4-2 Milestone 3: Enhancement Two Narrative

Richard VanSkike

CS-499 Computer Science Capstone

Southern New Hampshire University

Artifact Overview

For this milestone, the focus was on further improving the Grazioso Salvare Dog and Monkey database, which had already been transitioned from Java to Python. The original project, built during my IT-145 course, was a terminal-based application that managed a database of rescue animals with basic functions like adding, reserving, and displaying animals. The goal was to enhance its modularity, maintainability, and accessibility.

This enhancement introduced two new modules: search.py and validation.py. The search.py module enables users to search and modify records for dogs and monkeys, while validation.py handles input validation, making sure all data is correct and consistent across the application. Another key change was transitioning the data structure from simple lists to dictionaries. This upgrade improved efficiency, particularly with functions like initialize_dog_list and initialize_monkey_list, by making data retrieval more scalable and organized.

Justification for Inclusion

This artifact highlights how my programming skills have evolved, particularly in refactoring code to enhance both performance and scalability. By moving from Java to Python, I had the chance to apply object-oriented programming (OOP) concepts in a new context while also refining the structure and readability of the code.

One of the key improvements to this artifact was the inclusion of the search.py module, which introduced the ability to search for animals within the database. This feature leverages the unique name validation for animals, allowing users to efficiently locate specific entries based off the name. Once an animal is found, users can modify three attributes: name, training status, and in-service country. I intentionally left out the ability to change reservation status, as that

functionality already existed. This enhancement significantly improved control over the database, offering more robust functionality for managing the animals.

Another significant improvement to the system was refactoring the data structure from lists to dictionaries. As the company anticipates rapid growth in the number of animals in the database, continuing to use lists would lead to performance degradation over time. Lists in Python are implemented as dynamic arrays, meaning that searching for an element requires iterating through the list sequentially. This results in a time complexity of O(n) for search operations, where "n" represents the number of elements in the list (Badr 2023). As the list grows, the time it takes to find a specific element increases linearly because each element must be checked individually.

In contrast, Python dictionaries are implemented using hash tables. Instead of iterating over each element, dictionaries use a hash function to compute an index for each key, allowing for near-instant access to values. This enables search, insert, and delete operations to be performed in O(1) average time complexity (Badr 2023). In practice, this means that accessing an element in a dictionary takes constant time, regardless of how many elements are stored. This is a dramatic improvement over lists, especially as the database size increases, as performance remains relatively unaffected by the number of entries.

Though, while dictionaries offer O(1) average time complexity, there is a potential catch. In rare cases of hash collisions, where different keys produce the same hash value, the dictionary must handle these collisions through chaining or open addressing, which can degrade performance to O(n) in the worst case. Despite that, Python's built-in dictionary implementation is highly optimized to minimize collisions, making O(1) performance typical in most real-world scenarios.

By transitioning from lists to dictionaries, the system ensures that as the number of animals in the database grows, operations such as searching for a specific animal remain efficient. This makes the system scalable and better equipped to handle the expected increase in data without significant slowdowns.

The final improvement was the introduction of a dedicated validation.py module. The original artifact lacked proper input validation, leading to inconsistent and sometimes inaccurate data within the database. With this module, robust validation mechanisms were introduced without compromising functionality or usability. For instance, the module ensures that names are unique and not null. Additionally, a regular expression was implemented to enforce the correct format (MM-DD-YYYY) for the date of acquisition. These validation improvements enhance overall program reliability by optimizing input handling and maintaining data integrity.

These enhancements demonstrated how leveraging Python's strengths can lead to more efficient, scalable solutions, and they lay a solid foundation for future expansion of the application.

Achievement of Course Outcomes

For this course I am striving to reach five course outcomes. They are as follows:

- Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science.
- Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.

- Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices.
- Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
- 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.

With Enhancement One, I successfully met the first two course outcomes. Enhancement two I implemented additional enhancements to the Grazioso Salvare Animal Rescue database, I fully achieved one outcome and made significant progress toward another. Specifically, I designed and evaluated computing solutions that addressed key problems by employing algorithmic principles and adhering to computer science standards, all while carefully managing design trade-offs. The transition from lists to dictionaries was a strategic decision, balancing memory efficiency with improved search performance, especially for larger datasets. This refactor optimized the search functionality and contributed to better scalability. I have also enhanced the code with the search.py and validation.py modules. With these I introduced enhanced capabilities such as database search and edit functions, as well as input validation, resulting in more robust functionality and maintainability that align with industry best practices.

In terms of the second outcome, I developed 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. I partially

achieved this outcome by anticipating potential vulnerabilities in the software architecture. By encapsulating validation logic within the validation.py module, I established a dedicated layer for data input validation, reducing the risk of adversarial exploits and ensuring data integrity. This proactive approach not only identified and addressed vulnerabilities early on but also improved the overall security and privacy of the system. While I aim to further refine these security features, the initial enhancements have already significantly strengthened the system's defenses.

Reflection on the Enhancement Process

Enhancing this codebase provided invaluable insights into the importance of modularity and thoughtful design. Initially, I expected that transitioning from lists to dictionaries would be challenging, but after delving into Python's data structures and conducting research, such as "Python for Beginners" (2023), I was able to implement these changes in a way that streamlined the application's performance and scalability. Extensive research, including resources such as GeeksforGeeks (2024b), informed my decision to adopt dictionaries. This transition not only simplified the code but also significantly enhanced its efficiency and maintainability.

During the development of the search.py and validation.py modules, I explored the concept of Separation of Concerns (SOC), guided by resources like GeeksforGeeks (2024a). I now fully grasp the significance of SOC in maintaining and expanding large-scale projects. To that point, I was performing input validation individually within each method, which led to bloated code and increased errors. After reviewing my implementation, I realized I could abstract the validation logic into a dedicated module and invoke its methods as needed. This approach dramatically reduced the code complexity across other modules and eliminated many of the issues I had encountered.

In developing the search.py functionality, I recognized that I had already incorporated elements of this process in other functions, such as in reserve_animal.py. The original functionality involved a pre-determined search and subsequent updates to the database. By expanding user control and allowing broader functionality for database management, I significantly enhanced the user experience and overall flexibility of the application. This process deepened my understanding of software development best practices and underscored the importance of continuous learning, particularly when integrating new functionality into an existing system.

REFERENCES

Badr, A. (Ed.). (2023, January 19). TimeComplexity. Python Wiki. https://wiki.python.org/moin/TimeComplexity

GeeksforGeeks. (2024a, February 13). Separation of concerns (SOC). https://www.geeksforgeeks.org/separation-of-concerns-soc/

GeeksforGeeks. (2024b, September 6). Difference between list and dictionary in Python. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-list-and-dictionary-in-python/

Raj, A. (2023, May 12). List vs dictionary in python. PythonForBeginners.com. https://www.pythonforbeginners.com/basics/list-vs-dictionary-in-python