# CS 340 PROJECT TWO README

## About the Project/Project Title

This project is a Python-based dashboard application and CRUD module that connects to a MongoDB database to manage animal records for the Austin Animal Center (AAC). The dashboard uses the Dash framework to provide interactive data visualization, while the CRUD module enables Create and Read operations through a reusable, object-oriented Python class.

The purpose of the project is to demonstrate how Python applications can integrate front-end dashboards with a back-end NoSQL database using the pymongo driver. It highlights key client/server development concepts, including modularity, database authentication, and secure interaction with persistent data. This work also lays the foundation for scaling the system into a full-stack web application in the future.

## Motivation

Grazioso Salvare’s mission is to save lives by training dogs for search-and-rescue operations. Identifying suitable candidates for training is a challenge, especially when working with large amounts of shelter data across multiple facilities. The client needed a solution that could reliably analyze and present animal data in a way that would reduce manual effort and speed up decision-making.

Our team at Global Rain was tasked with developing a dashboard application that connects to a secure MongoDB database and provides Grazioso Salvare staff with an intuitive interface. This dashboard not only allows users to filter and explore animal records but also helps highlight dogs that fit the criteria for different types of search-and-rescue training—such as age, breed, and behavioral characteristics.

The motivation behind this project is to provide Grazioso Salvare with a powerful, user-friendly tool that reduces training time, minimizes data-entry errors, and ensures that the most promising candidates are quickly identified. By leveraging technology, this project helps transform shelter data into actionable insights that support Grazioso Salvare’s life-saving work.

## Getting Started

If you would like to try this project in your own Apporto environment, here’s how to get started:

1. **Start MongoDB**
   * Make sure your MongoDB server is running in the Apporto environment.
2. **Update Connection Settings**
   * Open the Python CRUD module file
   * Check the **HOST** and **PORT** values and update them if necessary to match your MongoDB environment.
3. **Use the Provided Credentials**
   * Log in.
   * This ensures you have access to the AAC database and animals collection.
4. **Run the Code**
   * Open a **Jupyter Notebook** or Python shell in Apporto.
   * Import and test the class by creating, reading, or updating documents in the animals collection.
   * You can also launch the **dashboard notebook** to explore and visualize animal data through the interactive map and widgets.
5. **Verify Functionality**
   * Confirm that CRUD operations work correctly in the MongoDB database.
   * Interact with the dashboard to filter and display animal records.

## Installation

These are the tools and technologies used in this project:

* **Python 3.10**  
  Used as the programming language for the CRUD module due to its simplicity and broad support for libraries like pymongo.
* **MongoDB 6.0 (Community Server)**  
  Used to store and manage animal records in a NoSQL format.
* **PyMongo 4.5.0**  
  Documentation: <https://pymongo.readthedocs.io/en/stable/>   
  Chosen as the official MongoDB driver for Python. It provides a clean API for connecting, querying, and managing MongoDB databases using native Python syntax.
* **Jupyter Notebook (via Anaconda)**  
  Used for testing and documenting the execution of CRUD operations interactively.

## Usage

Here’s how the code works:*.*

### Code Example

Filter animals for Water Rescue and display results

query = {"breed": {"$in": ["Labrador Retriever", "Chesapeake Bay Retriever"]}}

results = list(shelter.read(query))

for animal in results[:3]: # Show first 3 matches

print(animal["name"], "-", animal["breed"])

### Tests

To verify that the query and database connection are working correctly, you can run a few basic tests:

**# Test 1: Database Connection**

assert shelter is not None, "Shelter object not initialized."

**# Test 2: Query for Water Rescue breeds**

query = {"breed": {"$in": ["Labrador Retriever", "Chesapeake Bay Retriever"]}}

results = list(shelter.read(query))

assert len(results) > 0, "No results found for Water Rescue breeds."

**# Test 3: Validate fields in the returned documents**

for animal in results[:3]:

assert "name" in animal, "Missing 'name' field in result."

assert "breed" in animal, "Missing 'breed' field in result."ts:

### Screenshots

A screenshot of a map

AI-generated content may be incorrect.

**Demo Video**

The screencast of the dashboard and widget functionality is included in the submission files as `ProjectTwoScreencast.mp4`.

## Contact

Jordan Caizza

For questions or collaboration, reach me at jordan.caizza@snhu.edu.