**Milestone Two Narrative: Software Design and Engineering Enhancement**

**Artifact Overview**

The artifact I selected for the software design and engineering category is a Contact Management System originally developed for CS-320: Software Testing, Automation, and Quality Assurance. The application was initially designed to perform basic CRUD (Create, Read, Update, Delete) operations for managing contacts. Each contact includes a unique contact ID, first name, last name, phone number, and address. The original implementation used an ArrayList for storage and had basic input validation and JUnit tests.

**Justification for Inclusion**

I selected this artifact because it reflects my foundational knowledge in object-oriented programming, modular software design, and unit testing. It was an ideal candidate for enhancement since the original version had limited performance optimization and structure. This project demonstrates my ability to apply real-world software engineering techniques and transition from a classroom-style program to a more robust, maintainable, and efficient application.

Enhancements Completed:

* Refactored Code to MVC-Ready Design: While not implementing a full MVC UI, the structure now supports future View and Controller layers with clean separation of logic in ContactService and data in Contact.
* Improved Performance with HashMap: Replaced the ArrayList with a HashMap<String, Contact> for O(1) time complexity in lookups, updates, and deletions.
* Robust Error Handling: Added meaningful exception handling using custom Exception messages for invalid operations such as looking up non-existent contacts.
* Input Validation: Enforced stricter validation for phone numbers, field lengths, and null values.
* Expanded Testing: Created a new, enhanced ContactServiceTest.java file using JUnit 5, which includes positive and negative test cases for CRUD operations and error handling.
* Updated Documentation: Added a professional README.md file with usage instructions and a compilation guide.

These improvements demonstrate my skills in software architecture, performance optimization, exception handling, and unit testing, and make the artifact portfolio-ready for technical recruiters and interview discussions.

**Alignment With Course Outcomes**

This enhancement helped me make measurable progress in the following CS 499 course outcomes:

* Outcome 3: Designed and evaluated a computing solution by selecting appropriate data structures and applying validation logic.
* Outcome 4: Demonstrated well-founded techniques in implementing clean and efficient object-oriented code that solves a real-world use case.
* Outcome 5: Applied a security mindset by implementing strict input validation and exception handling to prevent logic errors and application crashes.

I also improved my communication and documentation skills (Outcome 2) through the updated code structure and README file. The artifact is now clearer for future users or developers to understand and extend.

**Reflection on the Enhancement Process**

Refactoring this artifact taught me the importance of maintainable and modular code. Initially, converting the ArrayList to a HashMap caused cascading issues in the update and delete logic, so I had to rethink how IDs were assigned and managed. Writing test cases also revealed logic gaps in my original implementation, especially in how edge cases were handled, such as empty input or invalid IDs.

The most valuable part of this process was seeing my code evolve from a functional project to a professional-quality product. I now understand the importance of writing code not just for a program to "work," but for it to be secure, efficient, testable, and well-documented, qualities that are critical in a professional setting.