****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.y4p5hqloapk2)

[**Client 3**](#_heading=h.kbcddkdqppu)

[**Instructions 3**](#_heading=h.fbezn74bhp1v)

[**Developer 4**](#_heading=h.6wrzwempbnu1)

[**1. Algorithm Cipher 4**](#_heading=h.l10pwfoe0l3p)

[**2. Certificate Generation 4**](#_heading=h.bhrutaxxgcv5)

[**3. Deploy Cipher 4**](#_heading=h.1iyryxr8kk1j)

[**4. Secure Communications 4**](#_heading=h.nvp2mcx9xgcf)

[**5. Secondary Testing 4**](#_heading=h.xjfqfaxjdi66)

[**6. Functional Testing 4**](#_heading=h.uac4mhms2zq3)

[**7. Summary 4**](#_heading=h.67f40k5bpuhg)

[**8. Industry Standard Best Practices 4**](#_heading=h.wivx4vdxyblf)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **4/16/2025** | **Marissa Rhine** |  |

## 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

Marissa Rhine

## Algorithm Cipher: AES-256

The selection of AES-256-GCM for Artemis Financial's archive encryption is a well-founded decision based on a strong historical context, robust cryptographic principles, and the current encryption standards. AES's strength against brute-force attacks, particularly with a 256-bit key, its efficient performance, and the added benefits of data authenticity and integrity provided by the GCM mode make it an ideal choice for safeguarding sensitive financial data in the long term. Understanding the evolution of encryption and the specific mechanisms of AES-GCM underscores its suitability for meeting the stringent security and regulatory requirements of a financial institution operating in a complex and increasingly digital world.

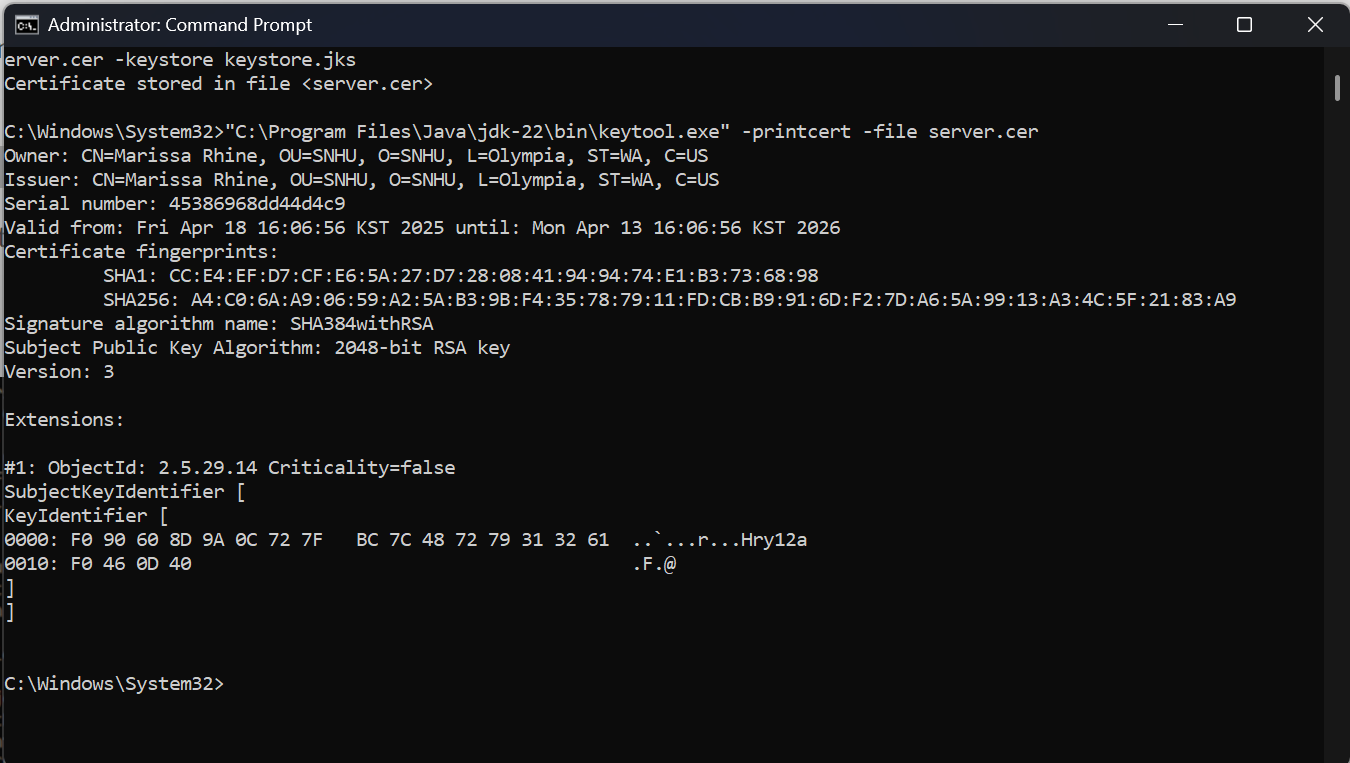
At a high level, an encryption algorithm cipher is a well-defined set of steps used to transform readable data into an unreadable format. This transformation process aims to ensure that only authorized parties, possessing the correct key, can reverse the process and recover the original plaintext. The integrity of encrypted data can be further enhanced through the use of hash functions. These are one-way functions that take an input of arbitrary length and produce a fixed-size output, known as a hash or message digest. A good hash function exhibits collision resistance, meaning it is computationally infeasible to find two different inputs that produce the same hash value. While not reversible like encryption, hash functions provide a digital fingerprint of the data. In the context of AES-GCM, the GCM mode incorporates a cryptographic hash to provide data authenticity, ensuring that any tampering with the ciphertext will result in a different hash value, alerting the receiver to potential security violations. The 256-bit key in AES-256 refers to the size of the secret key used in the encryption process. A larger key size exponentially increases the number of possible keys, making brute-force attacks computationally too slow with current technology.

Modern encryption algorithms often leverage random numbers to enhance security. In AES-GCM, a crucial element is the Initialization Vector (IV), a random number that is used only once for each encryption operation with the same key. The IV ensures that even if the same plaintext is encrypted multiple times, the resulting ciphertext will be different, preventing attackers from identifying patterns. AES is a symmetric algorithm, meaning it uses the same secret key for both encryption and decryption. This necessitates secure key management practices as the confidentiality of the data directly relies on the secrecy of this unique key. In contrast, non-symmetric or public-key cryptography uses a pair of keys: a public key for encryption and a private key for decryption. AES's efficiency, despite requiring secure key sharing, makes it well-suited for encrypting large volumes of data, such as Artemis Financial’s archive files. The operation of AES involves multiple rounds of transformations, including substitution, permutation, and mixing operations combined with the secret key. These rounds are designed to thoroughly mix the data, making the ciphertext unrecognizable.

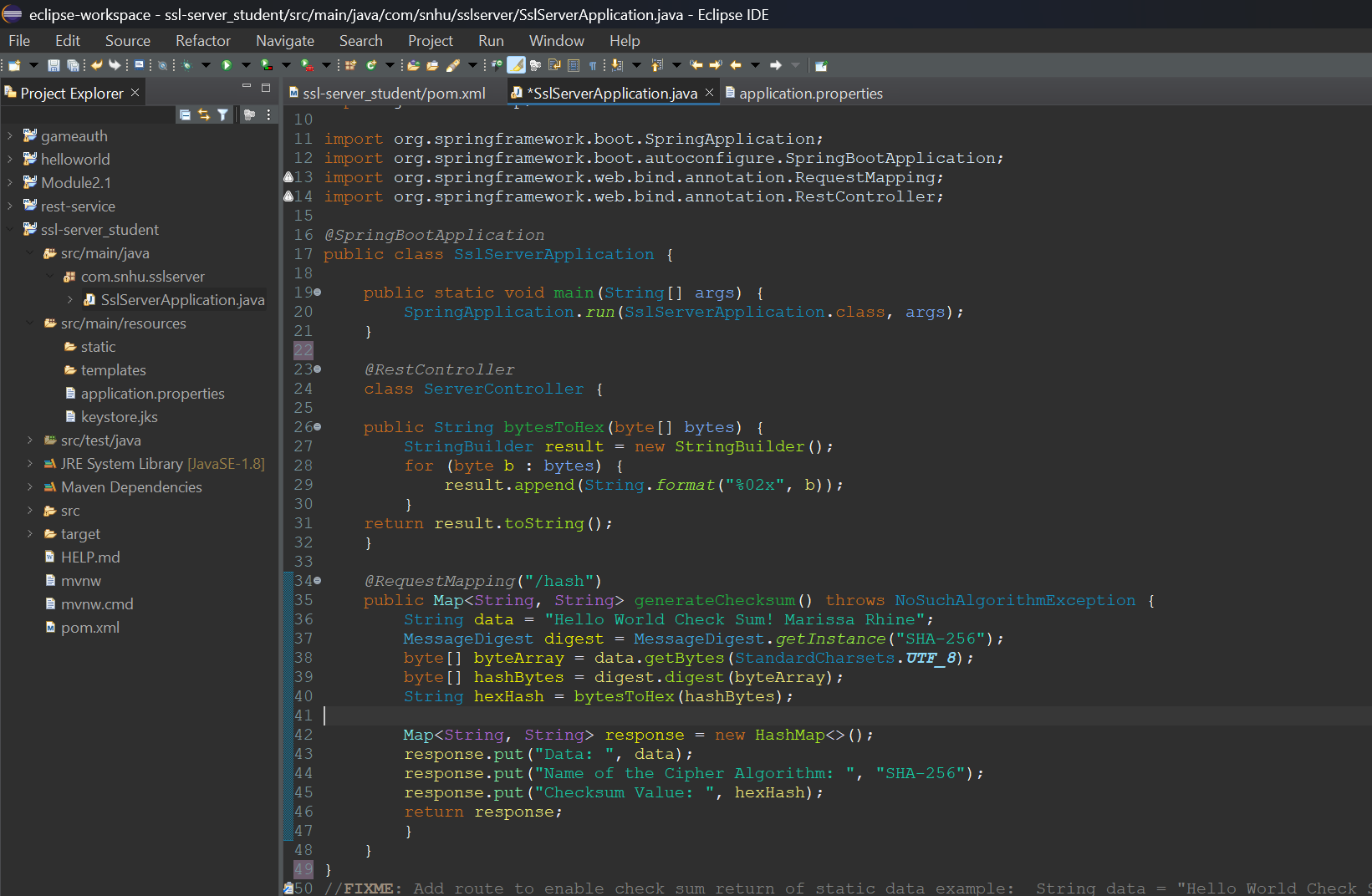
The history of encryption algorithms reveals a constant evolution driven by the need to counter increasingly sophisticated cryptographic techniques. Early methods like the simple Caesar Cipher offered minimal security and were easily broken. The advent of computers led to more complex algorithms like the Data Encryption Standard (DES). However, DES's relatively short 56-bit key became vulnerable to brute-force attacks as computing power improved. Triple DES (3DES) offered a temporary improvement but was slow and still relied on the aging DES core. Recognizing the need for a more robust and future-proof encryption standard, the National Institute of Standards and Technology (NIST) held a public competition in 1998. The Rijndael algorithm emerged as the winner and was subsequently adopted as the Advanced Encryption Standard (AES), becoming the golden standard of modern symmetric encryption.

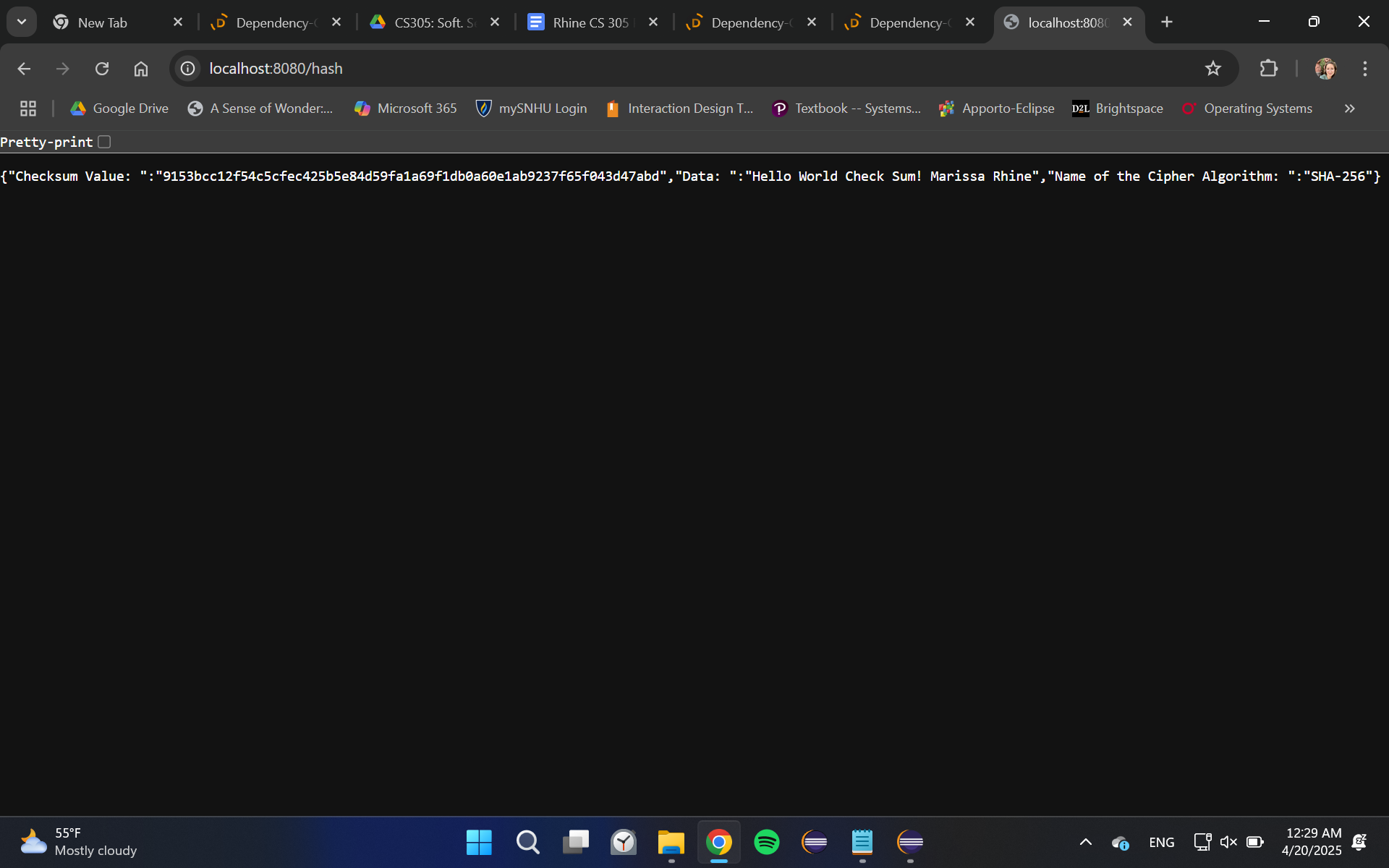
While AES is currently considered resistant to most security attacks, ongoing research focuses on defending against future quantum computers. For the time being, the wide use of AES by the cryptographic community provides ongoing opportunities for analysis, testing and further assurance of its security.

## Certificate Generation

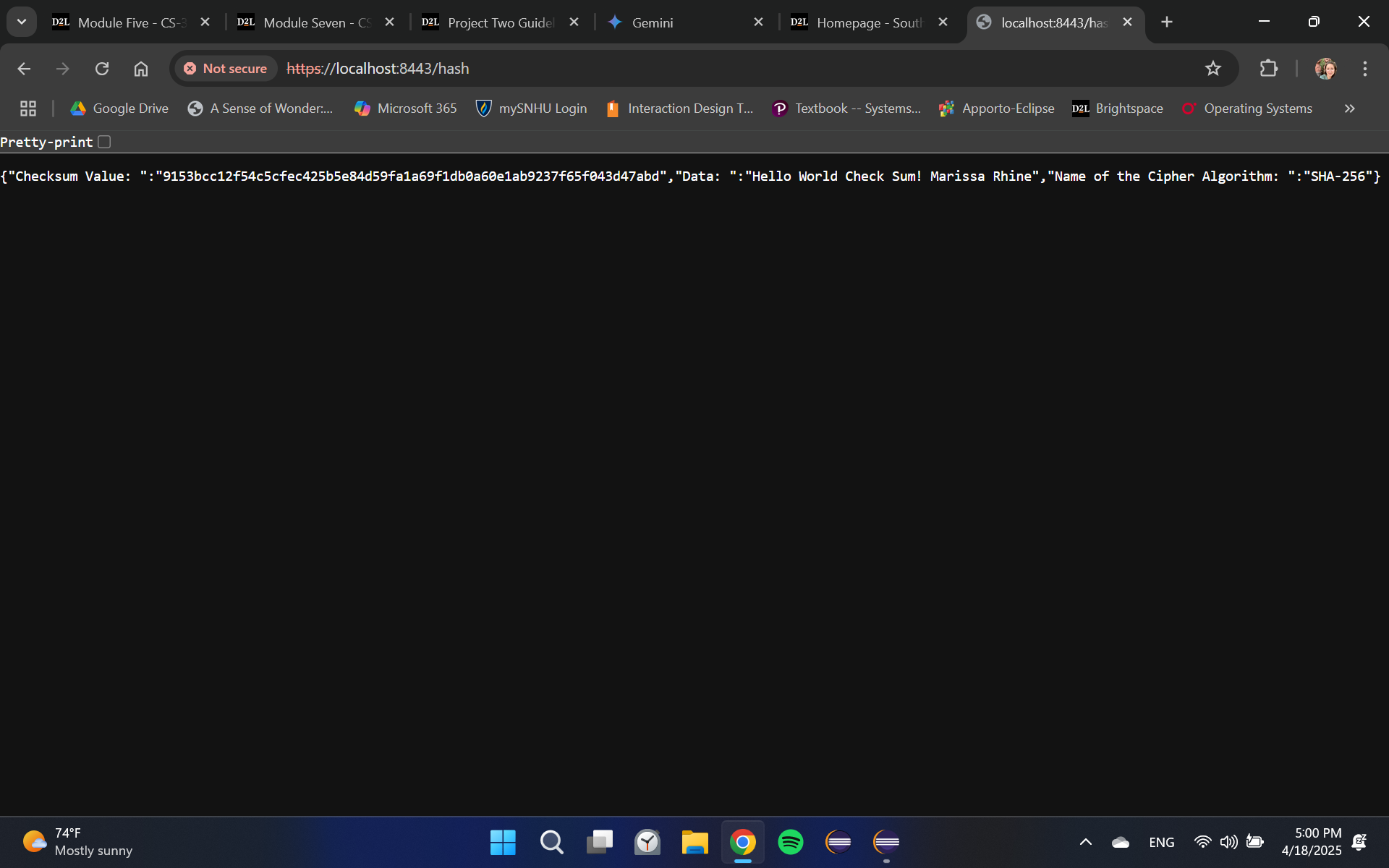


## Deploy Cipher

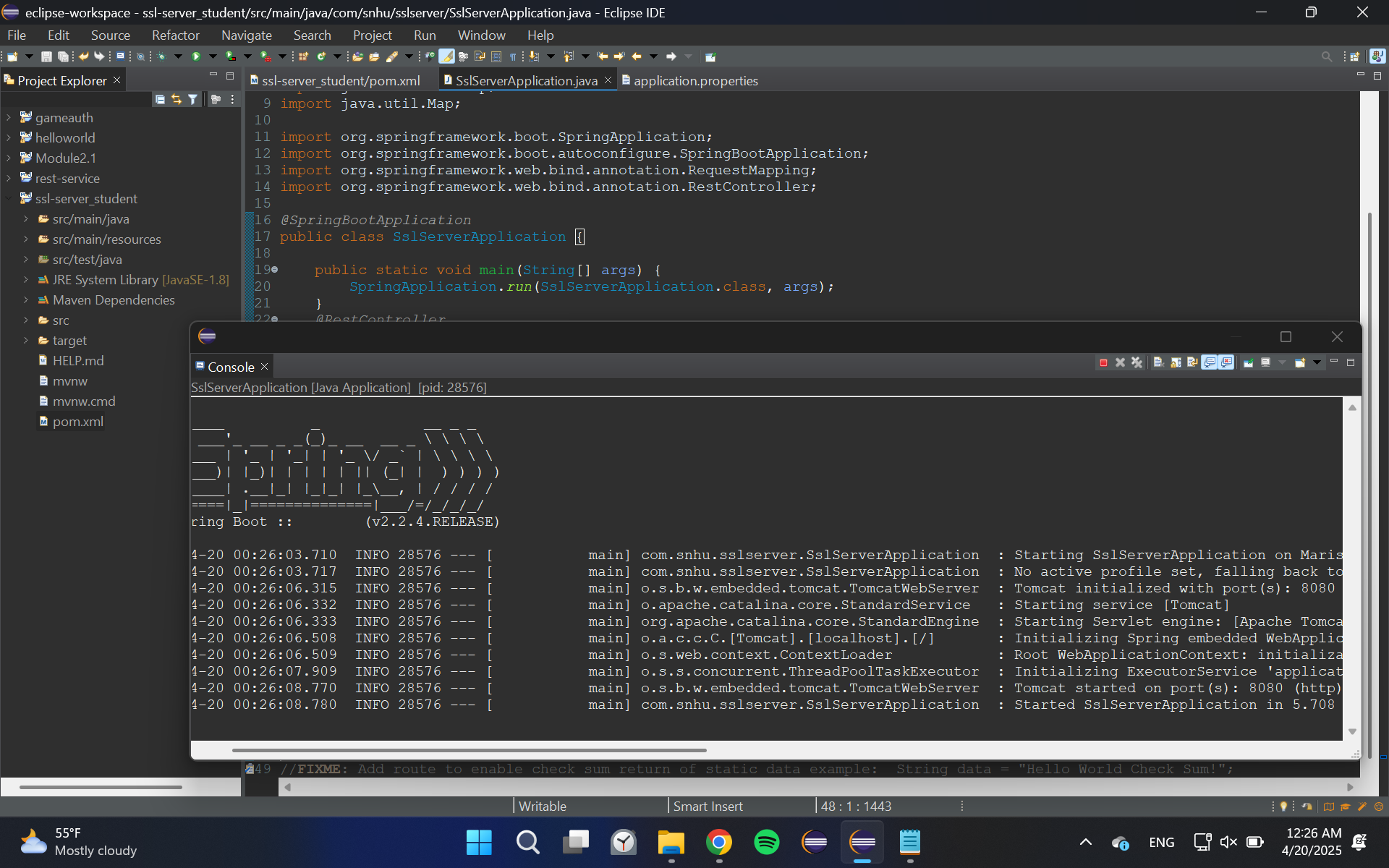


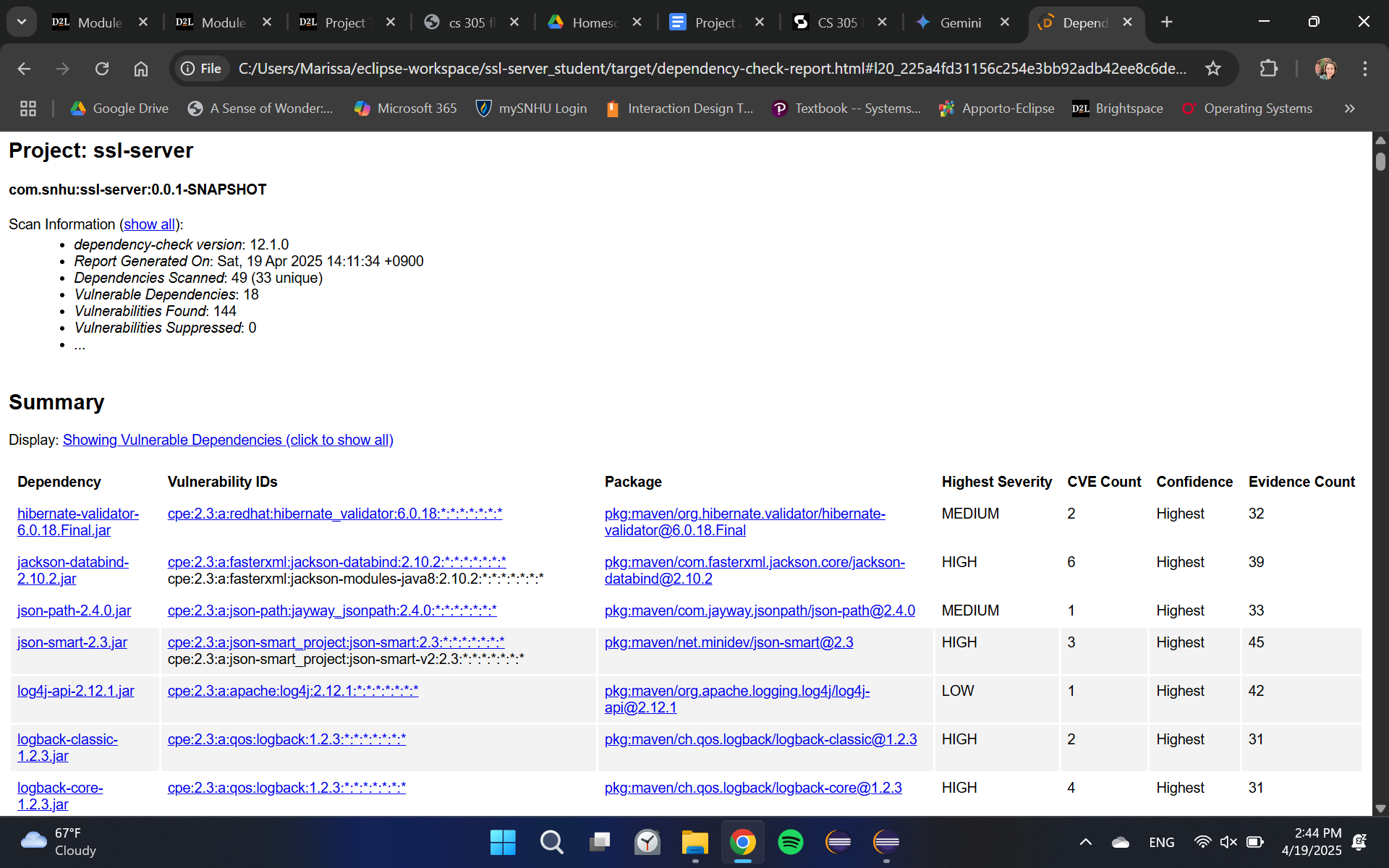


## Secure Communications

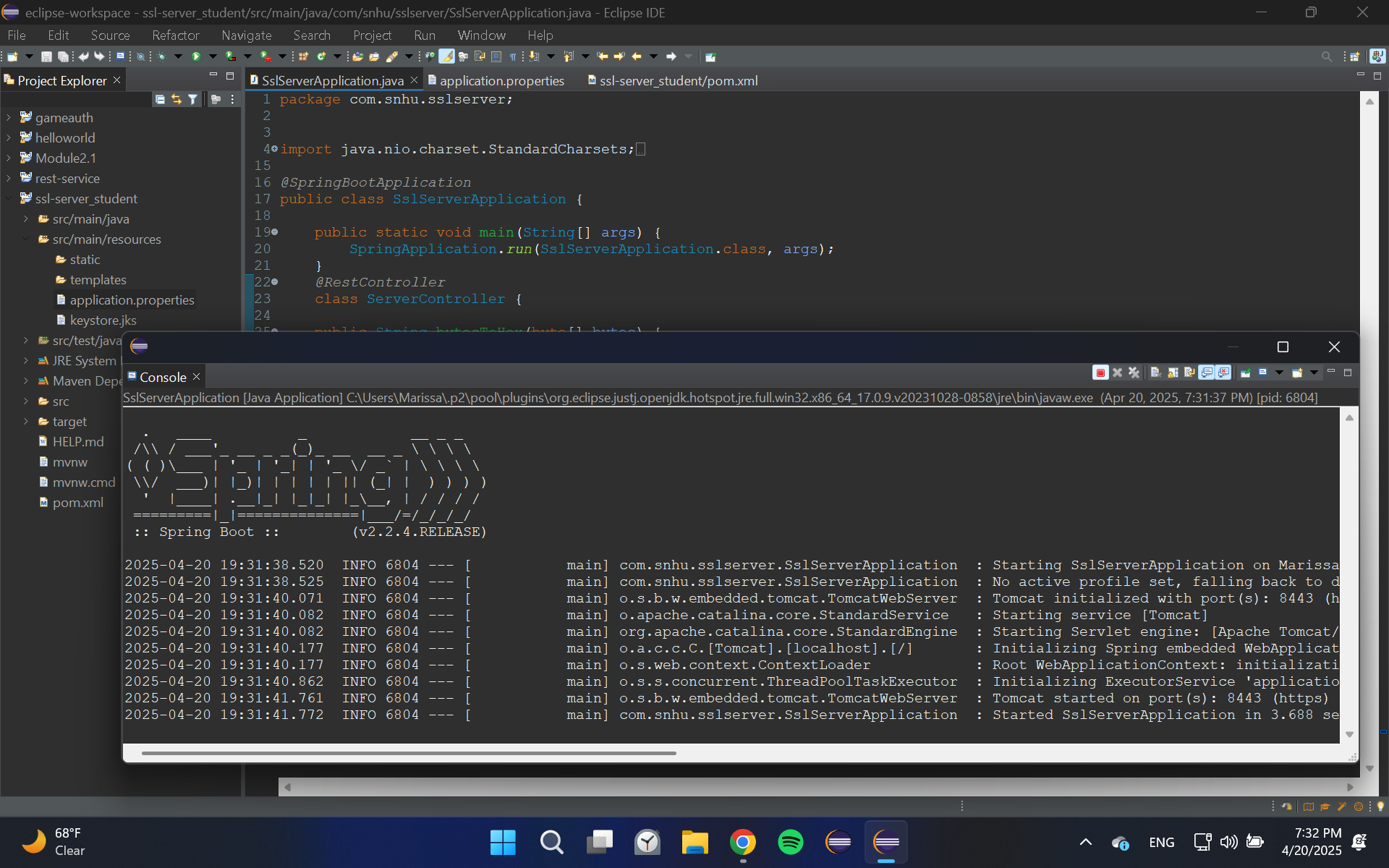


## Secondary Testing





## Functional Testing



## Summary

During our review of the SslServerApplication for Artemis Financial we’ve taken steps to review and improve the security of the program. We assessed areas of input validation, secure API interactions and encryption. First we implemented SHA256 as an encryption algorithm which is a current best practice for asymmetric data encryption. We explicitly enforced the use of HTTPS for all communication within the application and with external entities. This practice ensures that data transmitted is encrypted, protecting sensitive client information and internal communications from eavesdropping and man-in-the-middle attacks. This directly mitigates the risk of data breaches during transit, a significant concern for a financial institution handling sensitive data. We performed an OWASP dependency check as secondary testing to ensure that no new vulnerabilities were created during the refactoring of the code. Then we did a manual inspection of the code to check for additional syntactical, logical, and security vulnerabilities. This should be paired with continuing reviews of the security data and further security measures including data logging of security concerns and frequent updates to dependencies.

1. **Industry Standard Best Practices**

To address potential security vulnerabilities proactively, we implemented several industry-standard secure coding best practices. By utilizing DevSecOp we’ve integrated principles of security from the beginning. Instead of rebuilding code to add security at the end, we’ve developed code from the onset with security in mind. By testing our software vulnerabilities during dependency checks, we remain vigilant to potential vulnerabilities brought in through APIs and their libraries. By continuing to do manual security reviews and regularly updating dependencies we can ensure that we continue to meet the industry standard best practices. For a company like Artemis Financial working with financial customer information, the potential repercussions of security vulnerabilities extend beyond financial losses for clients and also include significant legal liabilities, regulatory violations, and the erosion of customer trust which continues to demonstrate the importance of designing and maintaining secure code. Security is not a single practice that can be done in one occurrence but must be repeatedly revised and integrated into the company’s mission.