**SNHU Project: Animal Shelter CRUD Application - Summary of Issues and Solutions**

**Project Overview**:  
The application was designed to manage animal shelter records through MongoDB using Python in Jupyter Notebook. The main objective was to implement CRUD (Create, Read, Update, Delete) operations to add, search, and delete animal data.

**Challenges and Solutions**

1. **Network Disruption & MongoDB Connection Issues**  
   **Issue**: The connection to MongoDB was frequently disrupted, particularly when using **MongoDB on Apporto** via **Mongosh** or Jupyter's Python environment.

**Solution**:

* + Ensured that the correct connection string with proper authentication was used.
  + Verified that the MongoDB server was up and running correctly before reconnecting.
  + Used **Firefox** to open Jupyter notebooks in case other browsers faced issues with displaying content or connecting to the server.

1. **Data Syntax & Code Errors**  
   **Issue**: Encountered multiple syntax errors, including **missing semi-colons**, incorrect indenting, and **data syntax errors** in Python and MongoDB commands.

**Solution**:

* + Carefully checked Python and MongoDB scripts for syntax issues like missing **semi-colons** (though Python does not require them, **correct indentation** is crucial).
  + Utilized Python's try-except blocks to handle exceptions gracefully and identify any issues during runtime.
  + Employed **PyCharm** and **VS Code** to spot syntax issues early by enabling linting and automatic suggestions.

1. **Data Insertion & CRUD Operations (Create & Read)**  
   **Issue**: There were issues with **data insertion** not returning the expected **True/False** values, and certain commands did not execute as expected.

**Solution**:

* + The **create** method was tested using various animal data, but it returned False when it should have returned True upon successful insertion.
  + Ensured that data passed into the **insert\_one()** method was a properly formatted dictionary.
  + **read** method results were tested using the **query** for a pet's name. If no pet was found, it returned **False** (expected behavior).
  + Corrected the code to check for **non-empty query results**, which could ensure that the expected **True/False** logic functioned properly.

1. **Switching Between MongoDB, Mongosh, and Jupyter Notebook**  
   **Issue**: Switching between different environments, such as **MongoDB terminal (Mongosh)** and **Jupyter terminal**, led to confusion and occasional **connection errors** when querying the database.

**Solution**:

* + Standardized the workflow by ensuring all MongoDB operations were either fully executed in **Mongosh** or handled within the **Jupyter Notebook** for consistency.
  + Used **Jupyter terminal** to interact with MongoDB when necessary but verified all terminal commands were accurately executed in **Mongosh** to avoid discrepancies.
  + Documented the necessary **Mongosh commands** like **use AAC**, **show collections**, and **db.animals.find()**.

1. **Running & Testing Code with Runtime Errors**  
   **Issue**: Encountered **runtime errors** that hindered code execution. Even though the program was expected to return **True** upon inserting new records, sometimes the result would show **False**.

**Solution**:

* + Ensured the correct syntax for **MongoDB queries** and reviewed Python scripts to ensure proper method calls (e.g., **insert\_one()**, **find()**).
  + Used **print statements** and **logging** to identify where the program failed and debug issues, especially with **MongoDB** connections and queries.
  + Corrected issues by ensuring **valid data formats** were passed into MongoDB operations.

1. **Retrieving Files from Apporto & Brightspace**  
   **Issue**: Difficulty retrieving files from **Apporto** and uploading them to **Brightspace** as part of the assignment submission.

**Solution**:

* + Exported **Jupyter Notebooks** as **.html** or **.py** files.
  + Used **Firefox** for better compatibility when opening **Jupyter Notebook** on **Apporto**.
  + Uploaded necessary files to **Apporto Library** and confirmed that the correct **AAC** folder was selected for submission.

1. **MongoDB Indexing Issues**  
   **Issue**: Struggled with **indexing** MongoDB collections to allow efficient querying, resulting in slower performance for large datasets.

**Solution**:

* + Used **both simple and complex indexes** to optimize queries, ensuring data retrieval was faster.
  + Managed the creation of indexes within **MongoDB shell** using commands like **db.animals.createIndex()**.

1. **Project Organization and File Management**  
   **Issue**: Difficulty with **organizing code** and keeping track of **test files** required to test CRUD operations across **.py**, **.pynb**, and **.html** files.

**Solution**:

* + Created separate **.py** files for backend logic and CRUD operations.
  + Structured the Jupyter notebooks with comments and sectioning to avoid confusion between different methods and operations.
  + Used **version control** (Git) to track changes across different files and projects, improving code management.

**Final Outcome**

By overcoming these challenges, I successfully developed the CRUD application for managing animal shelter records. Key outcomes include:

* **MongoDB database** setup and proper integration with Python.
* Implementation of CRUD operations, which allowed users to add and search for animal records.
* Successfully managed the technical environment, including **Apporto**, **MongoDB Terminal**, and **Jupyter Notebook**.
* The project reinforced the importance of **debugging**, **testing**, and **persistence** when learning new technologies and software development practices.

**Contact**

* **Name**: Marissa Lanza
* **Email**: marissa.lanza@snhu.edu
* **GitHub**: <https://github.com/LiXiu37YaHua>

Let me know if you need further adjustments or have more questions!

4o mini

**You said:**

success, reflection, learned alot

**ChatGPT said:**

ChatGPT

**Reflection: Success and Learning Experience**

This project was a significant challenge, and I'm proud to have overcome the hurdles I faced throughout the process. Initially, I encountered several obstacles, such as network disruptions and MongoDB connection issues, which were frustrating but taught me a lot about troubleshooting and persistence. By the end of the project, I gained a deeper understanding of how databases interact with Python and how to set up and manage MongoDB collections, which was something I hadn't fully grasped before.

I learned a great deal about the **CRUD operations** (Create, Read, Update, Delete) and how to effectively implement them using MongoDB and Python. While I struggled at times with the syntax and understanding the relationship between different components, like MongoDB, Python, and Jupyter notebooks, I was able to break down the challenges and resolve them one by one.

The experience was not just technical but also personal. I faced moments of frustration and imposter syndrome, but I pushed through by relying on resources like YouTube tutorials and forums, as well as my own problem-solving skills. The feeling of finally seeing the expected **True** output after hours of troubleshooting was incredibly rewarding. It reinforced the idea that persistence pays off, and it was a reminder that learning and growth often come from the most difficult challenges.

From this project, I also learned how to organize and manage my code better, how to handle errors, and how to keep my project files structured, making it easier to work with them in different environments. I now feel much more confident in my ability to work with databases and Python, and I look forward to applying these skills to future projects.

Ultimately, this project was a huge success because I not only completed it but also learned valuable lessons in both technical and personal aspects of software development. It gave me the confidence to keep moving forward in my software engineering journey, and I'm excited to see what I can accomplish next.