

# 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** | **October 15th** | **Joshua Williamson** |  |

## 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

Joshua Williamson

## Algorithm Cipher

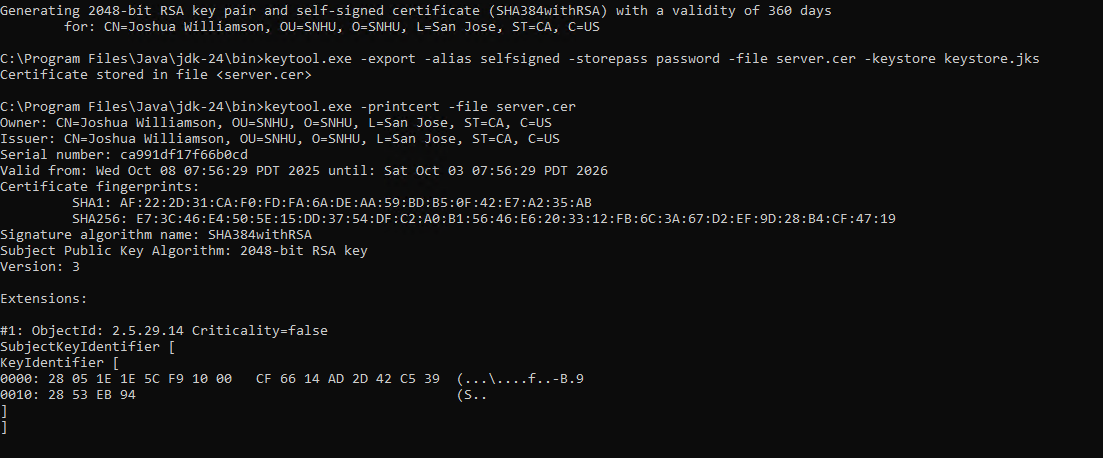
Artemis Financial is seeking to enhance the security of its web application to ensure safe and confidential communication. Given that financial institutions are prime targets for malicious actors seeking unauthorized access to sensitive data, encryption is the most appropriate safeguard. Proper encryption renders any intercepted data useless without the correct decryption key.

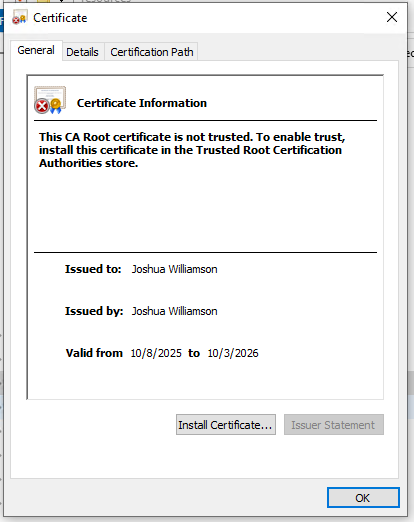
To secure communications effectively, I recommend implementing asymmetric encryption. This method uses a public key to encrypt data and a private key to decrypt it, ensuring that only the intended recipient can access the original information.

For the highest level of security when data may be transmitted externally. I suggest using the SHA-256 algorithm along with 256-bit keys. SHA-256 offers strong cryptographic security with a vast number of possible key combinations, making brute-force attacks extremely difficult. It leverages Java’s random number generator to produce secure, non-reversible checksums that validate data integrity. The SHA-256 hash function will generate a checksum of the transmitted message, helping confirm that it hasn’t been altered in transit.

## Certificate Generation

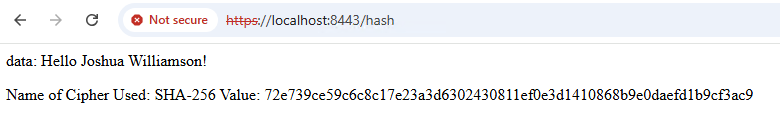
Insert a screenshot below of the CER file.





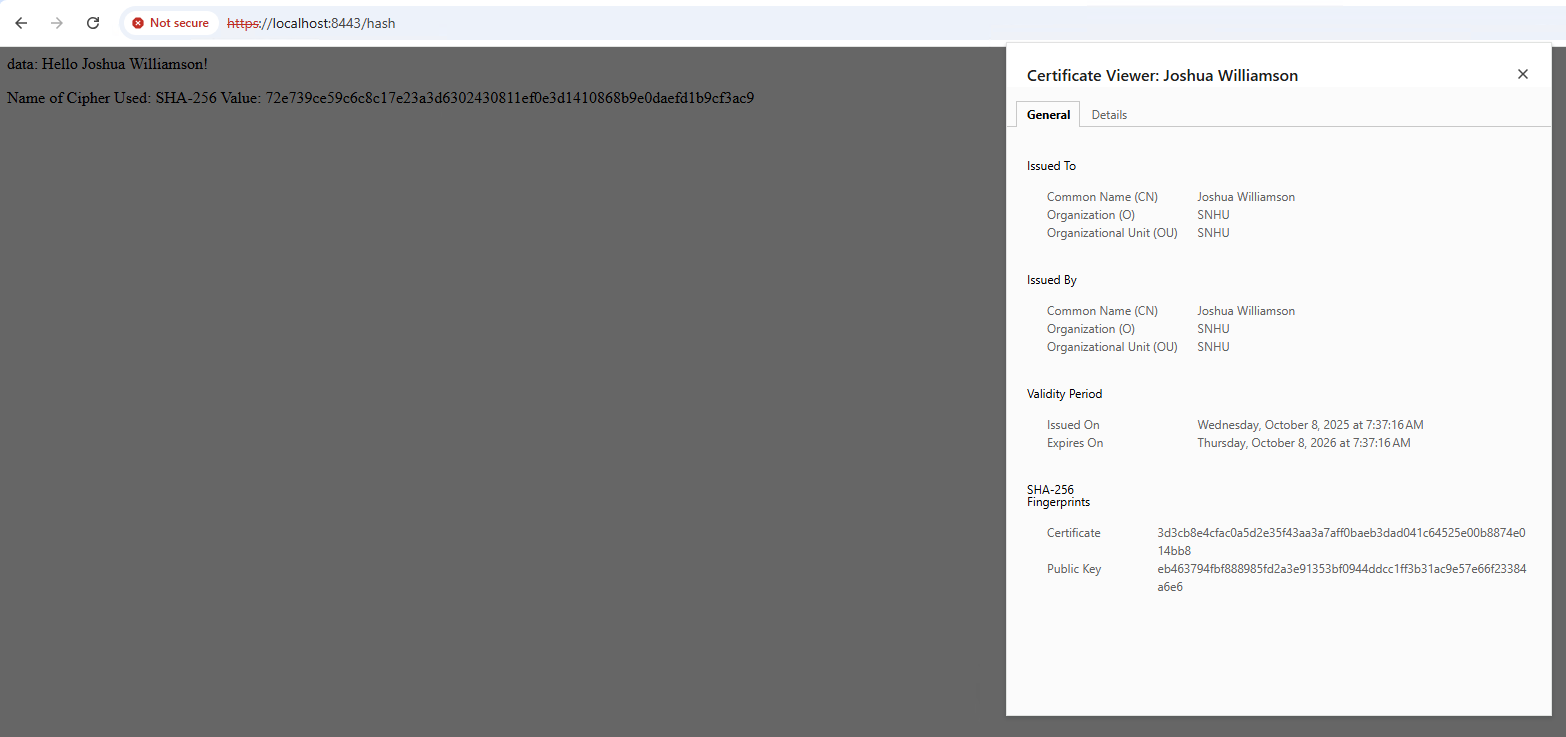
## Deploy Cipher

Insert a screenshot below of the checksum verification.



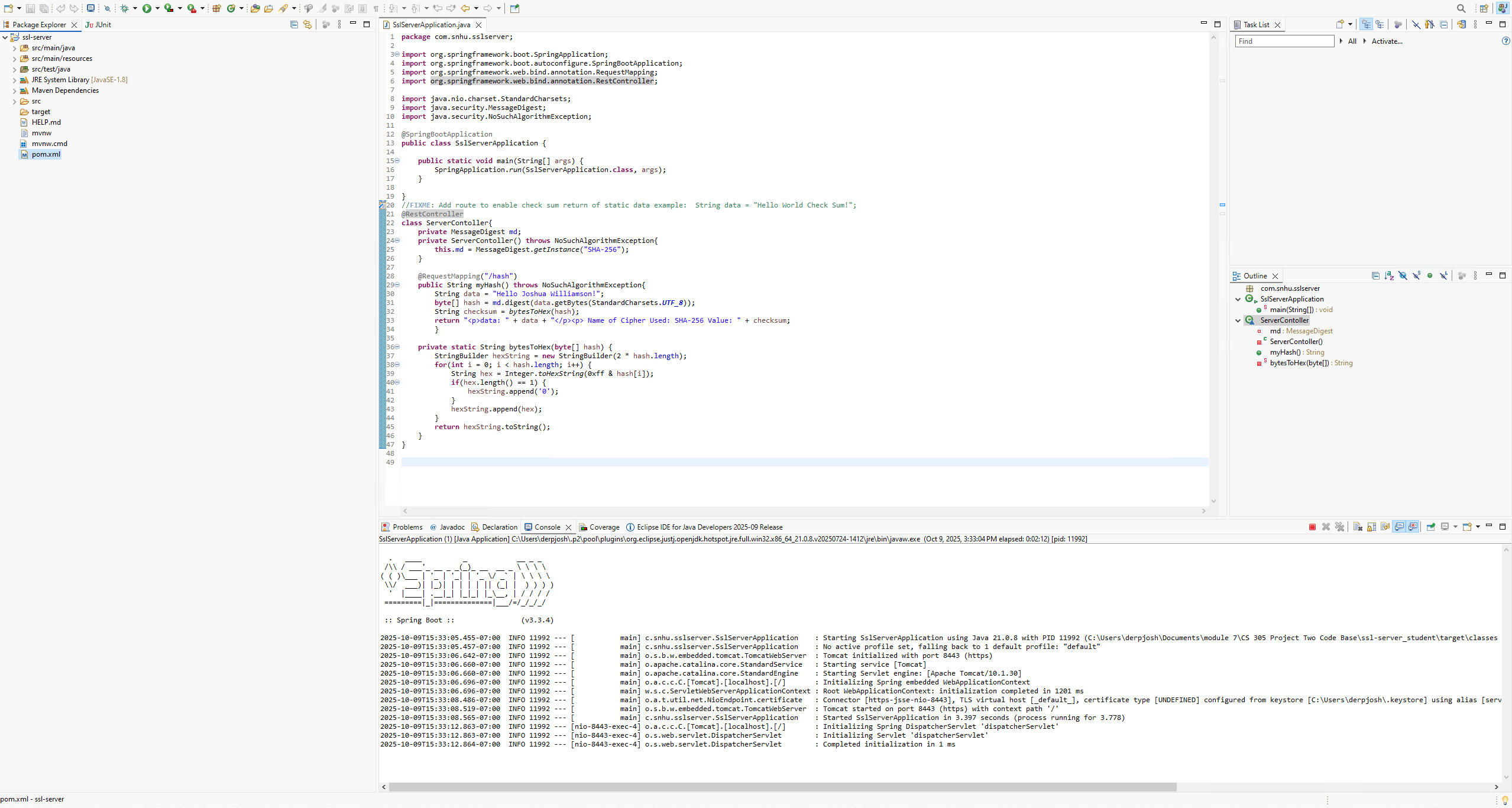
## Secure Communications

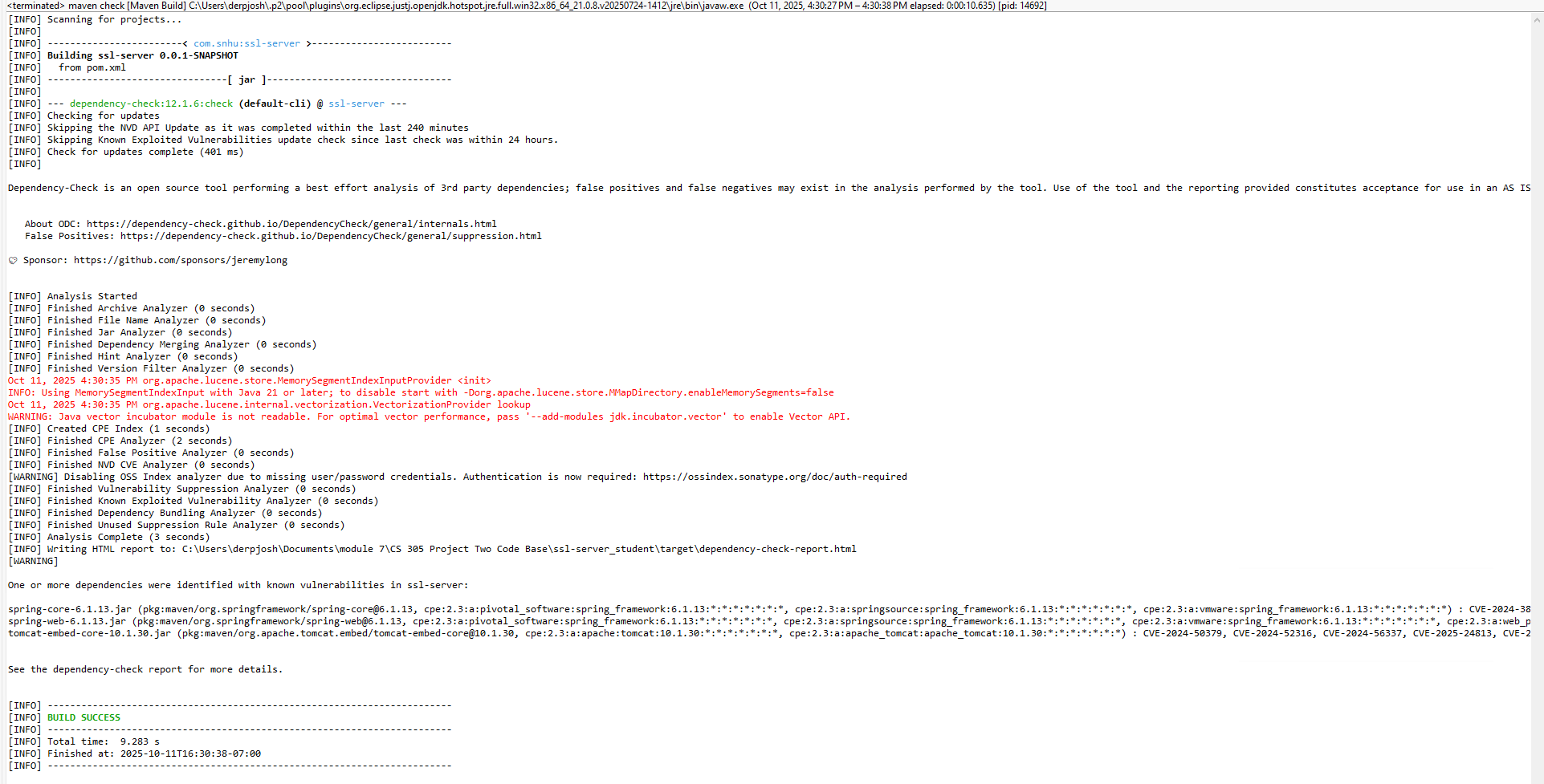
Insert a screenshot below of the web browser that shows a secure webpage.

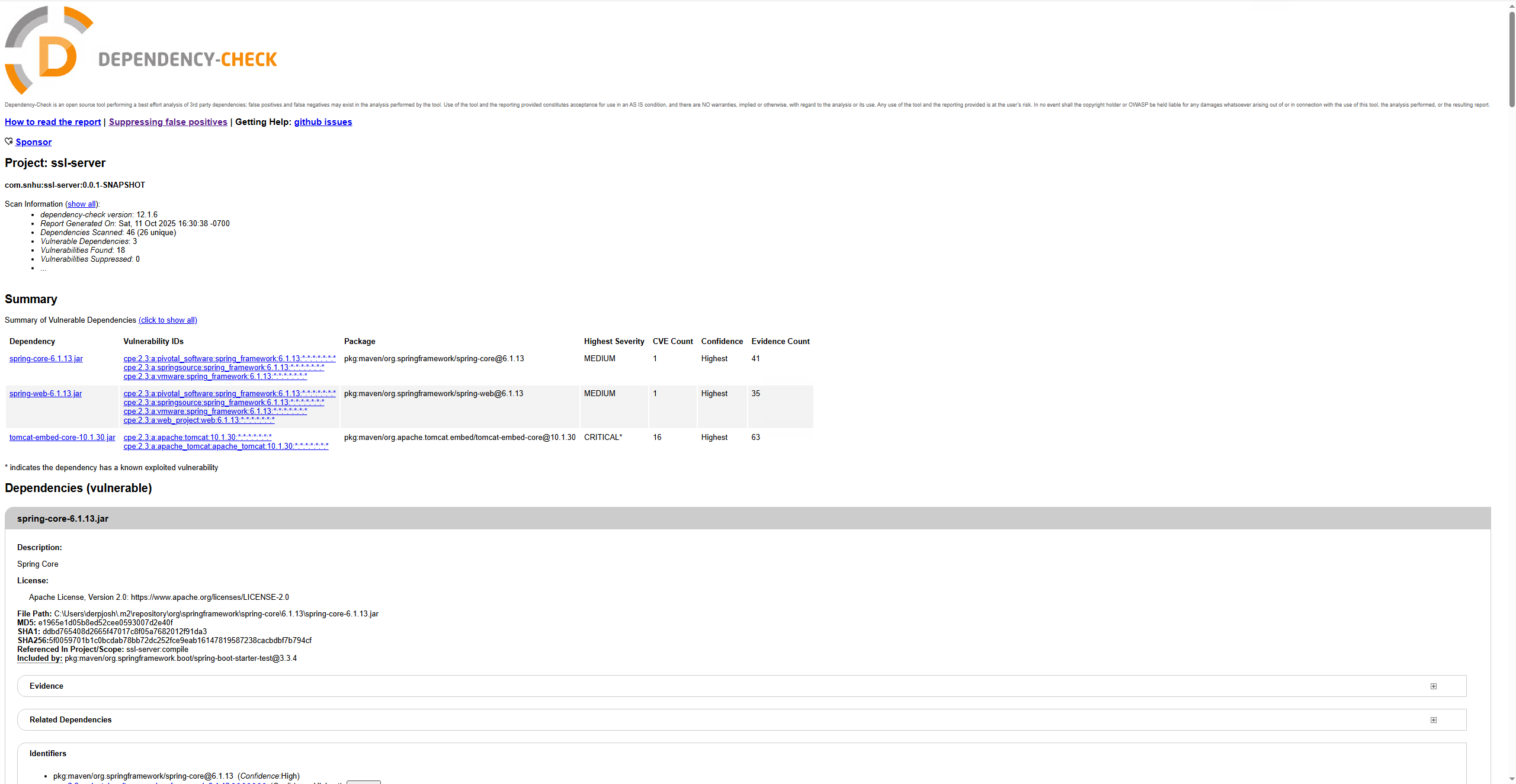


## Secondary Testing

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



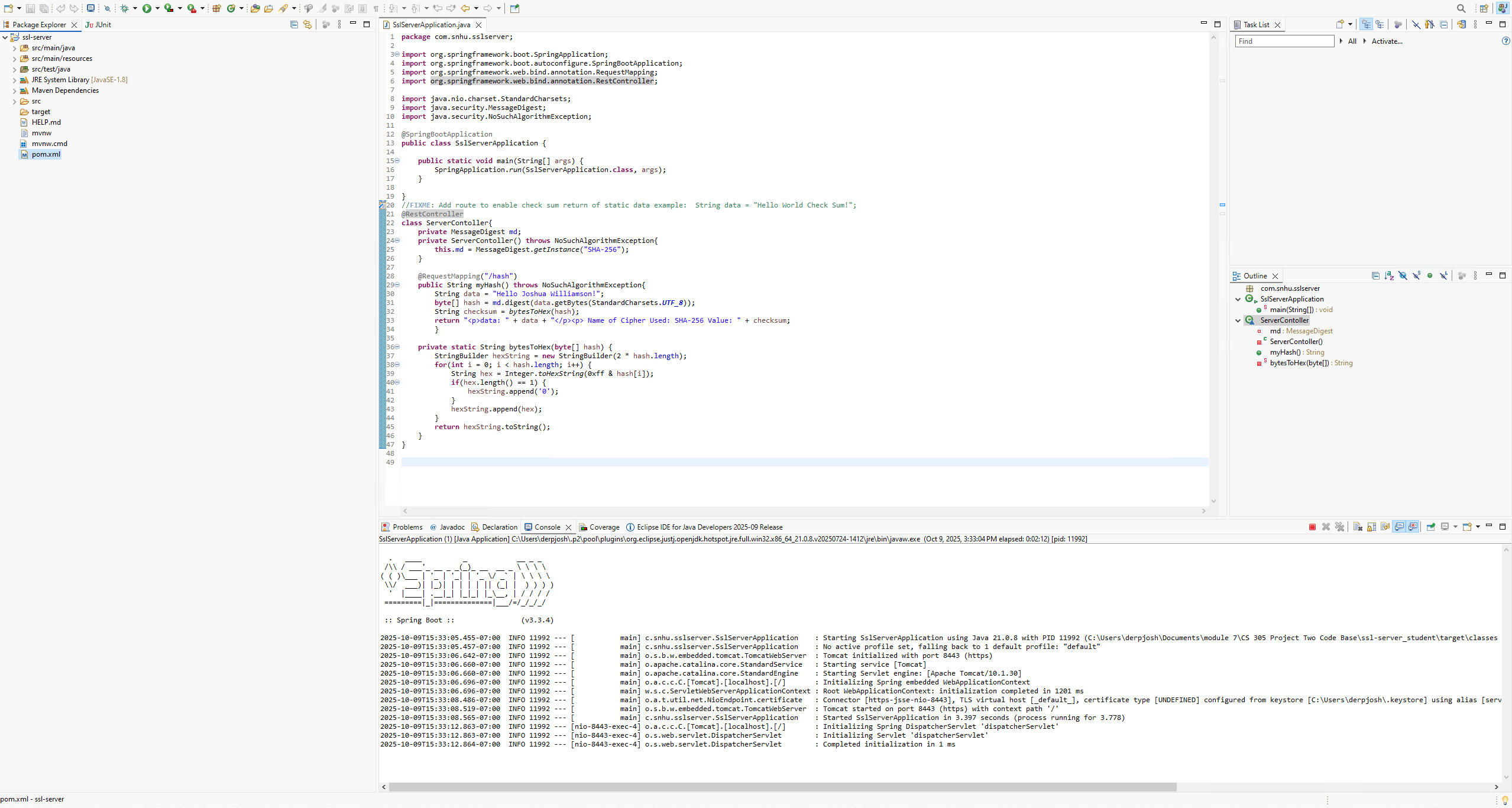




## Functional Testing

Insert a screenshot below of the

refactored code executed without errors.



## Summary

Throughout the process, the program was reviewed for potential coding errors and refined to promote secure error handling and consistent coding practices. Cryptography was also incorporated to enhance data protection and integrity.

The program accepts a string input from the user, which is then encrypted using industry-standard encryption methods before generating a checksum for validation. Because the purpose of the program is to encrypt any user-provided input, additional input validation was not required. However, the security implications of accepting user input were still taken into account during development.

A final area of focus involved securing the API. When creating the checksum API, steps were taken to ensure that all interactions remained safe and reliable. The API receives a single string input, encrypts it, and returns the checksum value, demonstrating a secure and efficient approach to handling user data.

## Industry Standard Best Practices

To maintain a strong level of security within this application, several industry-standard best practices were implemented. The process began with a static analysis of system dependencies to identify potential vulnerabilities. Additional code reviews were then conducted to detect and address any security concerns within the controllers, views, and APIs. By adhering to established secure coding practices, the risk of vulnerabilities and security threats was significantly reduced.