatically generated

**Read a record from the DB:**

### Code Example

   def read(self, query):

        if query is not None:

            # protect db collection query

            try:

                documents = self.collection.find(query)

                return list(documents) # return a list of documents someone can iterate on

            except Exception as e:

                print(f"An error occurred: {e}")

                return []

### Tests

A screenshot of a computer

Description automatically generated

### Screenshots

Showing that reading from the database allows us to see the intended record:

A screenshot of a computer

Description automatically generated

**Updating a record:**

Code:

A screenshot of a computer program

Description automatically generated

Updating a single record test –

A screenshot of a computer

Description automatically generated

Updating many records tests:

A screenshot of a computer

Description automatically generated

**Deleting a record:**

Code:

A screenshot of a computer program

Description automatically generated

Delete single record tests:

A screenshot of a computer

Description automatically generated

Deleting many records test:

A screenshot of a computer

Description automatically generated

**Screen Shot – Main dashboard launches app without errors:**

**A screenshot of a computer

Description automatically generated**

**Screen shot** – Dashboard with Logo

A screenshot of a computer

Description automatically generated

**Interactive options showing the three main required categories:**

A screenshot of a computer

Description automatically generated

**Screenshot of choosing Water rescue filter:**

A screenshot of a computer

Description automatically generated

**Screen of Mountain or wilderness rescue:**

A screenshot of a computer

Description automatically generated  
**Query code that powers the filter:**

**A computer screen shot of text

Description automatically generated**

**Screen of Disaster tracking filter enabled:**

A screenshot of a computer

Description automatically generated

**Dash app showing extra pie chart that shows distribution of dog breed for the filtered rescue type**:

A screenshot of a computer

Description automatically generated

## Roadmap/Features Upcoming features:

Method to update a record in the database.

Method to delete a record from the database.

Dashboard front-end UI.

**Challenges:**

A lot of the challenges I experienced during this project came from using Apporto, the virtual desktop was slow to respond and often unstable. I had many days of crashing, losing screens, logging in and out several times just to get things working. I’m very used to developing my local machine so it felt more like a handicap to be on Apporto at times than a help. Also I think the challenges of using the Jupyter Notebook tool was a bit of a learning curve for me, I hadn’t had any experience with the tool prior to this class so there was some bumps early on moving my code in and out of the tools and my local machine. Overall I had a good experience with the project and learned a good deal about integrating code, non-relational databases and web-servers to build out simple applications like the dashboard we built here!

## Contact

Your name: [Anthony.Fillmore@SNHU.edu](mailto:Anthony.Fillmore@SNHU.edu)

**Resources:**

* Leaflet Documentation. (n.d.). Retrieved 4/20/2024, from <https://dash-leaflet-docs.onrender.com/>
* MongoDB Documentation. (n.d.). MongoDB. Retrieved 4/20/2024, from <https://www.mongodb.com/docs/>
* Plotly Technologies Inc. (2020). Dash for Python Documentation. <https://dash.plotly.com/introduction>