# Artifact Narrative – TaskService (Software Design & Engineering)

## Artifact Description

The artifact I selected for the Software Design and Engineering category is my TaskService project from CS 320: Software Test Automation. This project originally provided a simple service for managing tasks, including creating, retrieving, updating, and deleting task objects. The original implementation met the minimum requirements but lacked robust input validation, meaningful exception handling, and consistent documentation.

## Justification for Inclusion

I chose this artifact because it clearly demonstrates my growth in applying software engineering best practices. The enhancements I implemented showcase improvements in input validation, exception handling, refactoring, and documentation. These enhancements directly align with professional expectations for clean, maintainable, and scalable software, and they illustrate my ability to deliver software that meets both functional and non-functional requirements.  
  
Specifically, this artifact showcases:  
- Custom exception handling through the creation of a ValidationException class.  
- Refactored helper methods (validateTask, requireNonBlank, validateId) that reduce code duplication.  
- Comprehensive input validation, preventing invalid IDs, null tasks, and empty fields.  
- Improved documentation with JavaDoc comments that explain the purpose and usage of each method.  
- Expanded unit test coverage to validate both positive and negative cases.  
  
These enhancements make the artifact significantly more professional and reflective of industry standards.

## Reflection on the Enhancement Process

Enhancing this artifact taught me the the importance of balancing readability, functionality, and maintainability. One challenge I faced was deciding how much validation logic to include in the service layer without overcomplicating the design. By extracting reusable validation helpers, I was able to simplify the main business methods while still ensuring robust error handling.  
  
Another challenge was replacing generic exceptions with a custom ValidationException. This change required updating both the service code and the associated test cases. However, the result is more meaningful error reporting that would help a development team diagnose issues quickly.  
  
The enhancement process also reinforced my understanding of the software development life cycle and iterative testing. Each change I made was immediately tested to ensure no regressions were introduced. This iterative approach mirrors real-world agile practices, where small, incremental improvements lead to a more reliable and maintainable codebase.

## Course Outcomes Demonstrated

Through this enhancement, I demonstrated the following capstone outcomes:  
- Software Design and Engineering: I applied principles of clean code, modularity, and maintainability by refactoring and documenting the service layer.  
- Communication: By adding JavaDoc and in-code comments, I improved the readability of my code, which is critical for collaboration in professional environments.  
- Security Mindset: Strong input validation and custom exceptions reduce the likelihood of invalid or malicious data being processed.  
- Tools and Techniques: Using JUnit tests and refactoring strategies shows my ability to apply professional tools and processes to deliver robust software.

## Conclusion

This enhanced TaskService artifact represents my growth in applying professional software engineering practices. By implementing validation, custom exceptions, refactoring, and documentation, I transformed a basic project into one that reflects the quality, maintainability, and professionalism expected in the software industry.