****

# 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 | 02/21/25 | Nawras Janoudi | Documented Algorithm Cipher and Secure Communications sections |
| 2.0 | 02/22/25 | Nawras Janoudi | Documented Deploy Cipher, Certificate Generation, Secondary Testing, Functional Testing, Summary, and Industry Standard BPs |
| 2.1 | 02/22/25 | Nawras Janoudi | Document review and corrections |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Nawras Janoudi

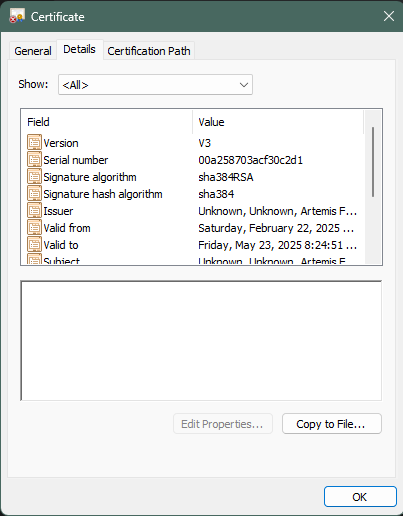
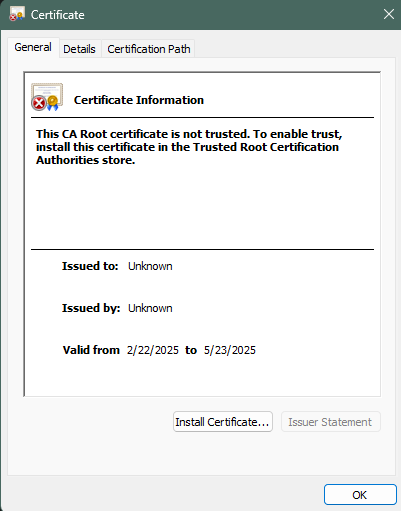
## Algorithm Cipher

For Artemis Financial's needs of secure file verification and web communication, SHA-256 is recommended for generating checksums due to its robust 256-bit hash output and widespread adoption. For secure web communication, TLS 1.3 with AES-256 encryption is advised, providing a strong combination of asymmetric and symmetric encryption for data protection. SHA-256 is a one-way cryptographic hash function producing a fixed-size 256-bit output, ensuring data integrity. AES-256 is a symmetric encryption algorithm utilizing a 256-bit key, offering high-level data confidentiality. TLS 1.3, the latest secure transport protocol, employs both asymmetric encryption for key exchange and symmetric encryption for data transfer, leveraging random numbers for session key generation and attack prevention. Cryptographic algorithms have evolved significantly, from historical ciphers to modern standards like SHA-256 and AES, driven by the need to counter evolving cyber threats. The current state emphasizes strong, widely adopted algorithms and continuous research into post-quantum cryptography to ensure future security.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.

Even though I successfully installed the certificate, I am still getting a ‘Not secure’ connection which may be due to the certificate not being officially issued by the CA.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

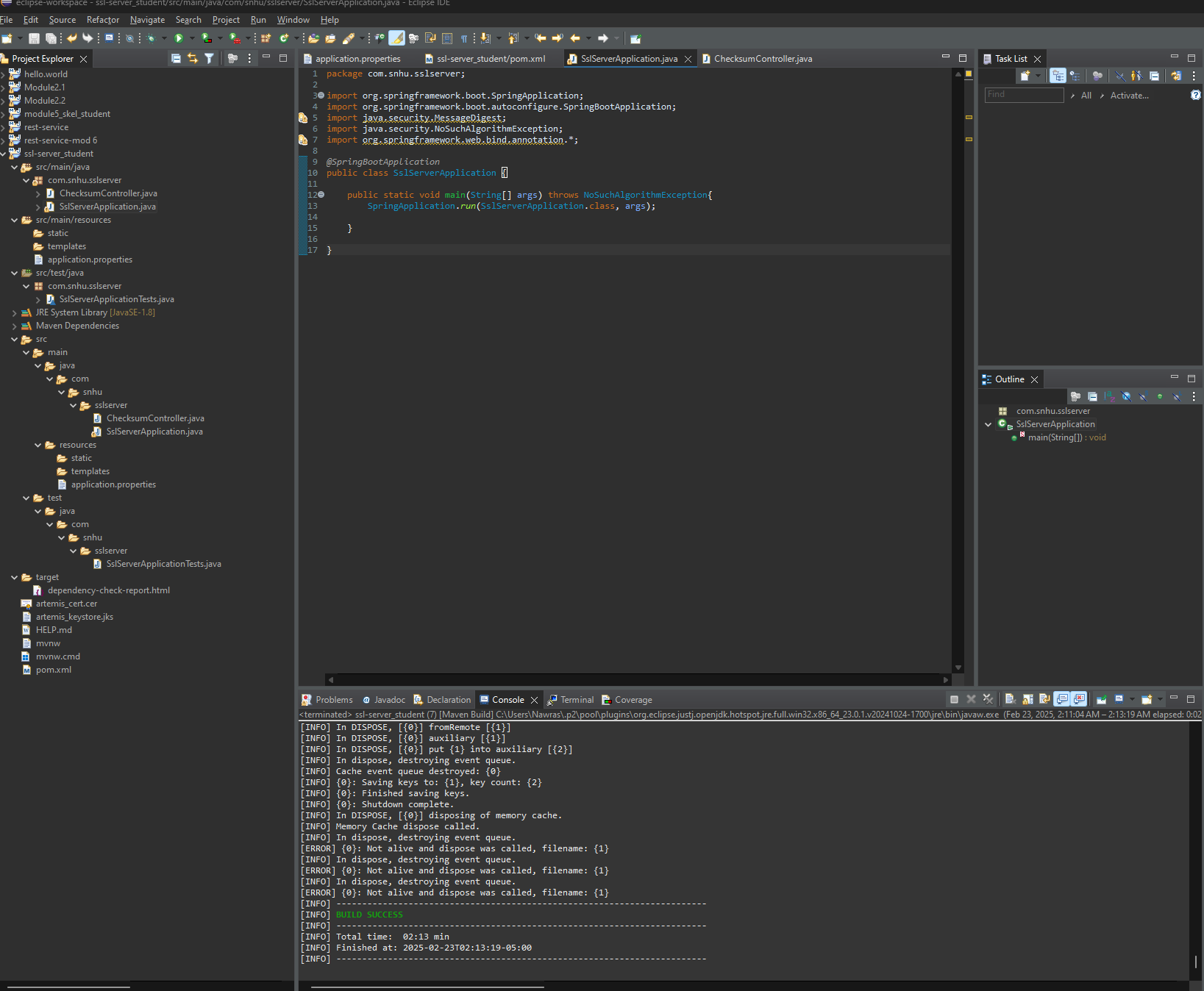
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

To enhance the security of the Artemis Financial application, several key refactoring were implemented. First, SHA-256 checksum functionality was integrated to ensure data integrity during file transfers, verifying that files remain unchanged. Second, HTTPS was configured by adding SSL properties to the *application.properties* file, enabling secure communication and protecting sensitive data during transmission. Second, a thorough manual code review was conducted to identify and address potential vulnerabilities, including input validation checks to prevent injection attacks and secure storage practices for sensitive data. These changes focused on data integrity, confidentiality, and protection against common web vulnerabilities. By following secure coding practices and conducting rigorous testing, the refactored application aligns with security testing protocols and mitigates potential risks, safeguarding user data and maintaining the company's reputation

## Industry Standard Best Practices

This project adhered to industry standard best practices for secure coding to mitigate potential vulnerabilities and maintain the software application's existing security measures. Input validation techniques were meticulously applied to sanitize user inputs and prevent injection attacks, such as cross-site scripting (XSS) and SQL injection. Additionally, the application was fortified against common web vulnerabilities by implementing appropriate security controls and following guidelines from organizations like OWASP. By adhering to these best practices, Artemis Financial can protect customer data, prevent financial losses due to security breaches, maintain a good reputation, and comply with relevant regulations and standards. Secure coding practices are essential for the company's overall well-being, fostering trust and confidence among its customers and stakeholders.