5-2 Milestone Four: Enhancement Three: Databases

Victoria Franklin

Southern New Hampshire University

CS-499 Computer Science Capstone

Professor Gene Bryant

Sunday, August 4, 2024

CS 499 Milestone Four

I chose to enhance the AnimalShelter.py artifact, one of the course's final assignments. The AnimalShelter class in MongoDB performs CRUD operations on a collection of animals. This code includes initializing the connection to the MongoDB database and implementing the read, update, delete, and create methods. It is possible to improve the code by making several changes. The init method takes a username and password as inputs and passes them to the connection variables, ensuring secure access to the database. The deprecated insert_one was replaced with the new method creation, improving the efficiency and performance of the code. A list of documents is returned using the read method in addition to printing them, enhancing the readability of the code. The method update guaranteed consistency by including a return statement, which is crucial for data integrity. There is now a return statement in the Delete Method for consistency, making the code more predictable. This enhancement request fortifies the AnimalShelter class to deal with problems more effectively and provide more comprehensive logging, making debugging and maintenance easier.

The AnimalShelter.py artifact is not just a significant addition to my ePortfolio but a testament to the depth of my expertise in database management, CRUD operations, secure coding methods, and other crucial areas essential for a software engineering and development project. Its inclusion reflects my skills and knowledge, as detailed below.

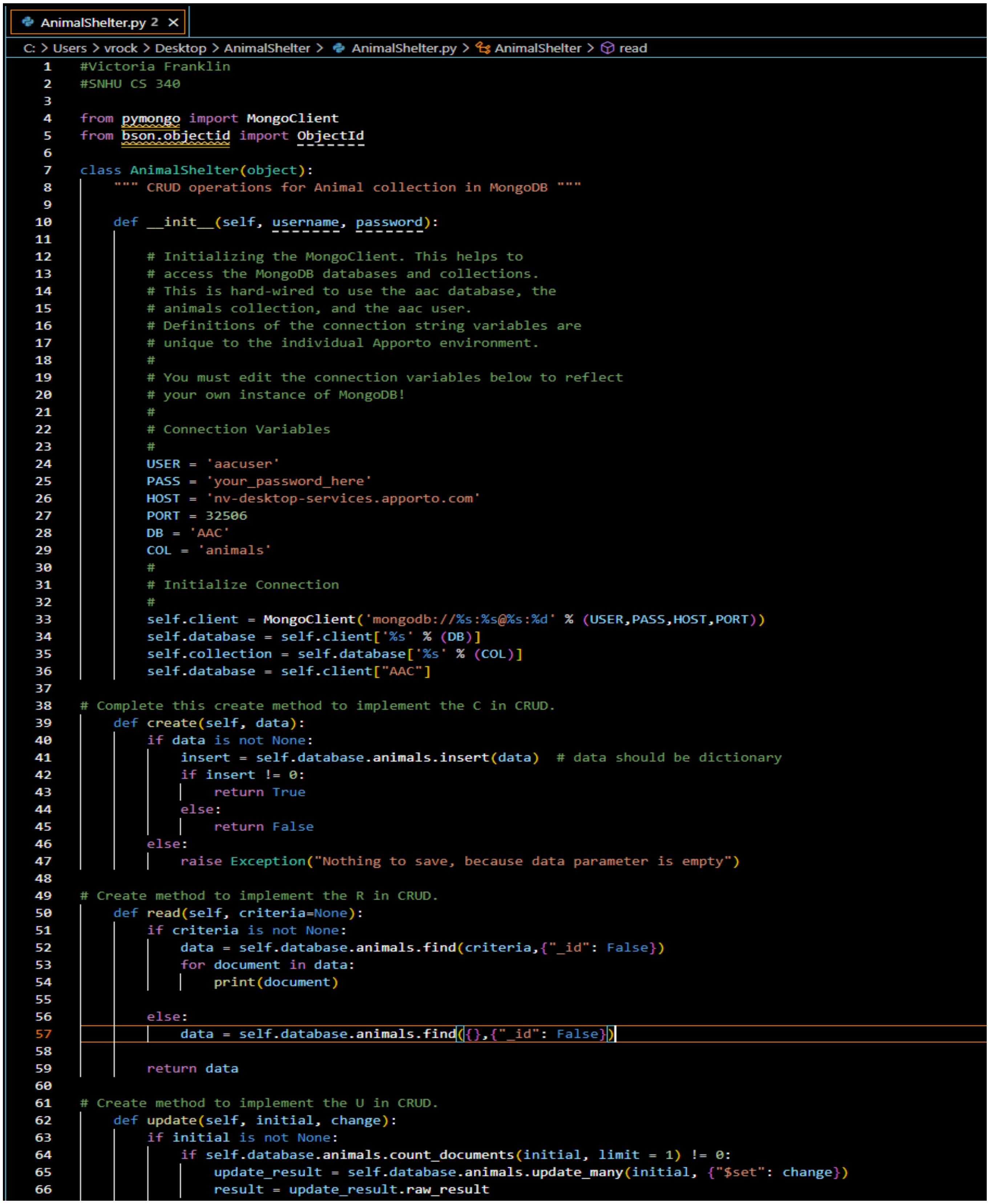
- **Create:** Update the database with additional documents.
- **Read:** Retrieve documents based on specific criteria.
- **Update:** Modify existing documents.
- **Delete:** Remove documents from the database.

Using MongoDB, a well-known NoSQL database, showcases my proficiency with modern database technologies. That is particularly true when considering future technological developments and system designs. Credentials like usernames and passwords are securely managed while connecting to databases, a skill I homed in CS 405: Code Security, which emphasizes safe coding and credential management methods. In case of a fundamental error, such as trying to execute an action on null data, the script will immediately resolve the situation. For software development to be effective, this is paramount. In keeping with object-oriented programming principles, the script uses an AnimalShelter class to contain the database operations. With its scalable and easy-to-maintain modular architecture, the class code paves the way for future modifications or expansions. This design issue is crucial for scalable system design and aligns with the talents I aim to showcase in my ePortfolio.

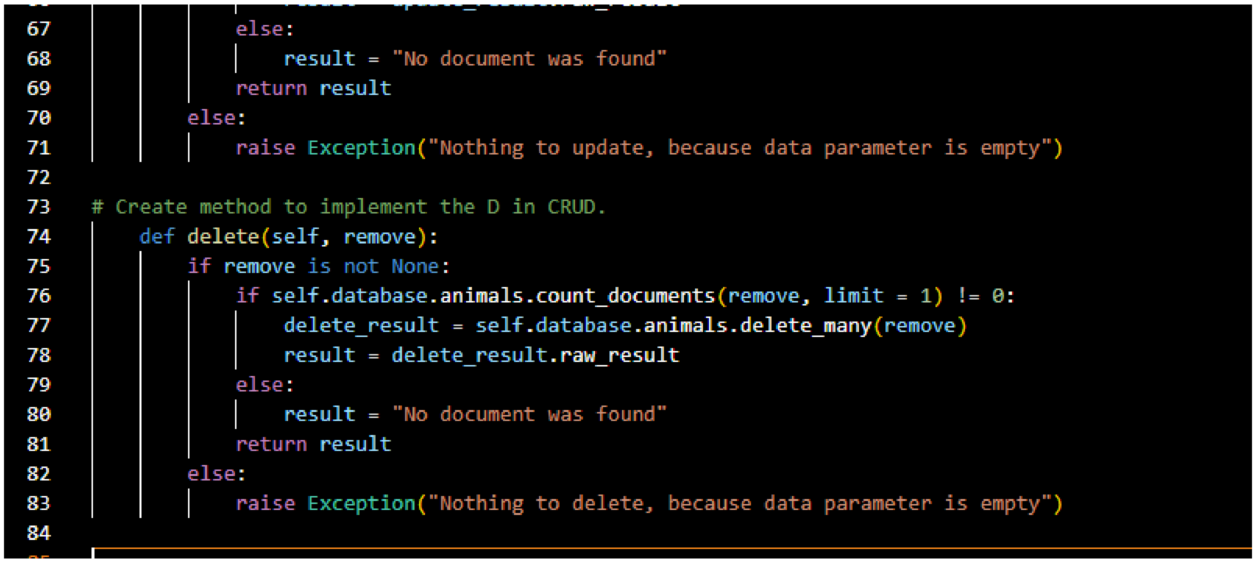
Incorporating AnimalShelter.py into my ePortfolio showcases my software engineering skills and demonstrates my ability to develop and construct a fully operational backend service. This completes the software engineering and design artifact requirements of my CS 499 project. In conclusion, AnimalShelter.py showcases my proficiency in managing databases, coding securely, understanding object-oriented design, and using advanced programming concepts in real-world scenarios. This section of my ePortfolio is crucial because it showcases the knowledge and abilities I have acquired via my classes and activities.

I ensure that my work aligns with the course goals by updating my outcome coverage plans to reflect the improvements made using the AnimalShelter.py software. This alignment enhances the security and reliability of my software systems and makes my ePortfolio even more impressive. It demonstrates that I have mastered contemporary database administration techniques, robust software design, and safe coding. Through these updates, I gained a better grasp of secure coding, database management, and software design concepts, and I honed my technical abilities by improving and altering the AnimalShelter.py artifact. My difficulties allowed me to hone my problem-solving skills and incorporate industry best practices, resulting in a more robust and safer implementation. This method has also highlighted the significance of iterative and continuous improvement in software development, ensuring that the systems we create are dependable and easy to manage.

Original AnimalShelter.py

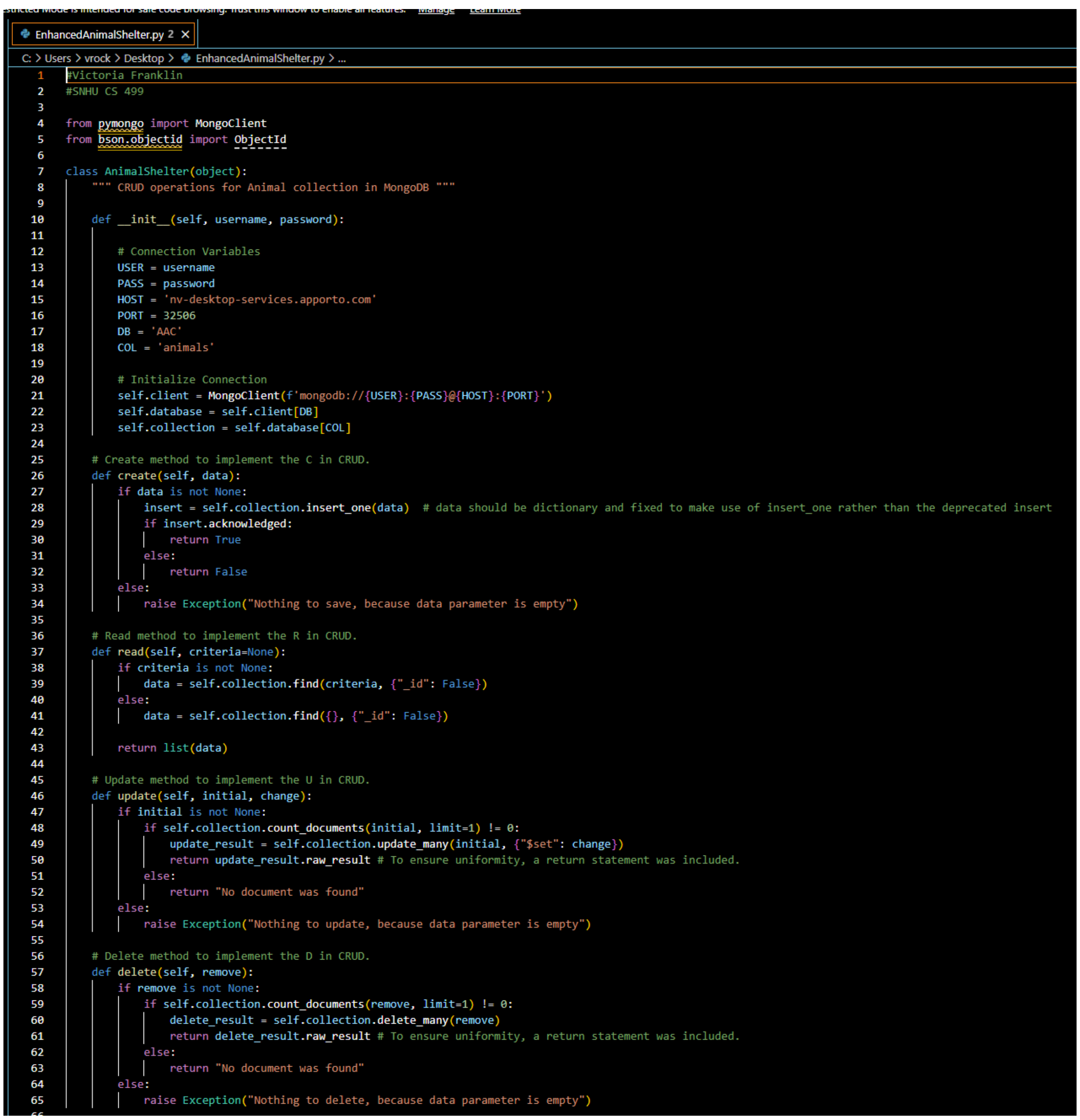

AnimalShelter.py 2
C: > Users > vrock > Desktop > AnimalShelter > AnimalShelter.py > AnimalShelter > read

```python
#Victoria Franklin
#SNHU CS 340

from pymongo import MongoClient
from bson.objectid import ObjectId

class AnimalShelter(object):
    """ CRUD operations for Animal collection in MongoDB """

    def __init__(self, username, password):

        # Initializing the MongoClient. This helps to
        # access the MongoDB databases and collections.
        # This is hard-wired to use the aac database, the
        # animals collection, and the aac user.
        # Definitions of the connection string variables are
        # unique to the individual Apporto environment.
        #
        # You must edit the connection variables below to reflect
        # your own instance of MongoDB!
        #
        # Connection Variables
        #
        USER = 'aacuser'
        PASS = 'your_password_here'
        HOST = 'nv-desktop-services.apporto.com'
        PORT = 32506
        DB = 'AAC'
        COL = 'animals'
        #
        # Initialize Connection
        #
        self.client = MongoClient('mongodb://%s:%s@%s:%d' % (USER,PASS,HOST,PORT))
        self.database = self.client['%s' % (DB)]
        self.collection = self.database['%s' % (COL)]
        self.database = self.client["AAC"]

# Complete this create method to implement the C in CRUD.
    def create(self, data):
        if data is not None:
            insert = self.database.animals.insert(data)  # data should be dictionary
            if insert != 0:
                return True
            else:
                return False
        else:
            raise Exception("Nothing to save, because data parameter is empty")

# Create method to implement the R in CRUD.
    def read(self, criteria=None):
        if criteria is not None:
            data = self.database.animals.find(criteria,{"_id": False})
            for document in data:
                print(document)

        else:
            data = self.database.animals.find({},{"_id": False})

        return data

# Create method to implement the U in CRUD.
    def update(self, initial, change):
        if initial is not None:
            if self.database.animals.count_documents(initial, limit = 1) != 0:
                update_result = self.database.animals.update_many(initial, {"$set": change})
                result = update_result.raw_result
```



```
            else:
                result = "No document was found"
            return result
        else:
            raise Exception("Nothing to update, because data parameter is empty")

# Create method to implement the D in CRUD.
    def delete(self, remove):
        if remove is not None:
            if self.database.animals.count_documents(remove, limit = 1) != 0:
                delete_result = self.database.animals.delete_many(remove)
                result = delete_result.raw_result
            else:
                result = "No document was found"
            return result
        else:
            raise Exception("Nothing to delete, because data parameter is empty")
```

Enhanced AnimalShelter.py


stricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
EnhancedAnimalShelter.py 2
C: > Users > vrock > Desktop > EnhancedAnimalShelter.py > ...

```python
#Victoria Franklin
#SNHU CS 499

from pymongo import MongoClient
from bson.objectid import ObjectId

class AnimalShelter(object):
    """ CRUD operations for Animal collection in MongoDB """

    def __init__(self, username, password):

        # Connection Variables
        USER = username
        PASS = password
        HOST = 'nv-desktop-services.apporto.com'
        PORT = 32506
        DB = 'AAC'
        COL = 'animals'

        # Initialize Connection
        self.client = MongoClient(f'mongodb://{USER}:{PASS}@{HOST}:{PORT}')
        self.database = self.client[DB]
        self.collection = self.database[COL]

    # Create method to implement the C in CRUD.
    def create(self, data):
        if data is not None:
            insert = self.collection.insert_one(data)  # data should be dictionary and fixed to make use of insert_one rather than the deprecated insert
            if insert.acknowledged:
                return True
            else:
                return False
        else:
            raise Exception("Nothing to save, because data parameter is empty")

    # Read method to implement the R in CRUD.
    def read(self, criteria=None):
        if criteria is not None:
            data = self.collection.find(criteria, {"_id": False})
        else:
            data = self.collection.find({}, {"_id": False})

        return list(data)

    # Update method to implement the U in CRUD.
    def update(self, initial, change):
        if initial is not None:
            if self.collection.count_documents(initial, limit=1) != 0:
                update_result = self.collection.update_many(initial, {"$set": change})
                return update_result.raw_result # To ensure uniformity, a return statement was included.
            else:
                return "No document was found"
        else:
            raise Exception("Nothing to update, because data parameter is empty")

    # Delete method to implement the D in CRUD.
    def delete(self, remove):
        if remove is not None:
            if self.collection.count_documents(remove, limit=1) != 0:
                delete_result = self.collection.delete_many(remove)
                return delete_result.raw_result # To ensure uniformity, a return statement was included.
            else:
                return "No document was found"
        else:
            raise Exception("Nothing to delete, because data parameter is empty")
```