Kendal Guizado

Milestone Three

For Milestone Three, I focused on enhancing the tenant screening application, originally created as a simple Python program during my coursework. This artifact evaluates rental applicants based on a series of financial factors, including credit score, income-rent ratio, and employment length. The primary enhancement was transitioning the core algorithm to the backend while optimizing it to handle multiple applicants efficiently. The goal was to improve both the performance and scalability of the application, ensuring that it aligns with industry standards for robust computing solutions.

The tenant screening application was initially built as a standalone Python script that computed a score for individual applicants based on their financial history. In this enhancement, the Python logic was refactored and moved to a Node.js backend, providing a more secure, scalable, and efficient solution. The algorithm evaluates applicants across multiple criteria, such as credit score, income-to-rent ratio, employment length, and landlord references, among other factors. These scores are then combined to generate a total score, which informs the decision to approve, conditionally approve, or reject an applicant.

Skills Demonstrated and Course Outcomes

Course Outcome 3: Algorithms and Data Structures

One of the significant improvements in this enhancement was optimizing the algorithm for better performance. The original Python code handled a single applicant’s data, but the new backend logic allows for multiple applicants to be evaluated simultaneously. The algorithm was refactored to work efficiently with multiple data points using array-based operations, reducing the overhead of processing each applicant individually. The time complexity of the enhanced algorithm is O(n), where n is the number of applicants. This is achieved by processing each applicant's data through various scoring functions, all of which operate in constant time (O(1)). As a result, the overall time complexity remains linear, which ensures that the system performs consistently even as the number of applicants scales.

This enhancement demonstrates my ability to design efficient algorithms that can handle larger datasets without performance degradation, directly aligning with Course Outcome 3 (Designing and evaluating computing solutions using algorithmic principles).

Course Outcome 4: Software Engineering and Design

In addition to performance improvements, I implemented modular functions for each scoring criterion. For example, I created separate functions to calculate the score for credit score, income-rent ratio, and employment length. Each function assigns scores based on real-world financial assessment thresholds. This modular approach made the code more maintainable, scalable, and easier to update if new scoring criteria are introduced.

Transitioning the algorithm to a backend-based architecture also ensured that the system is more secure and scalable, which aligns with Course Outcome 4 (Implementing well-founded software engineering techniques that deliver value and meet industry-specific goals).

Course Outcome 5: Security

Another key aspect of this enhancement was the introduction of error handling and input validation. In the original implementation, fields like credit score and monthly income could default to zero, potentially skewing the applicant's evaluation. Now, the backend performs checks to ensure that each input is valid and meaningful before proceeding with the calculation. This improvement ensures that the accuracy and reliability of the scoring system are maintained, and it prevents invalid data from corrupting the evaluation process.

By incorporating secure data handling practices, I ensured that all data is processed securely and correctly, reducing the risk of inaccurate evaluations or security vulnerabilities, which directly supports Course Outcome 5 (Developing a security mindset that mitigates design flaws and ensures data privacy).

Reflection on the Enhancement Process

Throughout this enhancement process, I gained a deeper understanding of algorithm optimization and backend development. One challenge I encountered was ensuring that the algorithm handled edge cases efficiently, such as applicants with missing or incorrect data. Integrating validation checks and improving the scoring granularity were critical to overcoming these challenges. By modularizing the algorithm and moving it to the backend, I not only improved the performance of the application but also made it more scalable and maintainable.