James Galbreath

CS-499

8/3/25

This artifact is a Python class, AnimalShelter, that connects to a MongoDB database and performs basic CRUD operations. It was originally created in December of last year as part of coursework to demonstrate database integration and manipulation in Python using PyMongo. The program allows the user to insert (create) and retrieve (read) records from a MongoDB collection.

I selected this artifact because it demonstrates my ability to design, implement, and interact with a NoSQL database through Python, which is a critical skill in software development. This artifact showcases my ability to connect to a database securely, structure Python classes for maintainability, and handle database transactions. It also illustrates my understanding of secure coding practices by preventing NoSQL injection attacks through a custom sanitization method.

Enhancements Made  
The original version did not have adequate protection against injection attacks or strict input validation. I improved it by:

Adding a sanitize\_input() method to block dangerous MongoDB operators (e.g., $ne, $gt) and prevent dot‑notation exploits.

Implementing input validation to ensure only properly formatted dictionaries are processed.

Improving code documentation and inline comments to increase readability.

Correcting Python best practice errors (e.g., fixing the constructor from \_init\_ to \_\_init\_\_, using if \_\_name\_\_ == "\_\_main\_\_":).

This enhancement met the Databases category outcome I planned in Module One. I successfully demonstrated the ability to design and interact with databases while applying secure coding practices. No updates to my outcome‑coverage plan are required for this category.

During the enhancement process, I learned how NoSQL injection attacks differ from SQL injection and the importance of sanitizing both queries and inserted data. I also reinforced my skills in secure software development by applying strict validation and improving the maintainability of the code. One of the challenges I faced was ensuring the sanitization logic did not break legitimate MongoDB queries while still blocking potentially malicious input. This process strengthened my understanding of both database security and clean, maintainable Python code design.