****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## 

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **September 20, 2025** | **Medhawi Bista** | **Final Draft** |
| **1.1** | **September 22, 2025** | **Medhawi Bista** | **Edits** |
| **1.2** | **September 23, 2025** | **Medhawi Bista** | **Final Edits** |

## Client



## Developer

Medhawi Bista

## Algorithm Cipher

To ensure that the integrity and authenticity of Artemis Financials’ sensitive client data, it is recommended to use a cryptographic hash algorithm from Java’s MessageDigest class. SHA-256 is the primary recommendation with SHA3-256 as an optional choice for additional security in the future. Both algorithms generate fixed-length digital fingerprints called checksums. Checksums verify that files have not been tampered with or been accessed by unauthorized individuals while in transmission or during storage (Oracle, 2017; NIST, 2015a).

SHA-256 and SHA3-256 both produce 256-bit hash values, which means there are 2^256 possible outputs. This makes accidental or malicious collisions extremely unlikely. Collisions are a result of different inputs hashing out the same outputs. While these algorithms do not directly encrypt data, they put a reliable mechanism in place that makes it easier to detect unauthorized changes, as well as support data, auditability, and non-repudiation (NIST, 2015a; Wikipedia Contributors, 2025a).

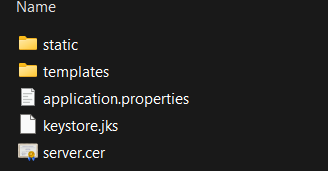
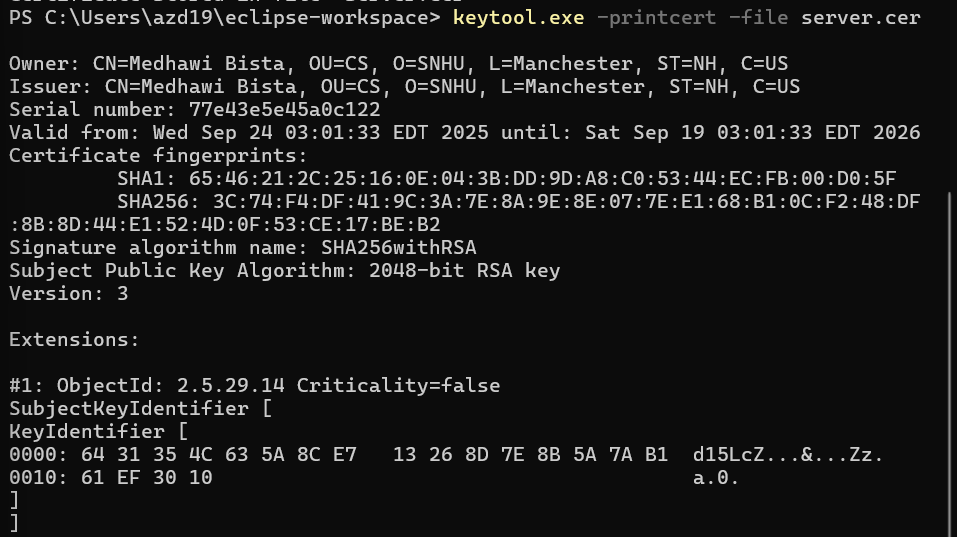
As stated before, SHA-256 and SHA3-256 are one-way functions that generate a 256-bit digest. The bit length ensures that the algorithms have a high resistance to common attacks like preimage, collision, and brute-force. Just changing 1-bit in the input yields an entirely different output, this ensures that any modifications made to a file can be easily and reliably detected (NIST, 2015a; Oracle, n.d.).

With that said, hashing differs from encryption because it’s keyless and deterministic. To up the security level hashing can easily be combined with a secret key in HMACs or salted with a random string to prevent things like replay or rainbow table attacks. Symmetric and asymmetric are used when confidentiality and decryption are required, this is because with symmetric keys a file is encrypted and decrypted using the same key, while with asymmetric keys are encrypted and decrypted with different keys. For something like verifying the integrity of a file’s integrity, hashing will be sufficient (NIST, 2017; OWASP, 2023b).

Older hash functions like MD5 and SHA-1 are now considered insecure due to their higher risks for collision vulnerabilities. The SHA-2 and SHA-3 families are more commonly used. SHA-256 is a part of the SHA-2 family and remains widely adopted in financial, government, and industry applications. The SHA-3 family, of which SHA3-256 is a member, is based on the Keccak algorithm and provides an alternative cryptographic model and offers additional protection and resilience against certain attack vectors. These are also good algorithm choices as Java’s native support for both recommended algorithms makes integration into existing systems fairly smooth and easy (NIST, 2015a; Oracle, 2017; NIST, 2015b).

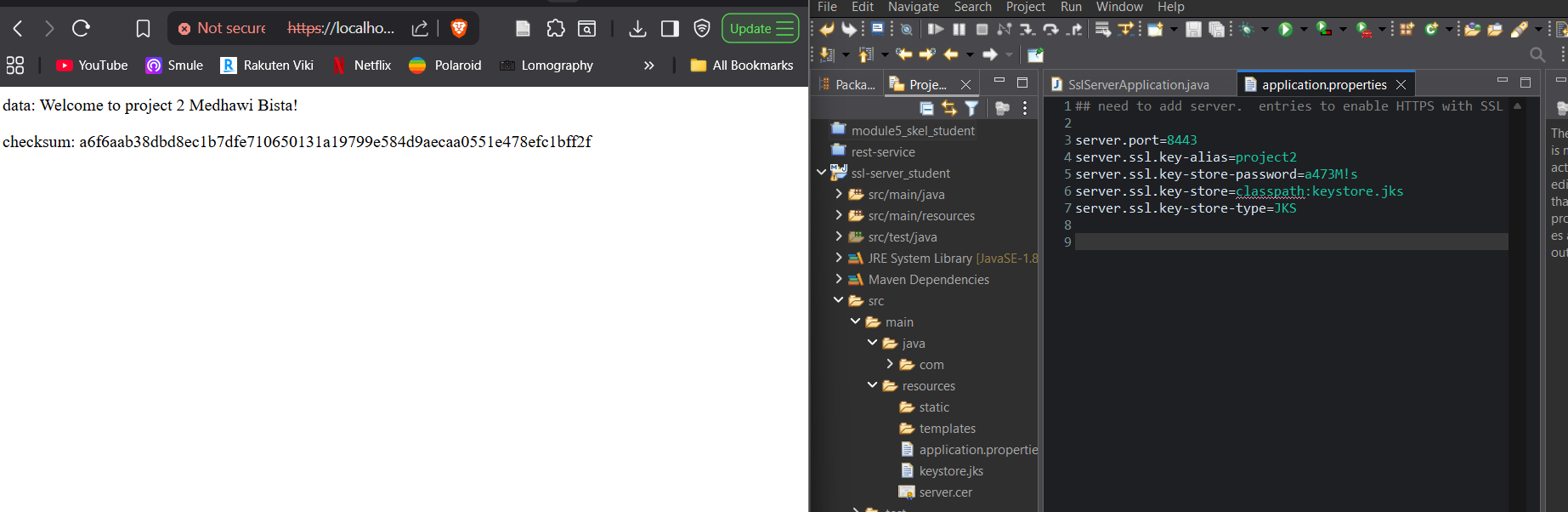
SHA-256 is recommended for Artemis Financial because of how it balances security, performance and compatibility. It provides a strong resistance against collisions and preimage attacks and is widely implemented and supported in enterprise and financial systems. SHA3-256 is recommended as a future-proof option due to the fact that it may have slower performance and limited legacy support. Combining these algorithms with additional security measures like salting or HMACS aids in mitigating many common risks and concerns, like collisions, malicious file tampering, or replay attacks. These hashing algorithms also aid in ensuring that regulatory requirements like PCI DSS, GDPR, SOX, and GLBA, are met and upheld (European Union, 2016; Federal Trade Commission, 2023; PCI Security Standards Council, 2022; U.S. Congress, 2022)

## Certificate Generation

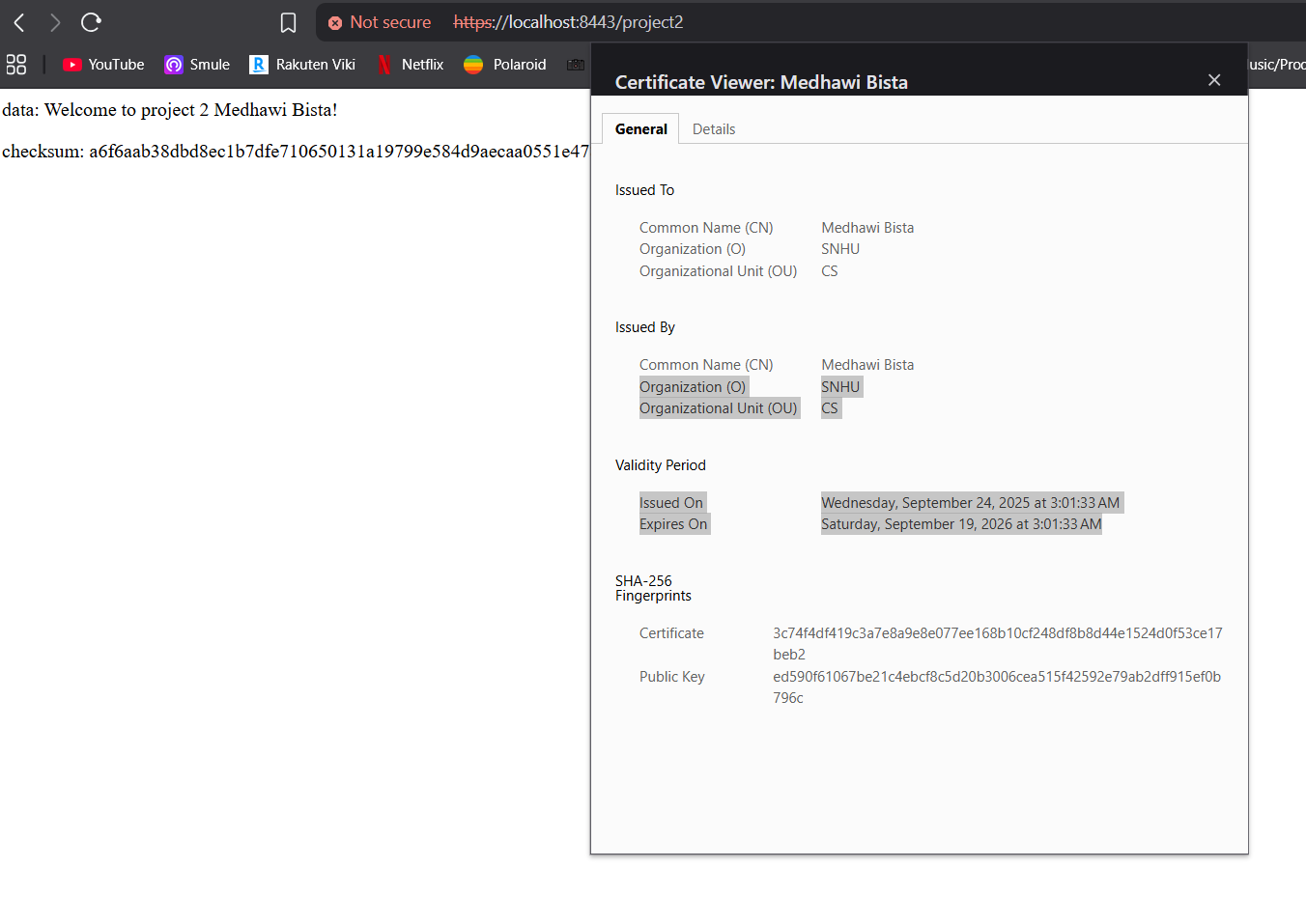


## Deploy Cipher

A screenshot of a computer

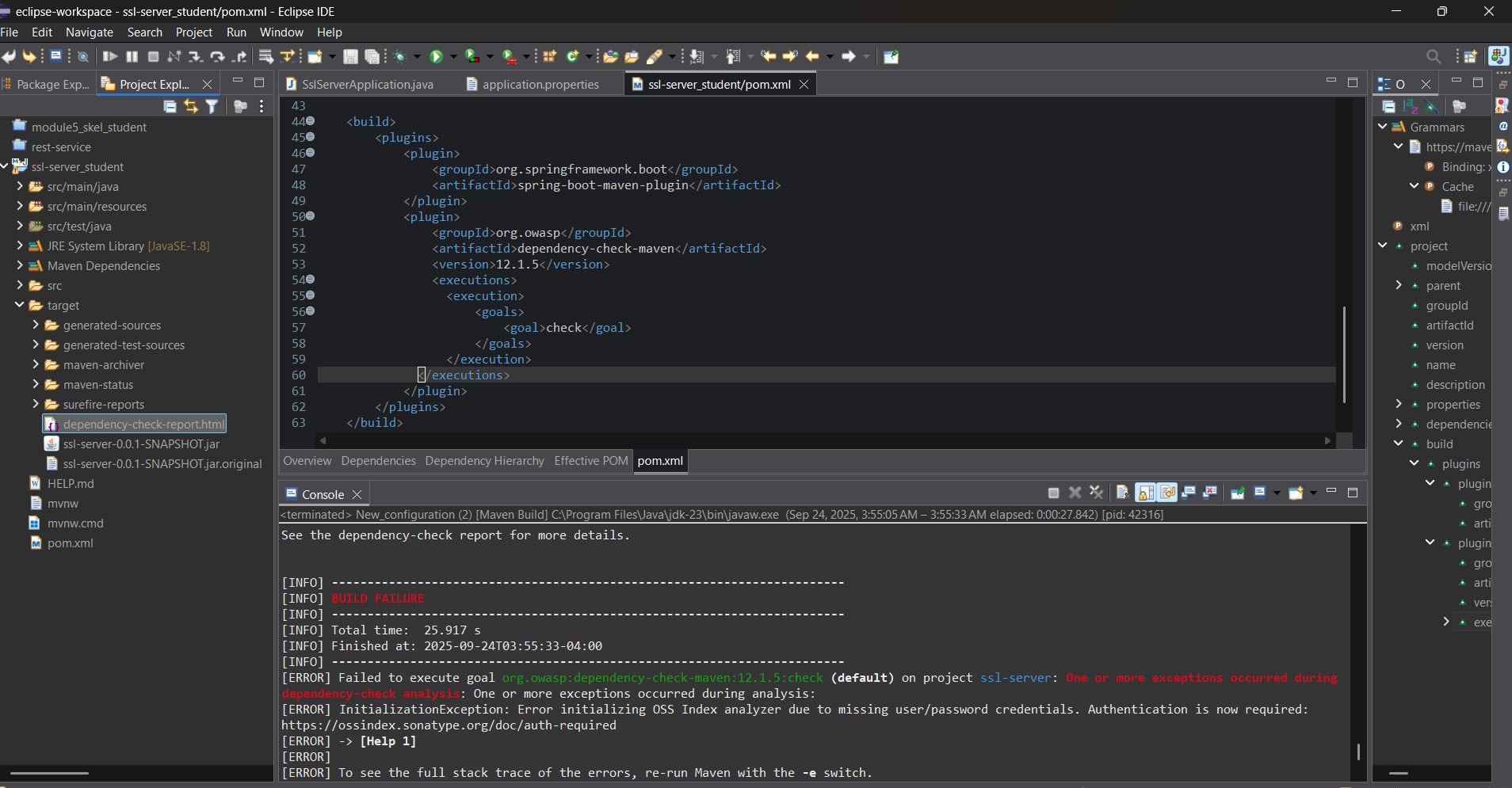
AI-generated content may be incorrect.

## Secure Communications



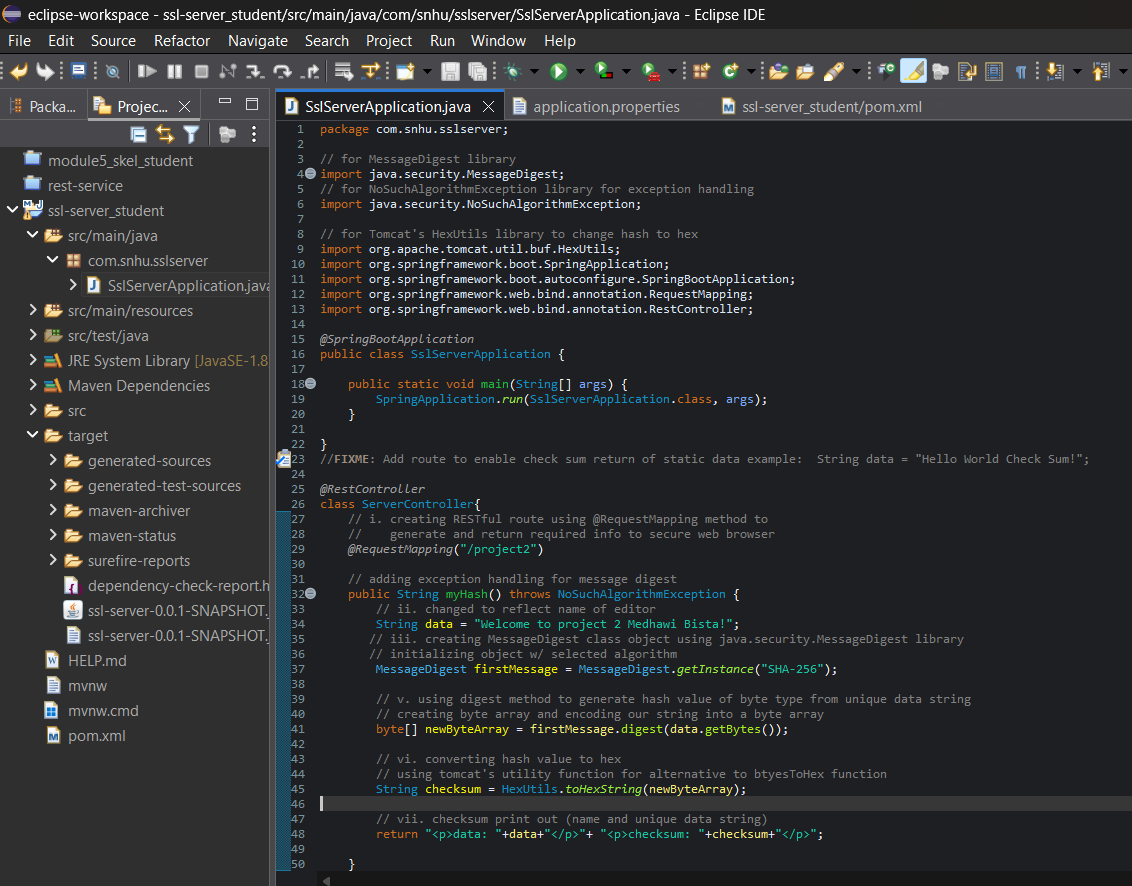
## Secondary Testing

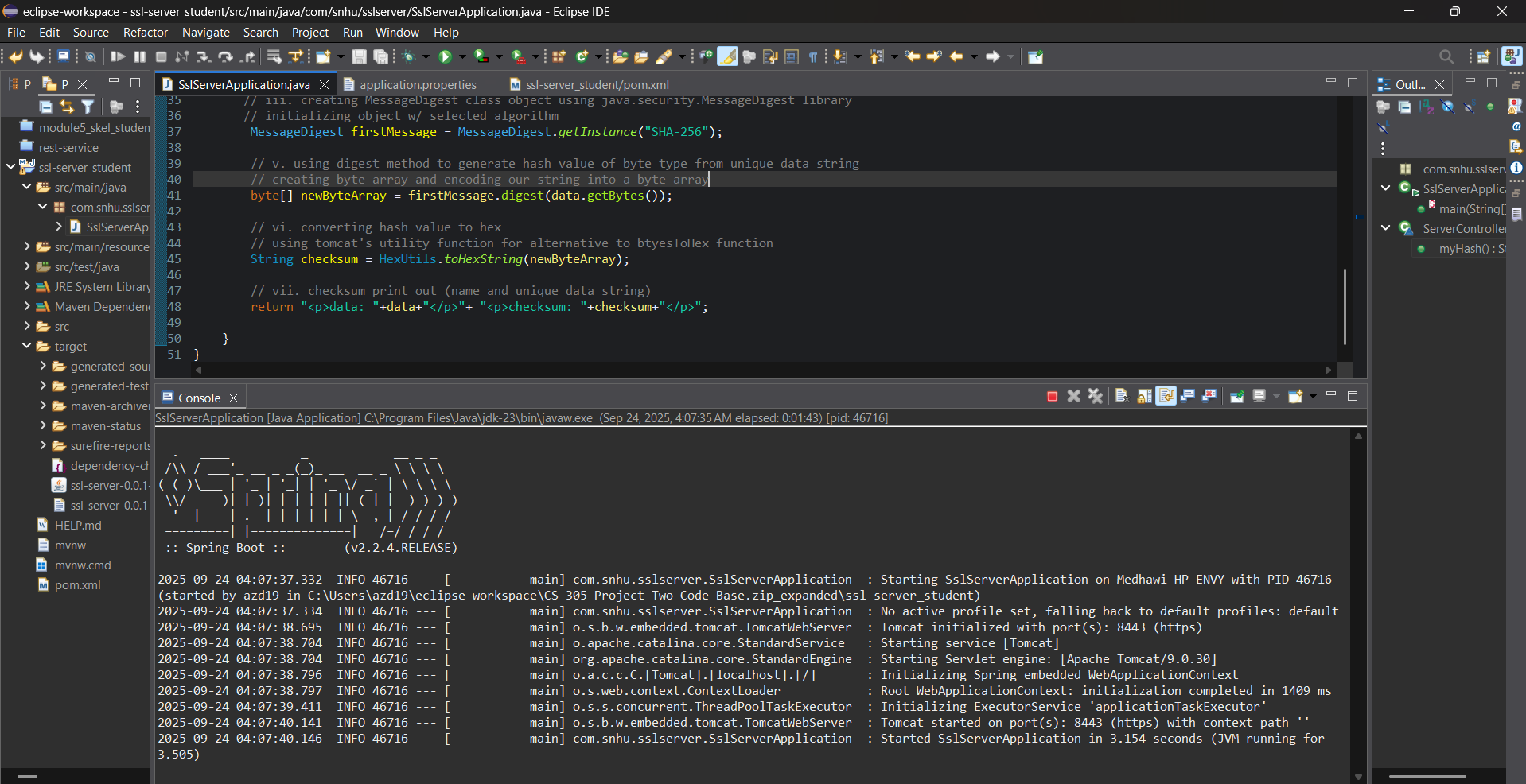
## Due *to OSS index analyzer requiring credentials, no matter what I tried it kept saying build failure.*



## 

## Functional Testing





## Summary

The code was refactored to address several vulnerabilities that were identified during the security testing process in accordance with the vulnerability assessment diagram. The main focus was in ensuring that input validation and error handling was taken care of so that malicious inputs cannot bypass the applications logic. Focus was also put on improving code quality, implementing cryptography, and ensuring client/server security.

The application has been strengthened against threats, whether the attacks come externally or happen due to insider misuse. This has been done by layering several authentication safeguards and proper exception handling. By taking a layered approach, it is possible to ensure that each stage of the vulnerability assessment from identification to remediation is accounted for and helps ensure that the final product algins with secure coding protocols and meets industry and government regulations and requirements.

**Input validation** was done by adding checksum verification. This lets the application ensure that files are not altered while in transit, which in turn mitigates injection and manipulation risks (OWASP, 2021). By implementing SSL/HTTPS certificates, client-server communication, along with sensitive financial data during transmission, is more secure and protected, thus ensuring tackling **Cryptography**(OWASP, 2023a; IBM, 2023). The implementation of HTTPS enforces the communication between users and application servers to be authenticated, reducing the risks that are often associated with plain-text HTTP. It also defends against man-in-the-middle attacks. This tackles **Client/Server Security.**

The checksum function that was implemented throws a *NoSuchAlgorithmException*. This helps prevent system crashes and avoids exposing sensitive stack trances (which could be followed to end users) and aligns with secure **Code Error Handling** practices (IBM, n.d.). Lastly, secure code practices were applied, along with automated vulnerability scanning. This makes it possible to identify weaknesses that could have been introduced by third-party libraries and was done by using the OWASP Dependency-Check tool. The check was done alongside a manual code review. This double check ensures that the code is up to industry standards for security, readability, and maintainability (Devcom, 2024; Vaultinum, 2024). These implementations fall underneath **Code Quality**.

By addressing multiple areas where vulnerabilities were identified, or areas that were observed to be weak, means that the application now incorporates multiple different layers of defense. By taking a layered approach, it helps to ensure that if one safeguard were ever to fall other safeguards would still remain in place to protect the user data alongside the integrity of the system

## Industry Standard Best Practices

During the refactoring, applying and following industry standard best practices were an important part of ensuring that the application was secure. Standard best practices like **encryption for data in transit**, **proper coding methods and styles**,and **secure coding to mitigate known vulnerabilities**, are some examples of the practices that were employed. Due to the nature of the data that the company handles, it is imperative that the encryption protocols meet the minimum strength requirements, so things like validation of inputs and implementation of error handling were done.

To further strengthen the protection and security of the application and its data, a strong encryption, like HTTPS, was used alongside a SHA-256 hash algorithm. By doing this, the application is aligned with the regulatory requirements for financial institutions and helps mitigate interceptions risks (The SSL Store, 2019). Checksums allow for **robust input validation**. Validation of all user inputs, including any uploaded files is a stand best practice because it aids in preventing injections attacks (OWASP Cheat Sheet Series, n.d.). Tools like the OWASP Dependency-Check help to make sure that vulnerabilities that exist in third-party libraries are found, identified, and quickly remediated. Following this practice helps protect against risks that are introduced by external code dependencies (OWASP, 2021).

Doing things like **secure code and error handling** follows secure coding guidelines and greatly reduces the risk of attacks from things like arbitrary code execution (Wikipedia, 2025a; Secure Code Warrior, n.d.). The base part of standard best practice is maintaining high code quality. Something that is done through readability, testability, and the maintainability to ensure that the application stays secure, efficient, and adaptable over time. **Code quality assurance** greatly lowers the risk of vulnerabilities that are caused simply by poor coding patterns (Devcom, 2024); Vaultinum, 2024).

The final part of industry standard best practices is to **implement regular testing and review**. By continuing to regularly run static and dynamic testing, do manual code reviews, and encourage continuous integration of pipelines helps ensure and maintain security and compliance with regulatory requirements. Rather than tackling the aftermath, following these practices helps support resilience against new threats as they emerge

Following these practices not only fulfils compliance obligations but will also help foster trust in the company. Since the company handles highly sensitive personal data, following industry-standard safeguards helps ensure the integrity, security, confidentiality, and availability of that data and information.

## References

A02 Cryptographic Failures—OWASP Top 10:2021. (n.d.).

<https://owasp.org/Top10/A02_2021-Cryptographic_Failures/>

Arbitrary code execution. (2025). In *Wikipedia*.

<https://en.wikipedia.org/w/index.php?title=Arbitrary_code_execution&oldid=1304101188>

Code Quality: What It Is And How To Improve It | Devcom. (2024, November 13).

<https://devcom.com/tech-blog/code-quality-definition-how-to-improve-code-quality/>

CWE - CWE-917: Improper Neutralization of Special Elements used in an Expression Language

Statement ('Expression Language Injection’) (4.18). (n.d.).

<https://cwe.mitre.org/data/definitions/917.html>

European Union. (2016). General Data Protection Regulation (GDPR).

<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679>

Expression Language Injection | OWASP Foundation. (n.d.).

<https://owasp.org/www-community/vulnerabilities/Expression_Language_Injection>

Federal Trade Commission. (2023). Gramm-Leach-Bliley Act.

<https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act>

Getting Started | Building a RESTful Web Service. (n.d.). *Spring*.

<https://spring.io/guides/gs/rest-service>

IBM. (2023, November 14). *What Is Cryptography?*

<https://www.ibm.com/think/topics/cryptography>

IBM Developer. (n.d.). *Understanding software security with simple code examples.*

<https://developer.ibm.com/articles/coder-challenge-find-the-security-bug/>

Input Validation—OWASP Cheat Sheet Series. (n.d.).

<https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html>

Integrity, S. S. (2019, June 12). *How are code quality & code security related?* Medium.

<https://medium.com/@SW_Integrity/how-are-code-quality-code-security-related-6b740ebda465>

Kristin. (2024, February 12). *What is code quality and why is it important?* Vaultinum.

<https://vaultinum.com/blog/what-is-code-quality-and-why-is-it-important>

Manico, J., & Detlefsen, A. (2014). *Iron-Clad Java: Building Secure Web Applications*. McGraw Hill

Computing. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/>  
NIST. (2012). *Guide to Computer Security Log Management (SP 800-92)*.

<https://csrc.nist.gov/publications/detail/sp/800-92/final>  
NIST. (2015a). *Secure Hash Standard (SHS) (FIPS PUB 180-4)*. National Institute of Standards and

Technology. <https://csrc.nist.gov/publications/detail/fips/180/4/final>  
NIST. (2015b). *SHA3-256 Standard: Permutation-Based Hash and Extendable-Output Functions (FIPS PUB*

*202)*. National Institute of Standards and Technology.

<https://csrc.nist.gov/publications/detail/fips/202/final>  
NIST. (2017). *Digital Identity Guidelines (SP 800-63B)*.

<https://pages.nist.gov/800-63-3/sp800-63b.html>  
NIST. (2023). *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1*.

<https://www.nist.gov/cyberframework>  
Oracle. (2017). *Java Security Standard Algorithm Names*.

<https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#messagedigest-algorithms>

Oracle. (n.d.). *MessageDigest (Java Platform SE 8)*.

<https://docs.oracle.com/javase/8/docs/api/java/security/MessageDigest.html>  
OWASP. (2021). *Cryptographic Storage Cheat Sheet*.

<https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html>  
OWASP. (2023a). *Authentication Cheat Sheet*.

<https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html>  
OWASP. (2023b). *Password Storage Cheat Sheet*.

<https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>

OWASP. (2021). *OWASP Dependency-Check.*

<https://owasp.org/www-project-dependency-check/>

PCI Security Standards Council. (2022). *Payment Card Industry Data Security Standard (PCI DSS)*

*v4.0*. <https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss>

Poor coding patterns can lead to big security problems… so why do we encourage them? - Blog.

(n.d.). *Secure Code Warrior*. <https://www.securecodewarrior.com/article/poor-coding-patterns-can-lead-tobig-security-problems-so-why-do-we-encourage-them>

The Crucial Role of Cryptography in Cybersecurity—eInfochips. (2024, April 15).

<https://www.einfochips.com/blog/the-crucial-role-of-cryptography/>

The SSL Store. (2019). *FTC Safeguards Rule Compliance: Encryption Requirements.*

<https://www.thesslstore.com/blog/ftc-safeguards-rule-encryption-requirements/>

Understanding Cryptography: What It Is and How It’s Used. (2024, January 4). University of Tulsa Online.

<https://online.utulsa.edu/blog/understanding-cryptography/>

U.S. Congress. (2002). *Sarbanes-Oxley Act of 2002*.

<https://www.govinfo.gov/content/pkg/PLAW-107publ204/pdf/PLAW-107publ204.pdf>  
Wikipedia Contributors. (2025a). *Cryptographic hash function*. In *Wikipedia*.

<https://en.wikipedia.org/w/index.php?title=Cryptographic_hash_function&oldid=1307048714>  
Wikipedia Contributors. (2025b). *Rainbow table*. In *Wikipedia*.

<https://en.wikipedia.org/w/index.php?title=Rainbow_table&oldid=1309340383>