**Milestone Four Enhancements**

**Description of the Artifact**

The artifact I selected for this milestone is the database interaction layer of my project, which specifically involves CRUD operations and data handling with MongoDB. This artifact originated from my work in CS 340, where I initially built an application to manage animal shelter data using MongoDB as the backend database. The original code included basic CRUD operations to create, read, update, and delete records in the animal\_shelter.py file, but it lacked meaningful data validation features.

**Justification for Inclusion in My ePortfolio**

I chose this artifact for inclusion in my ePortfolio because it showcases my proficiency in managing database interactions, validating data, and implementing secure data handling practices. These skills that are essential for a computer science professional. The enhancements to this artifact specifically highlight my ability to improve database reliability, security, and integrity. In particular, by adding data validation and sanitization functions, I demonstrate my commitment to maintaining high standards in data quality and security.

These enhancements align with my career goal of working with data-centric applications where data integrity and security are critical. Including this enhanced artifact in my ePortfolio, I can showcase my expertise in database management and my understanding of security practices, both of which are skills valued by employers in the field.

**Achievement of Course Outcomes**

To achieve the required course outcomes, I added a function to my animal\_shelter.py file that is used to validate data before being inputted into the database.

First, I had to establish data types for the fields being altered, in this case str, float and int:

A screen shot of a computer program

Description automatically generated

Next, I created constraints for each data field type to be checked against when data is being manipulated. If it is out of range or has invalid characters, an error will be thrown:

A screen shot of a computer code

Description automatically generated

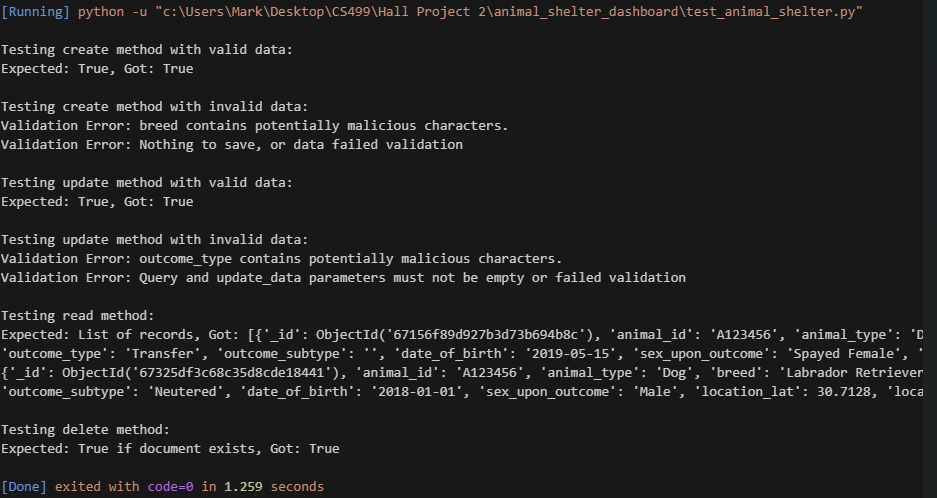
Next, I created a test script (test\_animal\_shelter.py) with functions to verify different invalid entries to ensure they are caught by the validate\_input() function.



A screen shot of a computer program

Description automatically generated

Finally, I ran the scripts to validate that it was working as expected:



The planned enhancements align with and support the following course outcomes:

* **Outcome 4**: Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for implementing solutions that deliver value and accomplish industry-specific goals. Through this enhancement, I’ve shown my ability to employ best practices in database validation to ensure data integrity and accuracy, which adds value to the application by preventing data corruption and improving reliability.
* **Outcome 5**: Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources. By adding validation and sanitization functions, I’ve addressed potential vulnerabilities that could arise from unvalidated or unsanitized data inputs, such as SQL injection attacks, which can compromise data integrity and security.

**Reflection on the Enhancement Process**

Throughout this enhancement process, I deepened my understanding of data validation and security principles. I learned how to implement data validation methods that check for correct data types, ranges, and values before any CRUD operations are performed. This validation process includes sanitizing input to prevent SQL injection and other malicious attacks, which is crucial for maintaining a secure database.

Implementing the validation functions required careful planning and testing. I started by identifying the fields in the data that needed validation, particularly those that could be vulnerable to malicious inputs. I then wrote the validation functions to check for proper data types and valid value ranges. For instance, I ensured that numerical fields were within expected ranges and that string fields did not contain any potentially harmful characters. This validation was integrated into the create and update methods so that any invalid data was rejected before it reached the database.

**Challenges Faced**

One challenge I faced was determining the specific validation rules for each data field, as different fields had different requirements. Another challenge was making sure the validation process did not impact the performance of the CRUD operations significantly, so I carefully tested and optimized the validation functions to maintain efficiency. Additionally, I added an error-handling mechanism that provided meaningful feedback in case of validation failure, which is essential to a positive user experience.

**Learning Outcomes**

This process taught me the importance of balancing security with usability and performance. While adding validation significantly improved the security and integrity of the database interactions, I also had to ensure that these additional checks did not create a bottleneck. I became more proficient in MongoDB and understood more deeply how to secure data operations in a database context. These skills will be valuable as I continue working on data-driven applications, where securing user data is essential.