Southern New Hampshire University

CS 340 README

Derricko Swink

About the Project/Project Title

This project is a Python module that connects to a MongoDB database and performs CRUD (Create, Read, Update, Delete) operations on the Austin Animal Center (AAC) Outcomes dataset. The main objective is to create a reusable Python module for database interaction, specifically to handle data insertion and querying within a MongoDB collection.

Motivation

The motivation for this project is to provide a scalable and maintainable way to interact with a MongoDB database. It will allow users to insert and query data from the AAC Outcomes dataset efficiently, supporting the development of a web-based dashboard in the subsequent project. This project forms the foundational CRUD operations for the larger system, enabling easy integration with other components.

Getting Started

To get a local copy of this project up and running, follow these simple steps:

1. Ensure you have Python installed on your system.
2. Install the necessary dependencies with pip install pymongo.
3. Download the aac_shelter_outcomes.csv file from the directory.
4. Set up MongoDB and import the dataset using the mongoimport tool.
5. Instantiate the AnimalShelter class to perform CRUD operations.

Installation

The tools required for this project are:

- Python (3.7 or later)
- MongoDB
- PyMongo (Python MongoDB driver)

NOTE: Ensure MongoDB is running locally or accessible through the given credentials in the project.

Usage

Use this space to show useful examples of how your project works and how it can be used. Be sure to include examples of your code, tests, and screenshots.

Code Example

MongoDB

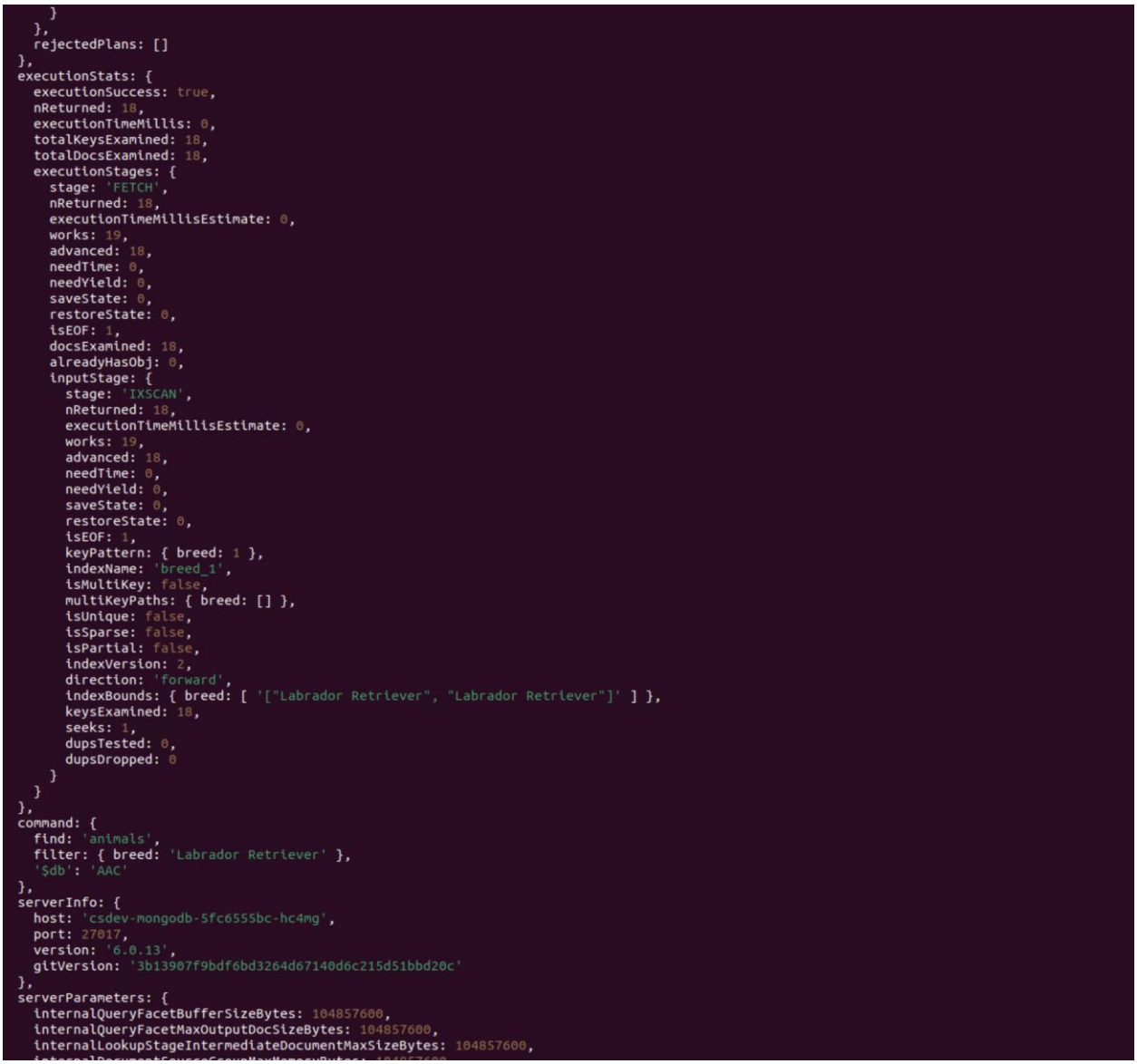
Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Southern New Hampshire University

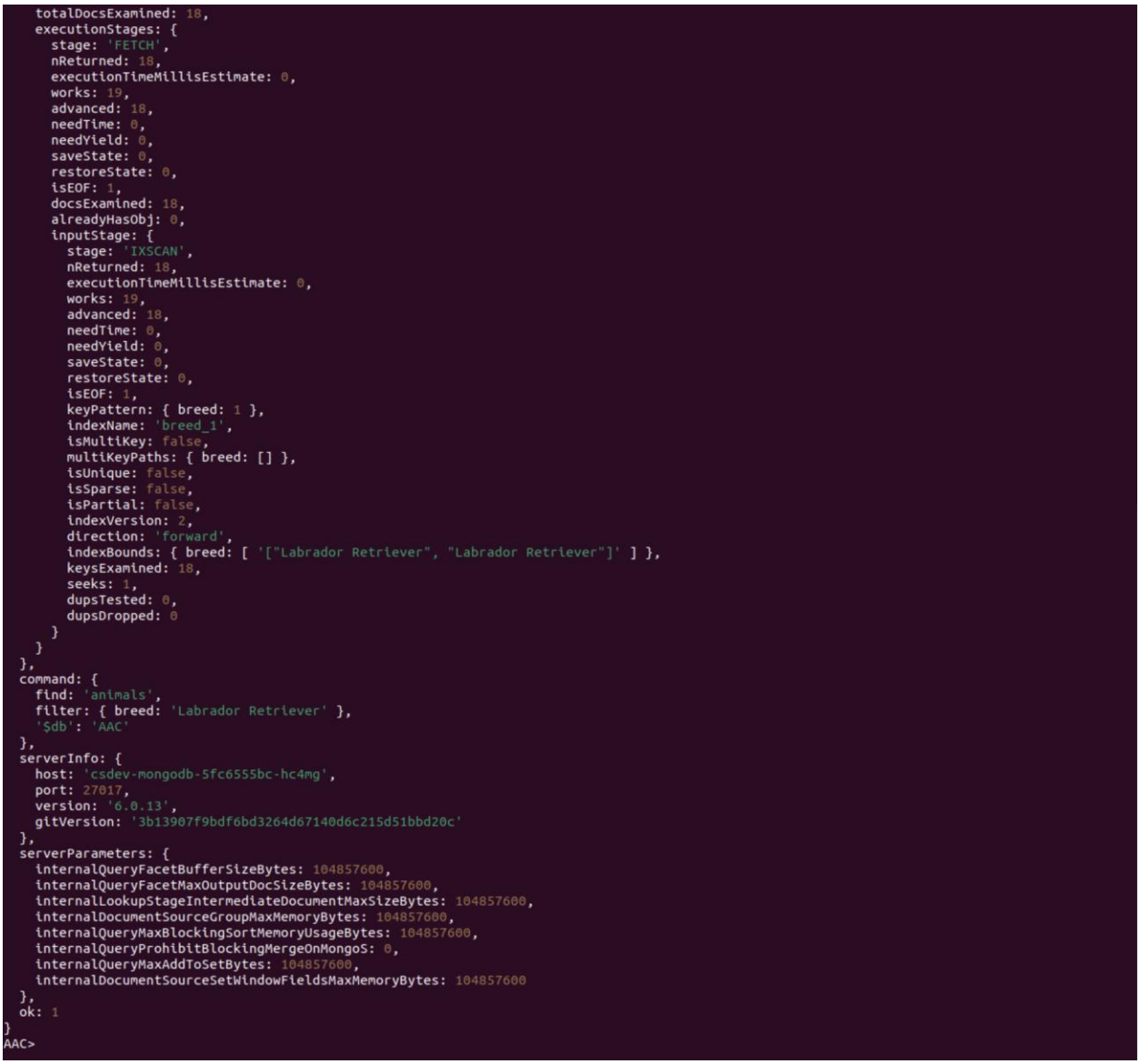
```
derrickoswink_snhu@nv-snhu8-l04: ~
(base) derrickoswink_snhu@nv-snhu8-l04:~$ mongoimport --host nv-desktop-services.apporto.com --port 31448 --username root --password 'ebx8x8wi
xl' --authenticationDatabase admin --db AAC --collection animals --type csv --file /usr/local/datasets/aac_shelter_outcomes.csv --headerline
2025-01-26T21:05:09.079+0000    connected to: mongodb://nv-desktop-services.apporto.com:31448/
2025-01-26T21:05:09.345+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
(base) derrickoswink_snhu@nv-snhu8-l04:~$
```

```
(base) derrickoswink_snhu@nv-snhu8-l04:~$ mongosh
Current Mongosh Log ID: 6796a3b2998ecd6e7dfbea10
Connecting to:          mongodb://<credentials>@nv-desktop-services.apporto.com:31448/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:          6.0.13
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-01-26T20:40:02.610+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongo
db.org/core/prodnotes-filesystem
   2025-01-26T20:40:04.735+00:00: Failed to read /sys/kernel/mm/transparent_hugepage/defrag
------

test> use AAC
switched to db AAC
AAC> db.animals.createIndex({ breed: 1 })
breed_1
AAC> db.animals.find({ breed: "Labrador Retriever" }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'AAC.animals',
    indexFilterSet: false,
    parsedQuery: { breed: { '$eq': 'Labrador Retriever' } },
    queryHash: '17252B62',
    planCacheKey: '8F94FD79',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { breed: 1 },
        indexName: 'breed_1',
        isMultiKey: false,
        multiKeyPaths: { breed: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { breed: [ '["Labrador Retriever", "Labrador Retriever"]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 18,
    executionTimeMillis: 0,
    totalKeysExamined: 18,
    totalDocsExamined: 18,
    executionStages: {
      stage: 'FETCH',
      nReturned: 18,
      executionTimeMillisEstimate: 0,
      works: 19,
      advanced: 18,
      needTime: 0,
      needYield: 0
```

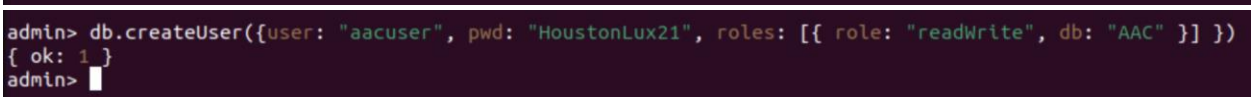
Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.



Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.



Note: This template has been adapted from the following sample templates: [Make a README], [Best README Template], and [A Beginners Guide to Writing a Kickass README].



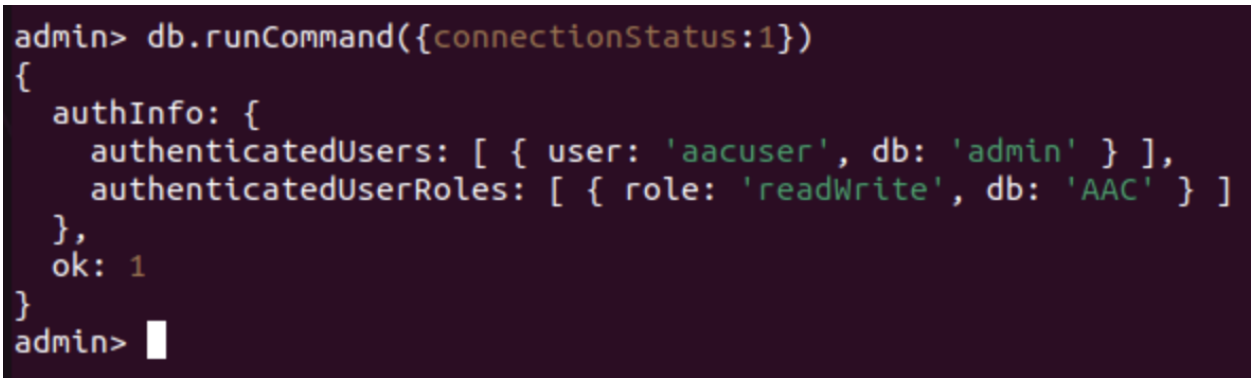
Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
admin> use AAC
switched to db AAC
AAC> db.animals.findOne()
{
  _id: ObjectId("6796a38525507d513b55ebd8"),
  rec_num: 3,
  age_upon_outcome: '2 years',
  animal_id: 'A716330',
  animal_type: 'Dog',
  breed: 'Chihuahua Shorthair Mix',
  color: 'Brown/White',
  date_of_birth: '2013-11-18',
  datetime: '2015-12-28 18:43:00',
  monthyear: '2015-12-28T18:43:00',
  name: 'Frank',
  outcome_subtype: '',
  outcome_type: 'Adoption',
  sex_upon_outcome: 'Neutered Male',
  location_lat: 30.7595748121648,
  location_long: -97.5523753807133,
  age_upon_outcome_in_weeks: 110.111408730159
}
AAC>
```

```
(base) derrickoswink_snhu@nv-snhu8-l04:~$ mongosh
Current Mongosh Log ID: 6796ace63b96ea3ef3108206
Connecting to:          mongodb://<credentials>@nv-desktop-services.apporto.com:31448/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:          6.0.13
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-01-26T20:40:02.610+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongo
db.org/core/prodnotes-filesystem
   2025-01-26T20:40:04.735+00:00: Failed to read /sys/kernel/mm/transparent_hugepage/defrag
------

test> use admin
switched to db admin
admin> db.createUser({ user: "accuser", pwd: "HoustonLux21", roles: [{ role: "readWrite", db: "AAC" }] })
{ ok: 1 }
admin>
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
AAC> db.animals.createIndex({ breed: 1, outcome_type: 1 })
breed_1_outcome_type_1
AAC> db.animals.find({ breed: "Labrador Retriever", outcome_type: "Transfer" }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'AAC.animals',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [
        { breed: { '$eq': 'Labrador Retriever' } },
        { outcome_type: { '$eq': 'Transfer' } }
      ]
    },
    queryHash: 'E7C1F5D3',
    planCacheKey: '653FF755',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { breed: 1, outcome_type: 1 },
        indexName: 'breed_1_outcome_type_1',
        isMultiKey: false,
        multiKeyPaths: { breed: [], outcome_type: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          breed: [ '["Labrador Retriever", "Labrador Retriever"]' ],
          outcome_type: [ '["Transfer", "Transfer"]' ]
        }
      }
    },
    rejectedPlans: [
      {
        stage: 'FETCH',
        filter: { outcome_type: { '$eq': 'Transfer' } },
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { breed: 1 },
          indexName: 'breed_1',
          isMultiKey: false,
          multiKeyPaths: { breed: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { breed: [ '["Labrador Retriever", "Labrador Retriever"]' ] }
        }
      }
    ]
  },
  executionStats: {
    executionSuccess: true,
```



```
admin> db.runCommand({connectionStatus:1})
{
  authInfo: {
    authenticatedUsers: [ { user: 'aacuser', db: 'admin' } ],
    authenticatedUserRoles: [ { role: 'readWrite', db: 'AAC' } ]
  },
  ok: 1
}
admin>
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Spyder

The create() function inserts a new document into the MongoDB collection. If the document already exists, MongoDB will automatically generate a unique ID for the new entry.

```
def create(self, data):
    """Insert a new document into the animals collection."""
    if not data:
        raise ValueError("Data cannot be empty.")
    try:
        result = self.collection.insert_one(data)
        return bool(result.inserted_id)
    except errors.PyMongoError as e:
        print(f"Error inserting data into MongoDB: {e}")
        return False
```

The read() function retrieves documents from the collection based on a query filter. If no filter is provided, it returns all documents in the collection.

```
def read(self, query):
    """Query documents from the animals collection."""
    try:
        results = self.collection.find(query)
        return list(results)
    except errors.PyMongoError as e:
        print(f"Error reading from database: {e}")
        return []
```

The update() function modifies an existing document that matches the specified filter. It uses MongoDB's update operators (e.g., $set) to define the modifications.

```
def update(self, query, new_values):
    """Update documents in the animals collection."""
    if not query or not new_values:
        raise ValueError("Query and new values cannot be empty.")
    try:
        result = self.collection.update_many(query, {"$set": new_values})
        return result.modified_count
    except errors.PyMongoError as e:
        print(f"Error updating data in MongoDB: {e}")
        return 0
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

The delete() function removes documents that match the provided query filter from the collection.

```
def delete(self, query):
    """Delete documents from the animals collection."""
    if not query:
        raise ValueError("Query cannot be empty.")
    try:
        result = self.collection.delete_many(query)
        return result.deleted_count
    except errors.PyMongoError as e:
        print(f"Error deleting data from MongoDB: {e}")
        return 0
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```python
class AnimalShelter:
    """CRUD operations for the Animal collection in MongoDB."""

    def __init__(self):
        """Initialize MongoDB connection."""
        # Connection Variables
        self._user = "aacuser"
        self._password = "HoustonLux21"
        self._host = "nv-desktop-services.apporto.com"
        self._port = 31448
        self._database_name = "AAC"
        self._collection_name = "animals"

        # Establish MongoDB connection
        try:
            self.client = MongoClient(f"mongodb://{self._user}:{self._password}@{self._host}:{self._port}")
            self.database = self.client[self._database_name]
            self.collection = self.database[self._collection_name]
            print("MongoDB connection successful.")
        except errors.ConnectionFailure as e:
            print(f"MongoDB connection failed: {e}")
            raise

    def create(self, data):
        """Insert a new document into the animals collection."""
        if not data:
            raise ValueError("Data cannot be empty.")
        try:
            result = self.collection.insert_one(data)
            return bool(result.inserted_id)
        except errors.PyMongoError as e:
            print(f"Error inserting data into MongoDB: {e}")
            return False

    def read(self, query):
        """Query documents from the animals collection."""
        try:
            results = self.collection.find(query)
            return list(results)
        except errors.PyMongoError as e:
            print(f"Error reading from database: {e}")
            return []

    def update(self, query, new_values):
        """Update documents in the animals collection."""
        if not query or not new_values:
            raise ValueError("Query and new values cannot be empty.")
        try:
            result = self.collection.update_many(query, {"$set": new_values})
            return result.modified_count
        except errors.PyMongoError as e:
            print(f"Error updating data in MongoDB: {e}")
            return 0

    def delete(self, query):
        """Delete documents from the animals collection."""
        if not query:
            raise ValueError("Query cannot be empty.")
        try:
            result = self.collection.delete_many(query)
            return result.deleted_count
        except errors.PyMongoError as e:
            print(f"Error deleting data from MongoDB: {e}")
            return 0
```

This CRUD Python module serves as a powerful tool for interacting with MongoDB. It covers the most common operations (Create, Read, Update, and Delete) and is easily extendable for more complex scenarios. With the help of the pymongo library, developers can easily (and effectively) integrate MongoDB seamlessly into their Python applications for efficient data management.

Tests

Jupyter Notebook:

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Import Dependencies

1. Open a new Jupyter Notebook.
2. Import the AnimalShelter class.
3. Instantiate the AnimalShelter class.

```
In [1]: from animalShelter import AnimalShelter
```

```
In [2]: # Instantiate the AnimalShelter class
        shelter = AnimalShelter()

        MongoDB connection successful.
```

```
In [3]: # Test Data
        test_animal = {
            "name": "Buddy",
            "species": "Dog",
            "breed": "Labrador Retriever",
            "age": 3,
            "weight": 50,
            "adopted": False
        }
```

Test the create Method

1. Define a dictionary with sample animal data.
2. Call the create method to insert the data into MongoDB.
3. Check if the insertion was successful and print a confirmation message.

```
In [4]: # Create - Insert a new animal document
        print("Inserting new animal...")
        create_result = shelter.create(test_animal)
        print(f"Create operation success: {create_result}\n")

        Inserting new animal...
        Create operation success: True
```

Test the read Method

1. Define a query to search for the inserted animal by name.
2. Call the read method to retrieve the data.
3. Check if any results were found.
4. Print the retrieved data.

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
In [5]: # Read - Retrieve inserted animal
print("Reading inserted animal...")
query = {"name": "Buddy"}
read_result = shelter.read(query)
print(f"Read operation result: {read_result}\n")
```

'datetime': '2017-07-20 11:48:00', 'monthyear': '2017-07-20T11:48:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Return to Owner', 'sex_upon_outcome': 'Intact Male', 'location_lat': 30.499684557721, 'location_long': -97.4484573088344, 'age_upon_outcome_in_weeks': 52.4988095238095}, {'_id': ObjectId('6796a38525507d513b560bf8'), 'rec_num': 8212, 'age_upon_outcome': '8 years', 'animal_id': 'A566493', 'animal_type': 'Dog', 'breed': 'Labrador Retriever/Mastiff', 'color': 'Cream/Yellow', 'date_of_birth': '2009-02-11', 'datetime': '2017-06-16 10:46:00', 'monthyear': '2017-06-16T10:46:00', 'name': 'Buddy', 'outcome_subtype': 'Foster', 'outcome_type': 'Adoption', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.735400849025, 'location_long': -97.4733007367498, 'age_upon_outcome_in_weeks': 435.349801587302}, {'_id': ObjectId('6796a38525507d513b56115e'), 'rec_num': 9596, 'age_upon_outcome': '1 month', 'animal_id': 'A698556', 'animal_type': 'Dog', 'breed': 'Labrador Retriever/Boxer', 'color': 'Black/White', 'date_of_birth': '2015-03-11', 'datetime': '2015-05-09 12:29:00', 'monthyear': '2015-05-09T12:29:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Adoption', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.6905858494311, 'location_long': -97.7346046107505, 'age_upon_outcome_in_weeks': 8.50287698412698}, {'_id': ObjectId('6796a38525507d513b561297'), 'rec_num': 9923, 'age_upon_outcome': '1 year', 'animal_id': 'A705566', 'animal_type': 'Dog', 'breed': 'Boxer/Bulldog', 'color': 'Brown/White', 'date_of_birth': '2014-06-19', 'datetime': '2015-06-23 17:14:00', 'monthyear': '2015-06-23T17:14:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Adoption', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.4049435962897, 'location_long': -97.5316179632121, 'age_upon_outcome_in_weeks': 52.8168650793651}, {'_id': ObjectId('6796a7e85285e097f5cffe3e'), 'rec_num': 724, 'age_upon_outcome': '4 months', 'animal_id': 'A675568', 'animal_type': 'Dog', 'breed': 'Chihuahua Shorthair Mix', 'color': 'Brown/White', 'date_of_birth': '2013-11-18', 'datetime': '2014-04-03 17:53:00', 'monthyear': '2014-04-03T17:53:00', 'name': 'Buddy', 'outcome_subtyp

```
In [6]: # Update - Modify animal's adoption status
print("Updating adoption status...")
update_query = {"name": "Buddy"}
new_values = {"adopted": True}
update_result = shelter.update(update_query, new_values)
print(f"Number of documents updated: {update_result}\n")
```

```
Updating adoption status...
Number of documents updated: 32
```

Run All Tests

1. Execute all the cells in Jupyter Notebook sequentially.
2. Ensure MongoDB is running and accessible.
3. Verify that both insertion and retrieval work correctly.

Screenshots

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Southern New Hampshire University

In [1]:
```
from animalShelter import AnimalShelter
```

In [2]:
```
# Instantiate the AnimalShelter class
shelter = AnimalShelter()
```

```
MongoDB connection successful.
```

In [3]:
```
# Test Data
test_animal = {
    "name": "Buddy",
    "species": "Dog",
    "breed": "Labrador Retriever",
    "age": 3,
    "weight": 50,
    "adopted": False
}
```

In [4]:
```
# Create - Insert a new animal document
print("Inserting new animal...")
create_result = shelter.create(test_animal)
print(f"Create operation success: {create_result}\n")
```

```
Inserting new animal...
Create operation success: True
```

In [5]:
```
# Read - Retrieve inserted animal
print("Reading inserted animal...")
query = {"name": "Buddy"}
read_result = shelter.read(query)
print(f"Read operation result: {read_result}\n")
```


```
Reading inserted animal...
Read operation result: [{'_id': ObjectId('6796a38525507d513b55eea9'), 'rec_num': 724, 'age_upon_outcome': '4 months
', 'animal_id': 'A675568', 'animal_type': 'Dog', 'breed': 'Chihuahua Shorthair Mix', 'color': 'Brown/White', 'date_o
f_birth': '2013-11-18', 'datetime': '2014-04-03 17:53:00', 'monthyear': '2014-04-03T17:53:00', 'name': 'Buddy', 'out
come_subtype': '', 'outcome_type': 'Adoption', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.708106432952,
'location_long': -97.3499268421422, 'age_upon_outcome_in_weeks': 19.5350198412698}, {'_id': ObjectId('6796a38525507d
513b55ef27'), 'rec_num': 842, 'age_upon_outcome': '6 years', 'animal_id': 'A736590', 'animal_type': 'Dog', 'breed':
'Jack Russell Terrier Mix', 'color': 'White/Brown', 'date_of_birth': '2010-01-30', 'datetime': '2016-11-05 13:53:00
', 'monthyear': '2016-11-05T13:53:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Adoption', 'sex_upon
_outcome': 'Neutered Male', 'location_lat': 30.466259071107, 'location_long': -97.5483512881312, 'age_upon_outcome_i
n_weeks': 353.082638888889}, {'_id': ObjectId('6796a38525507d513b55f1e4'), 'rec_num': 1549, 'age_upon_outcome': '3 m
onths', 'animal_id': 'A731554', 'animal_type': 'Cat', 'breed': 'Domestic Shorthair Mix', 'color': 'Black', 'date_of_
birth': '2016-05-08', 'datetime': '2016-08-13 12:27:00', 'monthyear': '2016-08-13T12:27:00', 'name': 'Buddy', 'outco
me_subtype': 'Partner', 'outcome_type': 'Transfer', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.45175157
24453, 'location_long': -97.7272941918411, 'age_upon_outcome_in_weeks': 13.93125}, {'_id': ObjectId('6796a38525507d5
13b55f3fe'), 'rec_num': 2092, 'age_upon_outcome': '7 years', 'animal_id': 'A697660', 'animal_type': 'Dog', 'breed':
'Staffordshire/Bull Terrier', 'color': 'White/Black', 'date_of_birth': '2008-02-26', 'datetime': '2015-05-16 16:46:0
0', 'monthyear': '2015-05-16T16:46:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Return to Owner', '
sex_upon_outcome': 'Neutered Male', 'location_lat': 30.3793418287565, 'location_long': -97.309496960952, 'age_upon_o
```


Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
In [6]: # Update - Modify animal's adoption status
print("Updating adoption status...")
update_query = {"name": "Buddy"}
new_values = {"adopted": True}
update_result = shelter.update(update_query, new_values)
print(f"Number of documents updated: {update_result}\n")
```

```
Updating adoption status...
Number of documents updated: 32
```

```
In [7]: # Read - Verify update
print("Verifying update...")
read_updated = shelter.read(query)
print(f"Updated record: {read_updated}\n")
```

```
Verifying update...
Updated record: [{'_id': ObjectId('6796a38525507d513b55eea9'), 'rec_num': 724, 'age_upon_outcome': '4 months', 'anim
al_id': 'A675568', 'animal_type': 'Dog', 'breed': 'Chihuahua Shorthair Mix', 'color': 'Brown/White', 'date_of_birth
': '2013-11-18', 'datetime': '2014-04-03 17:53:00', 'monthyear': '2014-04-03T17:53:00', 'name': 'Buddy', 'outcome_su
btype': '', 'outcome_type': 'Adoption', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.708106432952, 'locat
ion_long': -97.3499268421422, 'age_upon_outcome_in_weeks': 19.5350198412698, 'adopted': True}, {'_id': ObjectId('679
6a38525507d513b55ef27'), 'rec_num': 842, 'age_upon_outcome': '6 years', 'animal_id': 'A736590', 'animal_type': 'Dog
', 'breed': 'Jack Russell Terrier Mix', 'color': 'White/Brown', 'date_of_birth': '2010-01-30', 'datetime': '2016-11-
05 13:53:00', 'monthyear': '2016-11-05T13:53:00', 'name': 'Buddy', 'outcome_subtype': '', 'outcome_type': 'Adoption
', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.466259071107, 'location_long': -97.5483512881312, 'age_up
on_outcome_in_weeks': 353.082638888889, 'adopted': True}, {'_id': ObjectId('6796a38525507d513b55f1e4'), 'rec_num': 1
549, 'age_upon_outcome': '3 months', 'animal_id': 'A731554', 'animal_type': 'Cat', 'breed': 'Domestic Shorthair Mix
', 'color': 'Black', 'date_of_birth': '2016-05-08', 'datetime': '2016-08-13 12:27:00', 'monthyear': '2016-08-13T12:2
7:00', 'name': 'Buddy', 'outcome_subtype': 'Partner', 'outcome_type': 'Transfer', 'sex_upon_outcome': 'Neutered Male
', 'location_lat': 30.4517515724453, 'location_long': -97.7272941918411, 'age_upon_outcome_in_weeks': 13.93125, 'ado
pted': True}, {'_id': ObjectId('6796a38525507d513b55f3fe'), 'rec_num': 2092, 'age_upon_outcome': '7 years', 'animal_
id': 'A697660', 'animal_type': 'Dog', 'breed': 'Staffordshire/Bull Terrier', 'color': 'White/Black', 'date_of_birth
': '2008-02-26', 'datetime': '2015-05-16 16:46:00', 'monthyear': '2015-05-16T16:46:00', 'name': 'Buddy', 'outcome_su
btype': '', 'outcome_type': 'Return to Owner', 'sex_upon_outcome': 'Neutered Male', 'location_lat': 30.379341828756
```

```
In [8]: # Delete - Remove the test animal
print("Deleting the test animal...")
delete_result = shelter.delete(query)
print(f"Number of documents deleted: {delete_result}\n")
```

```
Deleting the test animal...
Number of documents deleted: 32
```

```
In [9]: # Read - Confirm deletion
print("Confirming deletion...")
final_read = shelter.read(query)
print(f"Read after deletion (should be empty): {final_read}\n")
```

```
Confirming deletion...
Read after deletion (should be empty): []
```

Roadmap/Features (Optional)

Provide an open issues list of proposed features (and known issues). If you have ideas for releases in the future, it is a good idea to list them in the README. What makes your project stand out?

Challenges Faced

1. MongoDB Connection Issues – Encountered authentication failures.
    a. Solution: Ensured correct connection string and allowed IP access.
2. Ensuring Data Validation – Some records lacked required fields.
    a. Solution: Added error handling and validation before inserting records.

Future Enhancements

3. Add a GUI interface using Tkinter or PyQt.

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

4. Implement bulk data upload from CSV.
5. Integrate error handling and logging for debugging.

Contact
Derricko Swink

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.