# CS 340 README Stephan

## About the Project/Project Title

This Python program provides functionalities to interact with a MongoDB database and perform CRUD operations. It allows you to connect to a MongoDB server, access a specific database, and perform various operations like inserting, querying, updating, and deleting documents. It uses 4 collections, to help keep information about the animal shelter, its members and project/task management.

## Motivation

The reason for this project is to create a learning environment for students. It is a good example of creating code that is useful to interact with noSQL databases and provides code for reusability and consistency moving forward with projects. It creates various functions that allow the user to query tasks for specific users and data analysis for the animal data.

## Getting Started

To get started, add the .py module with the base code and credentials to the working directory or a location you can access. Once there import the AnimalShelterCRUD class.

## Installation

Once the class is imported it is important to have the other needed libraries. pyMongo and bson.objectid need to be in the environment.

## Usage

In order to use this product you need to ensure you have a user within the desired database with the required rights. The current code uses database “AnimalShelter” and user “admin” If you have other users it may need to be modified.

### Code Example

# Create document

inserted\_id = crud.create\_document(document)

print("Inserted document ID:", inserted\_id)

# Read document

query = {"\_id": inserted\_id}

retrieved\_document = crud.read\_document(query)

print("Retrieved document:", retrieved\_document)

# Update document

update\_query = {"\_id": inserted\_id}

new\_values = {"$set": {"age": 26}} # Update age to 26

modified\_count = crud.update\_document(update\_query, new\_values)

print("Modified count:", modified\_count)

# Delete document

delete\_query = {"\_id": inserted\_id}

deleted\_count = crud.delete\_document(delete\_query)

print("Deleted count:", deleted\_count)

#Set up generic tasks

feeding\_animals\_task\_id = create\_task("Feeding Animals", "Ensure all animals are fed according to their dietary requirements.", rDukeId)

cleaning\_enclosures\_task\_id = create\_task("Cleaning Enclosures", "Clean and sanitize animal enclosures to maintain hygiene and prevent the spread of diseases.", rDukeId)

# Data Entry Task

data\_entry\_task\_id = create\_task("Data Entry", "Enter new animal records into the database.", pStephanId, [])

input\_basic\_info\_subtask\_id = add\_subtask(data\_entry\_task\_id, "Input Basic Information", "Enter name, species, breed, and age of the animal.", rDuke)

# Print Adoption Rates by Animal Type and Sex

print("\nAdoption Rates by Animal Type and Sex:")

for rate in adoption\_rates\_by\_type\_and\_sex:

animal\_type = rate['\_id']['animal\_type']

sex = rate['\_id']['sex\_upon\_outcome']

count = rate['count']

print(f"Animal Type: {animal\_type}, Sex: {sex}, Count: {count} adoptions")

# Display tasks by member

print("Tasks Assigned to Member:")

for task in cursor:

assigned\_to = task.get('assigned\_to', 'Unassigned') # Get the assigned member

task\_name = task['name'] # Get the task name

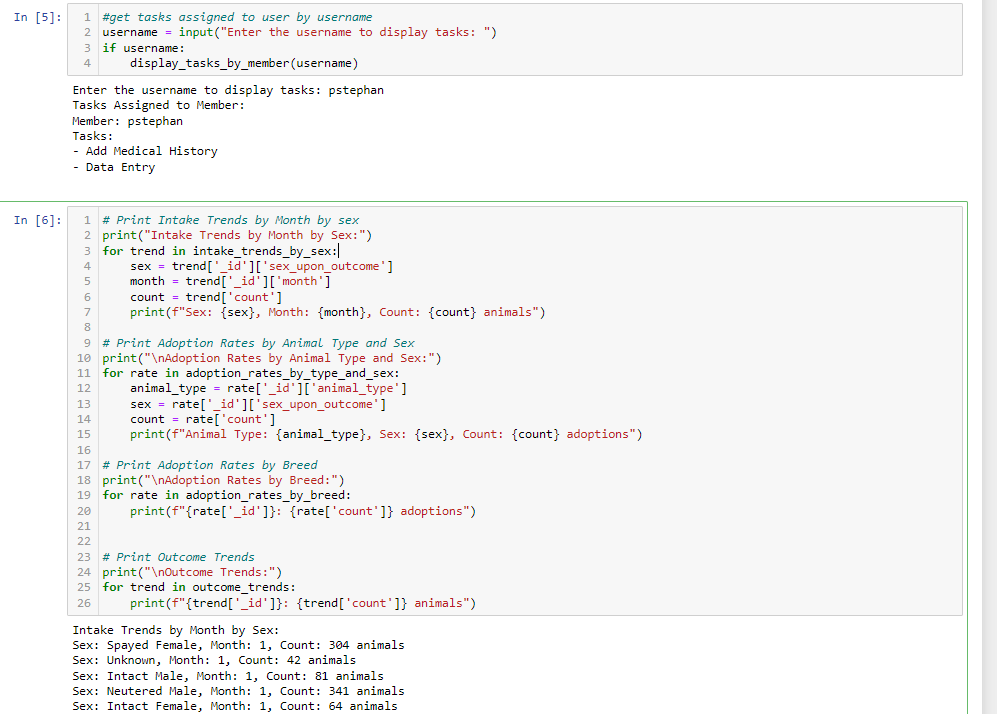
# Ensure assigned\_to is a string to use it as a key

assigned\_to = str(assigned\_to)

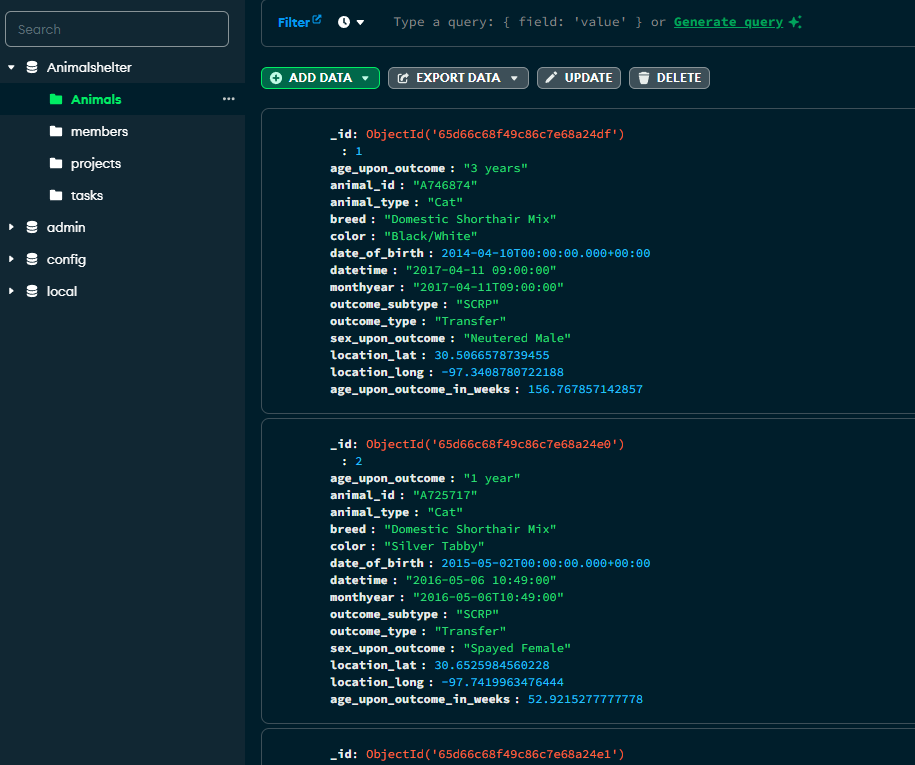
# Add the task to the set of tasks assigned to the member

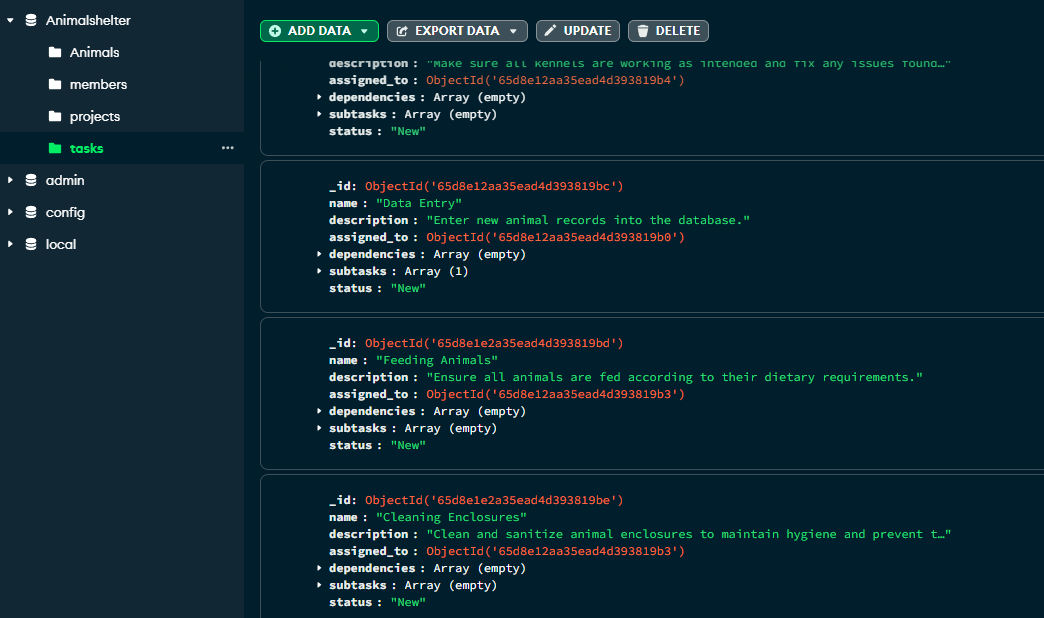
tasks\_by\_member.setdefault(assigned\_to, set()).add(task\_name)

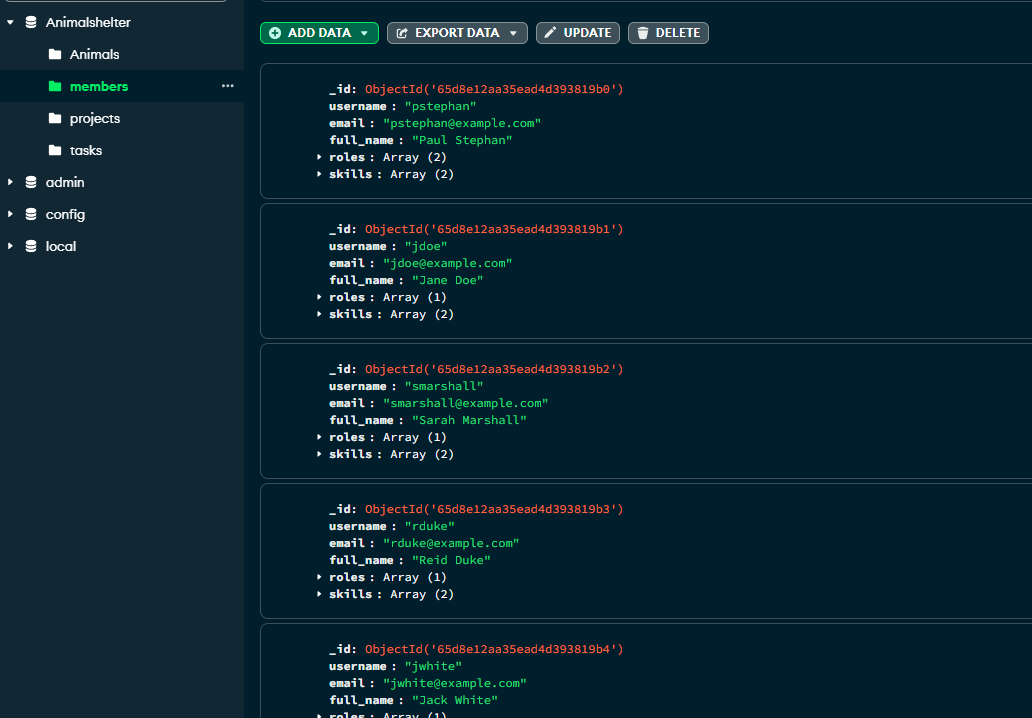
### Screenshots

**



**

**

**

## Roadmap/Features (Optional)

The roadmap is to create additional analysis functions for the users to query.

## Contact

Your name: Paul Stephan