****

# 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** | **[Date]** | **[Your Name]** |  |

## 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

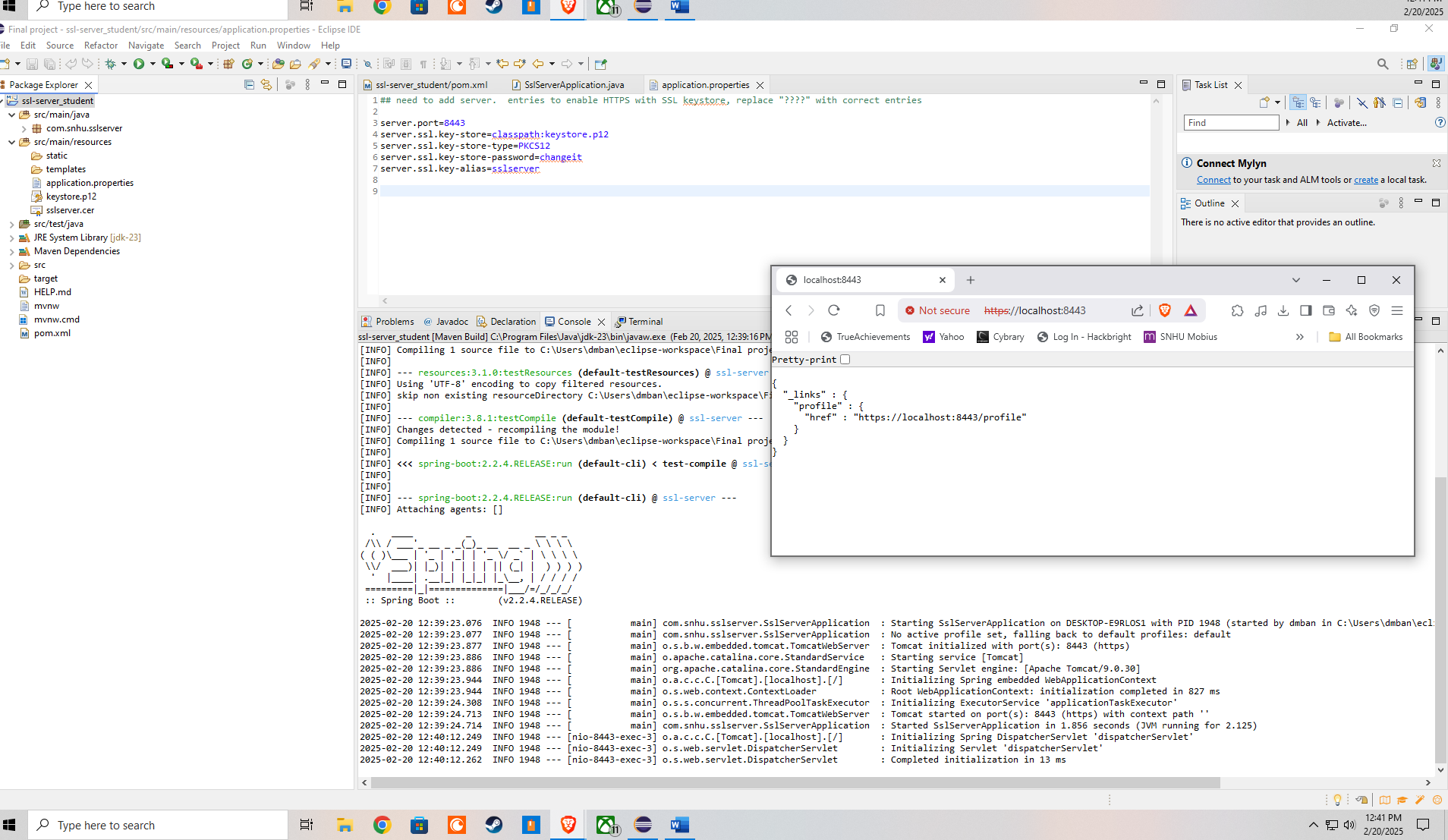
Zeppelin041

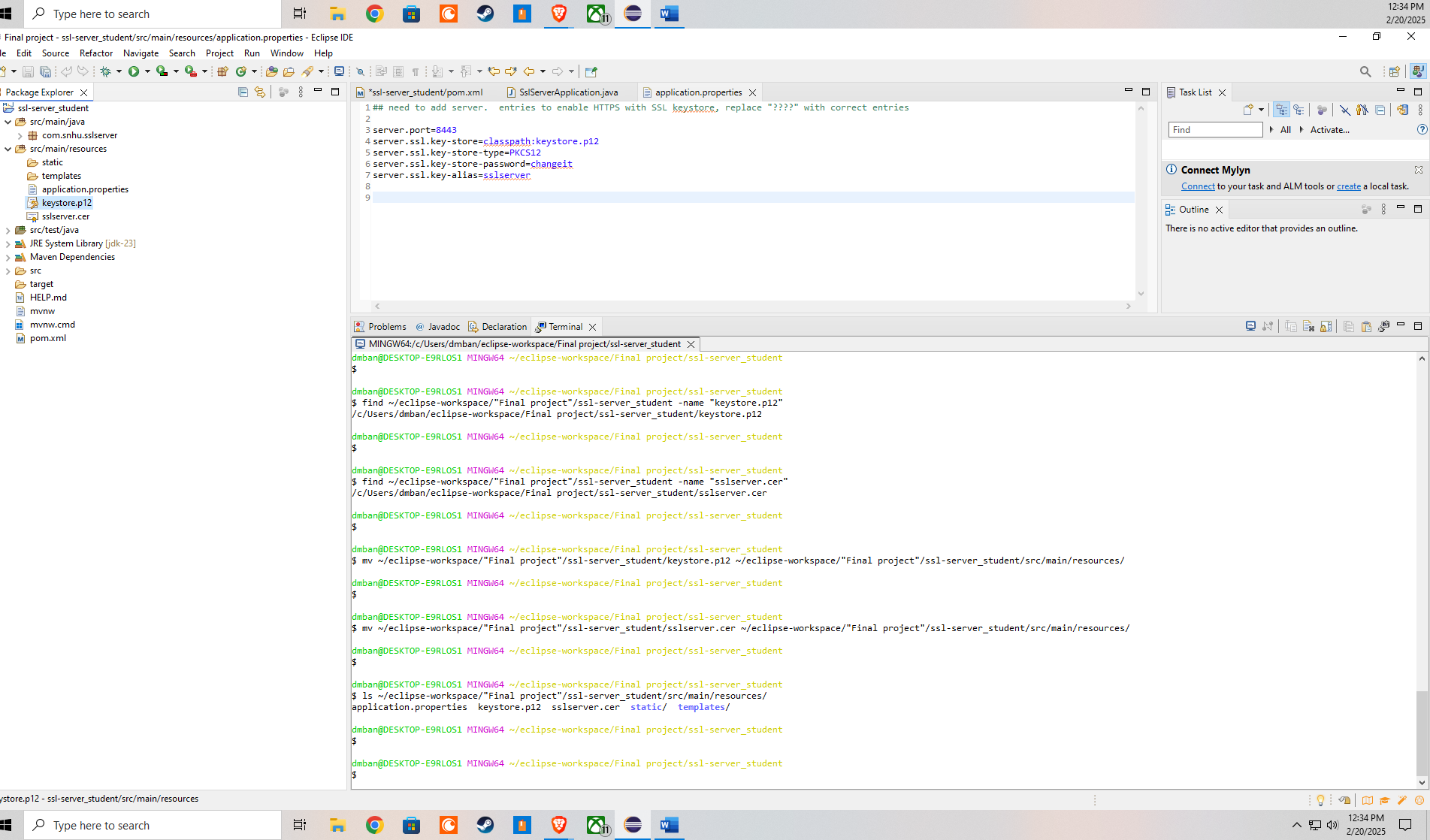
## Algorithm Cipher

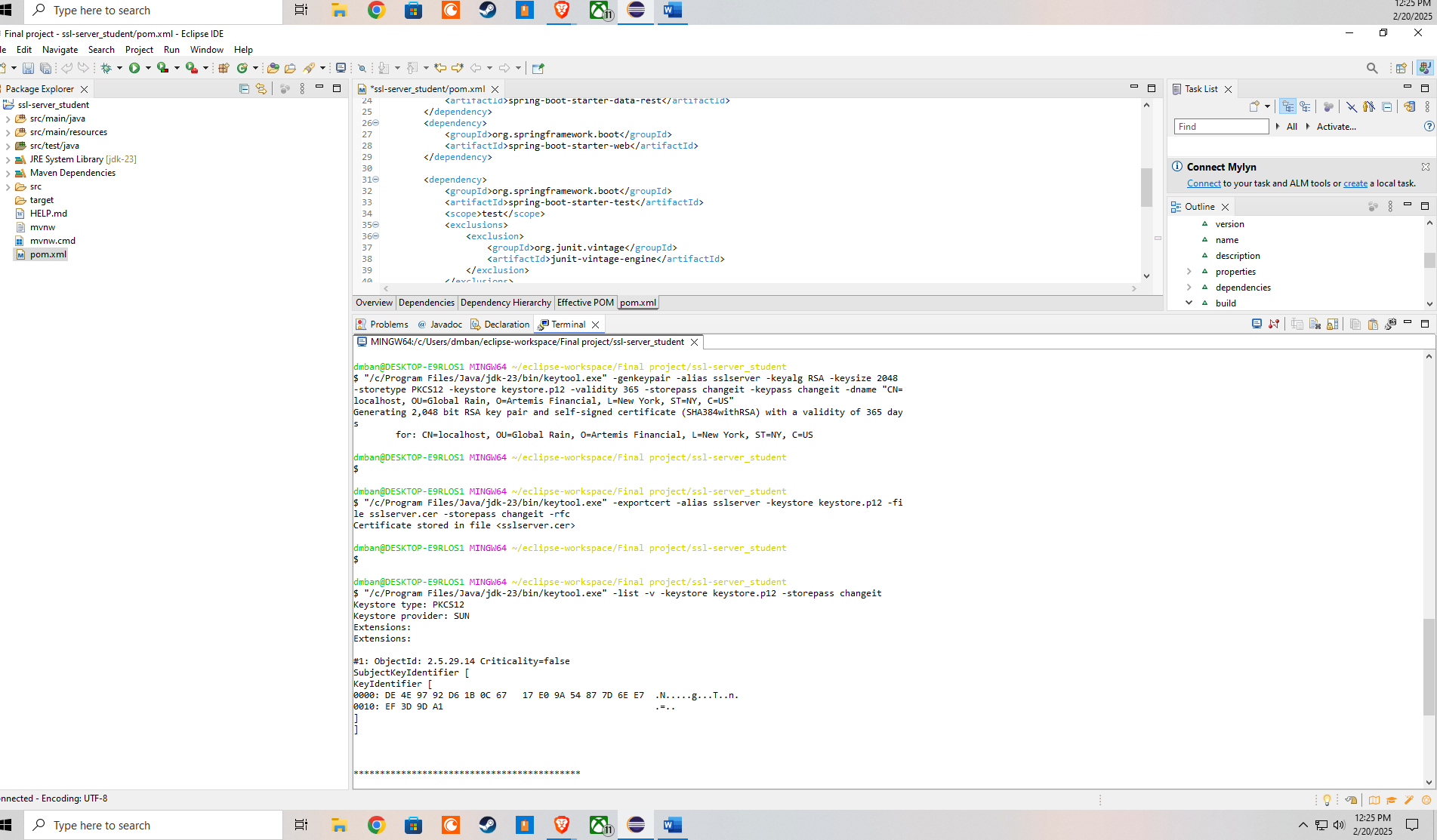
To keep Artemis Financial’s client data safe, we need to use strong encryption methods. Encryption is like putting data inside a secure lockbox that only the right key can open. For this project, I recommend using AES-256 for encrypting data and SHA-256 for verifying its integrity. AES-256 (Advanced Encryption Standard with a 256-bit key) is a widely used encryption method that scrambles data so that only someone with the correct secret key can read it. It works like a secure vault that requires a specific combination to unlock. Since it uses a 256-bit key, which is extremely long, it would take billions of years for hackers to break using brute force. Banks, government agencies, and secure messaging apps like WhatsApp use AES-256 to keep information safe.

For verifying data integrity, we will use SHA-256 (Secure Hash Algorithm - 256-bit). This is not used to hide data like AES, but rather to check if it has been changed. SHA-256 works like a fingerprint for a file—if even one tiny piece of the file is changed, the fingerprint (hash) will be completely different. For example, when downloading software, the website might provide a SHA-256 hash so users can verify that the file hasn’t been tampered with by hackers. Both AES-256 and SHA-256 play important roles in cybersecurity today. AES-256 is used to protect confidential information, while SHA-256 is used to verify data integrity in applications like Bitcoin and SSL/TLS certificates for secure websites. By implementing these in Artemis Financial’s system, we can ensure that sensitive financial data is both encrypted and verified, reducing the risk of cyberattacks.

## Certificate Generation

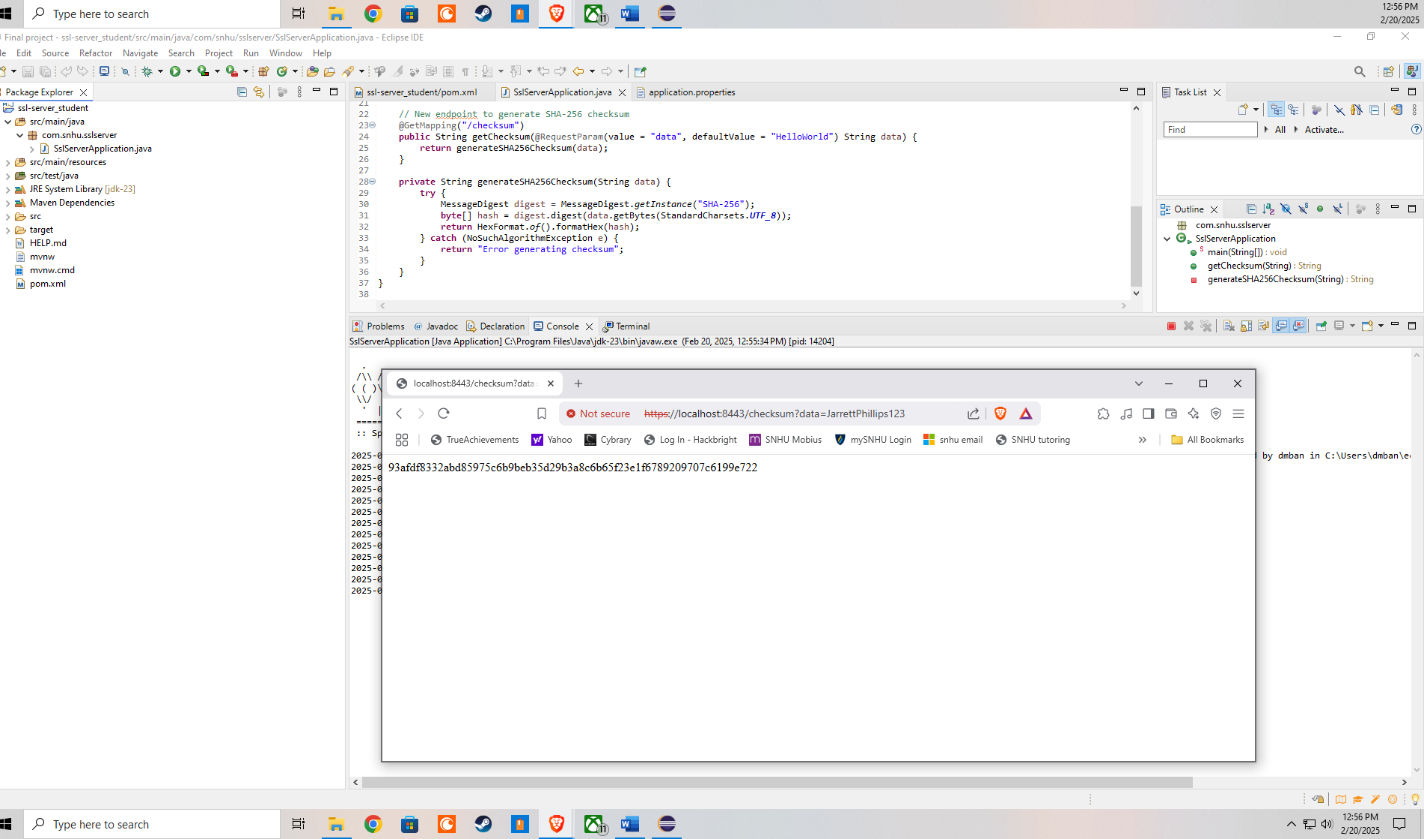
First, I must mention the commands did not want to work for me 80% of the time due to many typos. For future references I will remember to really look over this and decided to add this to the project to show how sensitive things can be, it also did not help that I did it backwards and forgot to check the application file first and change that, leaving me to have to figure out the commands to add the files to the project which I will also add screen shots of:  
  






## Deploy Cipher

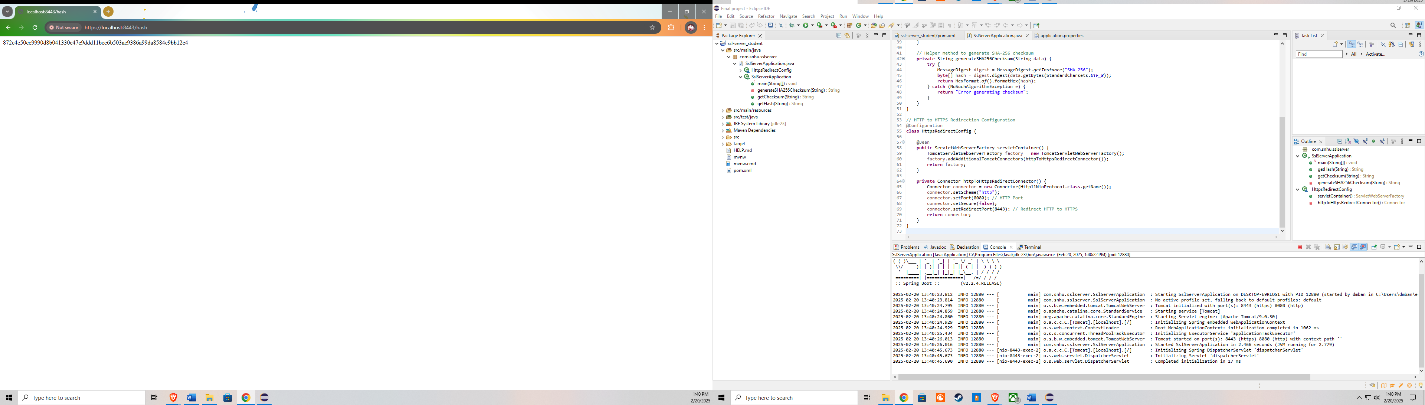
Insert a screenshot below of the checksum verification.



## Secure Communications

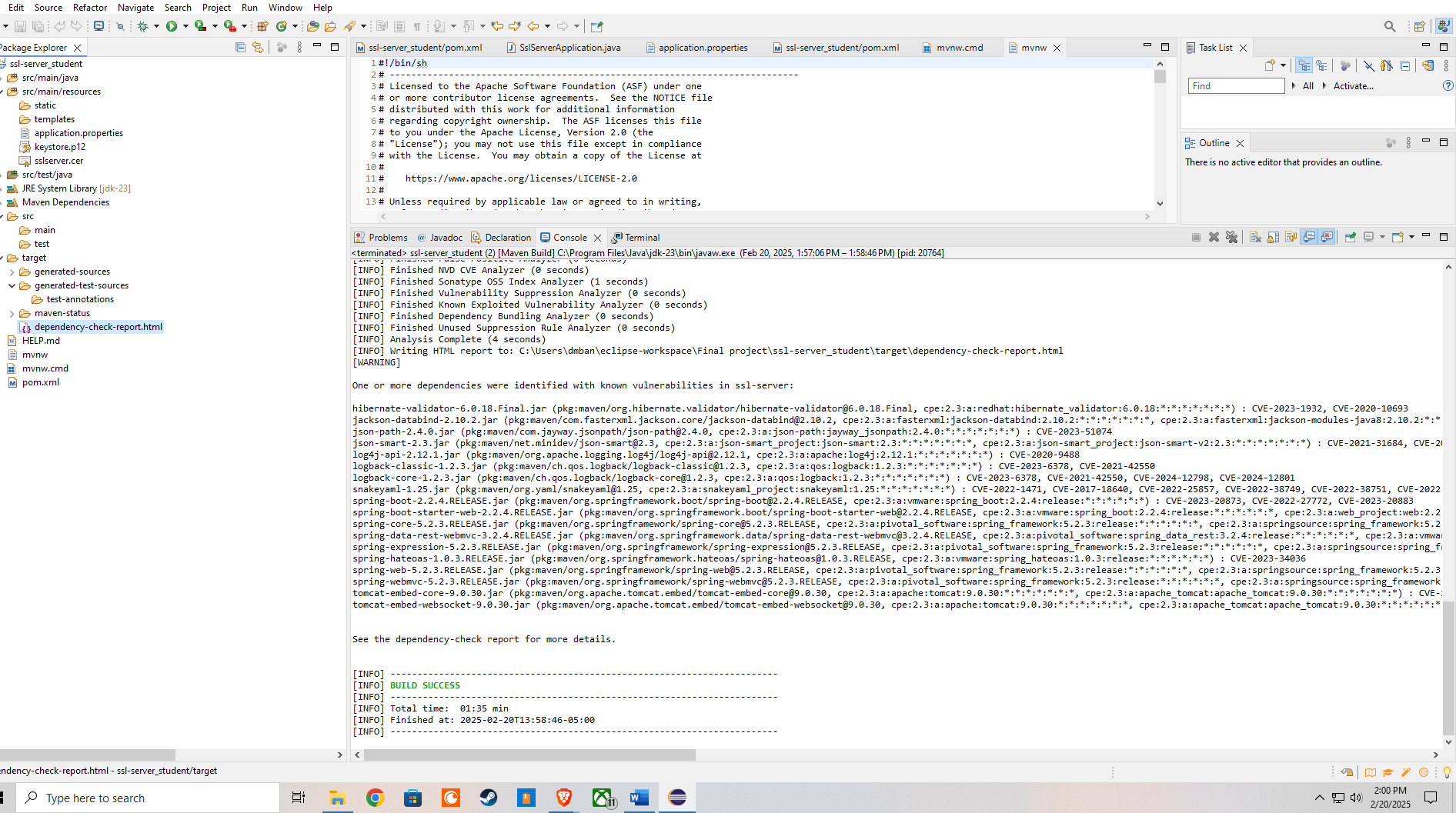
Insert a screenshot below of the web browser that shows a secure webpage.

I have tried a bunch of different things to get it to show a secure connection, I even updated my certificates on my computer and windows decided to warn me that I am taking a huge security risk and refuses to let me even do that, I have the proper code running to implement it as secure, I am not sure why this is doing this to me and why windows has to act this way, its also stating that is purely a web browser issue as well. I have tried for hours to get it to say secure even though it is running https, you can see that right in the running window.

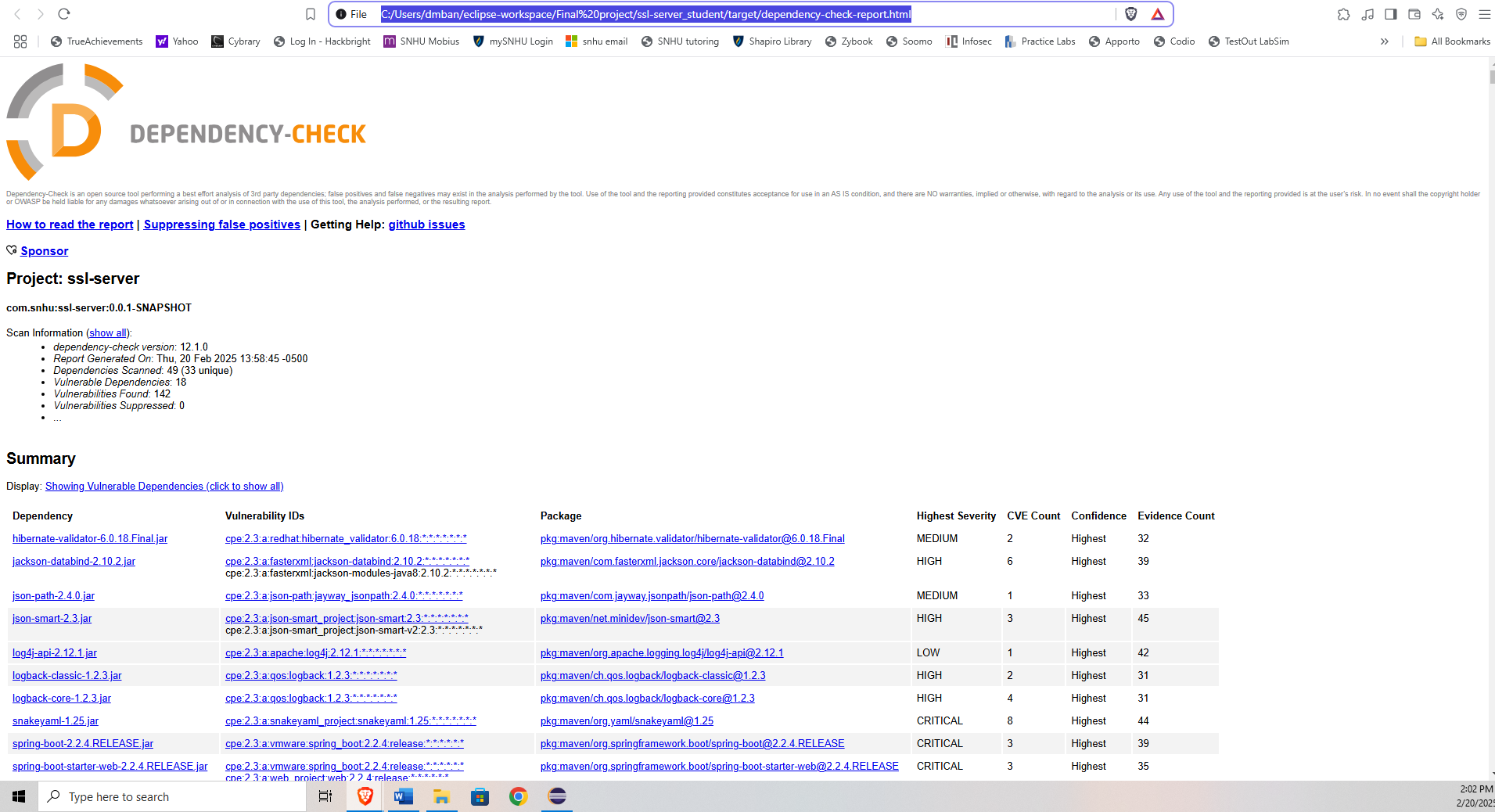


## Secondary Testing

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

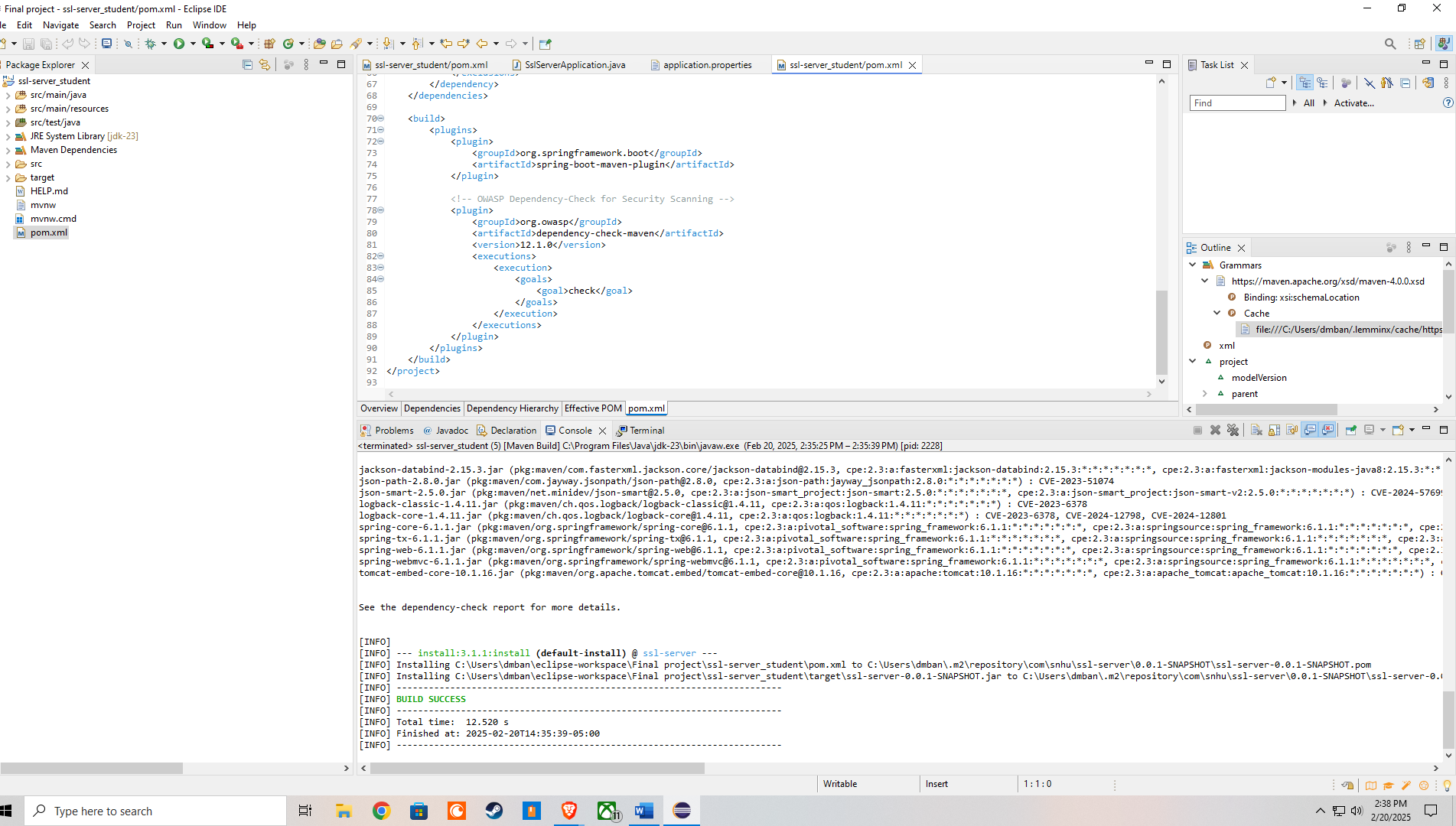


I will add the HTML as well: <file:///C:/Users/dmban/eclipse-workspace/Final%20project/ssl-server_student/target/dependency-check-report.html>



## Functional Testing

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



## Summary

As part of the project, I refactored the existing code to improve its security and ensure it met modern security testing standards. The vulnerability assessment process flow helped identify key security areas that needed improvement, including input validation, encryption, secure communication, and dependency management. One major change was upgrading the project to use HTTPS instead of HTTP, which protects data while it is being transferred over the internet (although I had browser errors that refused to use certificates I create myself). To achieve this, I generated a self-signed SSL certificate using Java Keytool and configured the application to enforce encrypted communication.

Another important step was improving the security of stored and transmitted data by adding a SHA-256 checksum function. This ensures that data has not been altered during transfer, providing data integrity. Additionally, I scanned the application for vulnerabilities using OWASP Dependency-Check, which flagged outdated and insecure libraries. By updating dependencies in the pom.xml file, I removed known security risks and ensured that the application runs on patched and secure software components. These changes helped strengthen the application’s security while keeping it functional and efficient. However, some vulnerabilities still remain, particularly in Spring Framework, Logback, and Tomcat, even in their latest available versions. These risks will need to be monitored for future patches and updates. Despite these remaining vulnerabilities, the refactored code significantly strengthens the security posture of the application while ensuring continued functionality.

## Industry Standard Best Practices

To maintain security in the software, I followed industry standard best practices recommended by OWASP, NIST, and other cybersecurity organizations. One of the most important practices was keeping dependencies up to date. Outdated libraries often have known vulnerabilities, so I updated components like Spring Boot, Jackson Databind, and Hibernate Validator to their latest secure versions. I also used encryption and secure communication protocols (TLS/SSL) to protect sensitive financial data, reducing the risk of cyberattacks. Applying secure coding practices is essential for a company’s overall well-being because it helps prevent data breaches, protects client information, and builds trust with customers. A financial company like Artemis Financial deals with sensitive data, so having strong security ensures that client information stays safe. By implementing layered security measures, such as input validation, encryption, and secure authentication, the application is more resilient against attacks. Following these best practices not only meets compliance standards but also reduces potential legal and financial risks for the company.

References

OWASP Foundation. (2023). OWASP secure coding practices quick reference guide.

Open Web Application Security Project (OWASP).

<https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/>

National Institute of Standards and Technology (NIST). (2023).

Framework for improving critical infrastructure cybersecurity, version 2.0. U.S. Department of Commerce. <https://www.nist.gov/cyberframework>

OWASP Foundation. (2024). OWASP Dependency-Check documentation.

Open Web Application Security Project (OWASP).

<https://jeremylong.github.io/DependencyCheck/>

Oracle Corporation. (2024). Java security overview and Java Keytool usage.

<https://docs.oracle.com/en/java/javase/>