****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.3ulqqfbpl9v2)

[**Client 3**](#_heading=h.paszokno46p4)

[**Instructions 3**](#_heading=h.i794koapyvnq)

[**Developer 4**](#_heading=h.pm14kxqlnkui)

[**1. Algorithm Cipher 4**](#_heading=h.3u2r2jnrvmwi)

[**2. Certificate Generation 4**](#_heading=h.glmtu8ay6kaw)

[**3. Deploy Cipher 4**](#_heading=h.synusnaw0zh7)

[**4. Secure Communications 4**](#_heading=h.xka0dnoa8v9)

[**5. Secondary Testing 4**](#_heading=h.frixne4jymjw)

[**6. Functional Testing 4**](#_heading=h.7uevl1nikzr6)

[**7. Summary 4**](#_heading=h.783cjnx9vha6)

[**8. Industry Standard Best Practices 4**](#_heading=h.kvy483cxeevs)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **Alexander Pessinis** |  |

## 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

Alexander Pessinis

## Algorithm Cipher

**Overview**

The recommended algorithm cipher used for Artemis Financial is SHA-256. SHA-256 is a highly regarded cryptographic hash function which is endorsed by NIST and is widely used in many applications. Although technically not the largest digest size (as 512 bit is offered as well), 256 offers a hash which is highly resistant to collisions and brute force attacks without compromising system performance. By using SHA-256, we can guarantee that the received data has not been altered in transit, thus preserving data integrity.

**Hash Functions and Bit Levels**

SHA-256 produces a 256 bit digest, and does so completely regardless of input size. The larger length of this provides very strong resistance against collisions, and does so without compromising performance. Greater sizes, like 512 also exist, and are technically more resistant, but the diminishing returns of using larger algorithms, along with the performance issues, led to the selection of SHA-256.

**Randomness and Symmetry**

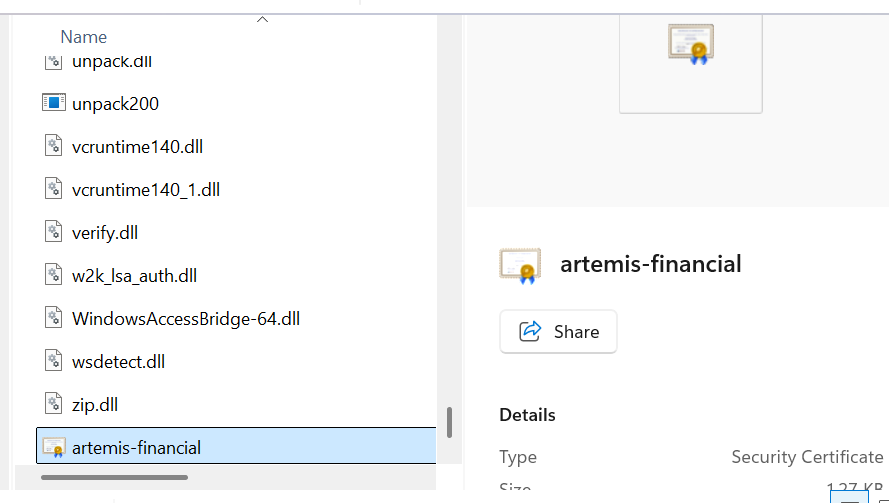
Randomness is a key issue and consideration of cryptography. The ability of these algorithms to produce random, non-reproducable functions is essential for reducing the chance of a collision. Symmetric encryption relies on the same key for encryption and decryption. While this is fine in many scenarios, it is a good idea to securely transfer the keys between parties. Asymmetric encryption uses a different key for encryption and decryption, eliminating the need for key sharing. SHA-256 does not use keys, and instead relies on cryptographic hashing.

**History**

Cryptography, since its inception thousands of years ago as a means to protect information from being read if intercepted, has evolved significantly and made its way onto cyberspace. Using algorithms and functions, data in transit as well as data at rest can be protected using encryption, which requires a key, or verified for integrity using a hash function. The SHA-256 hash function was introduced after MD-5 and SHA-1, both of which are now considered insecure due to collision risks. SHA-256 remains a trusted standard by NIST, due to its collision resistance.

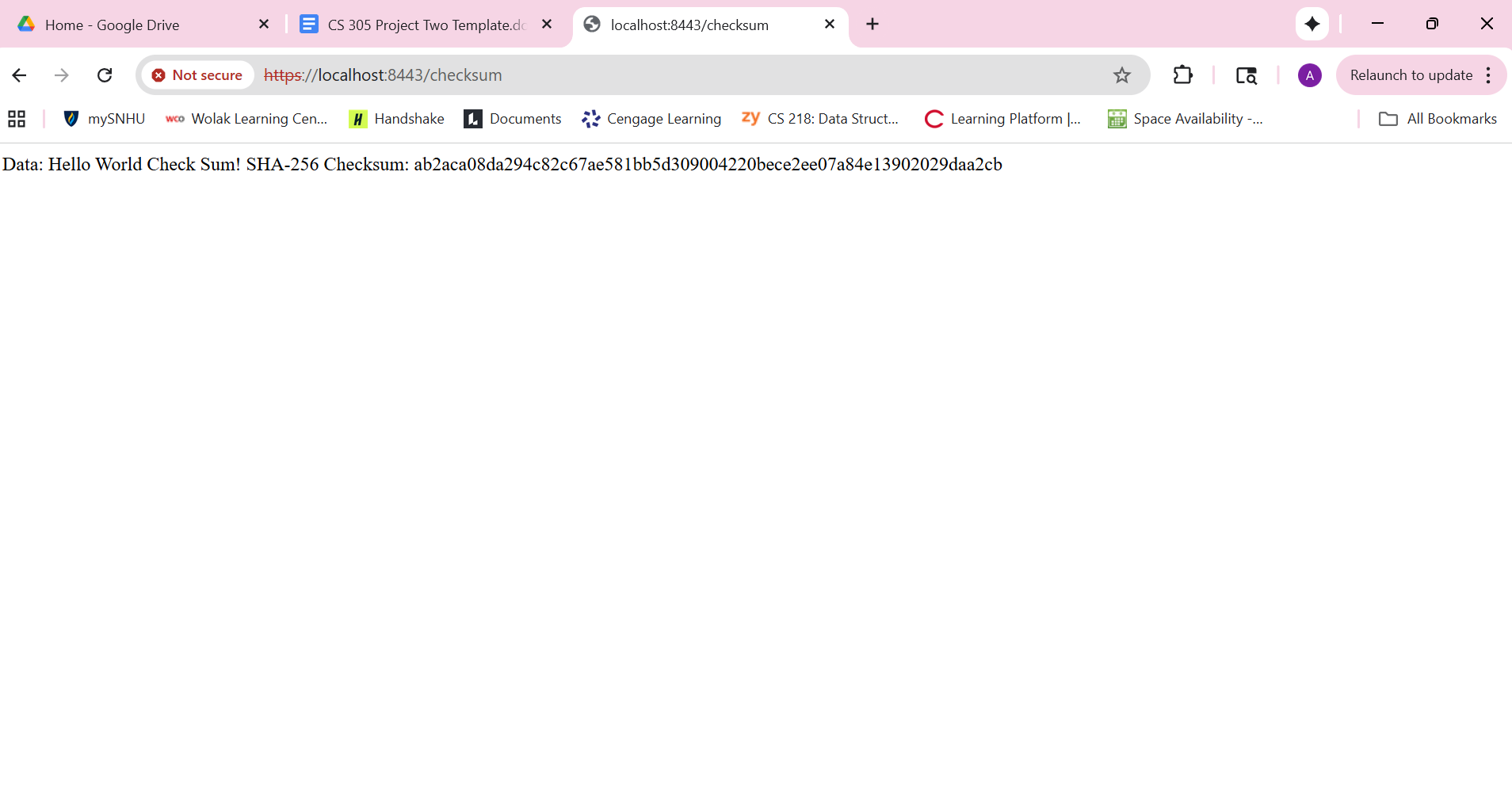
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

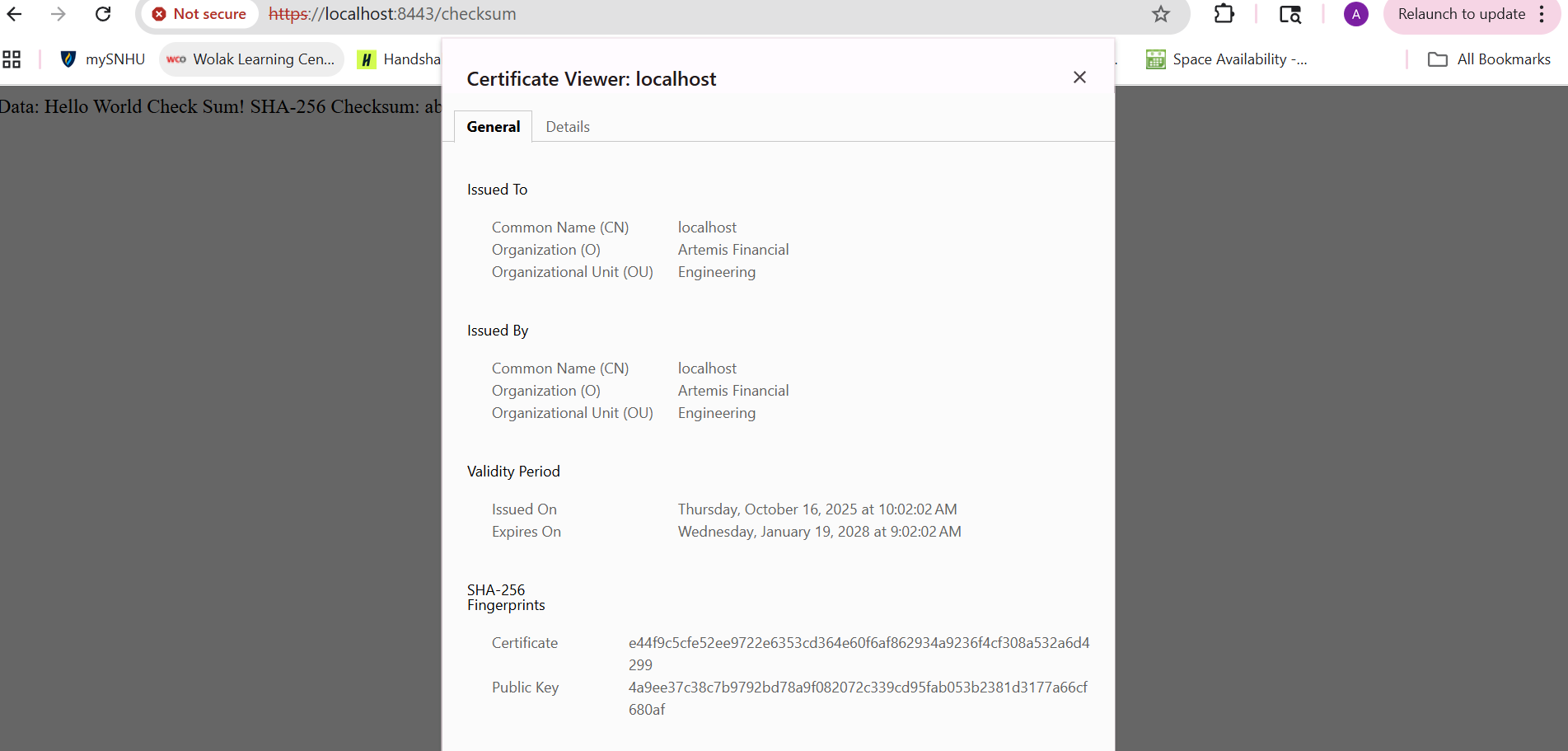
Insert a screenshot below of the checksum verification.



## Secure Communications

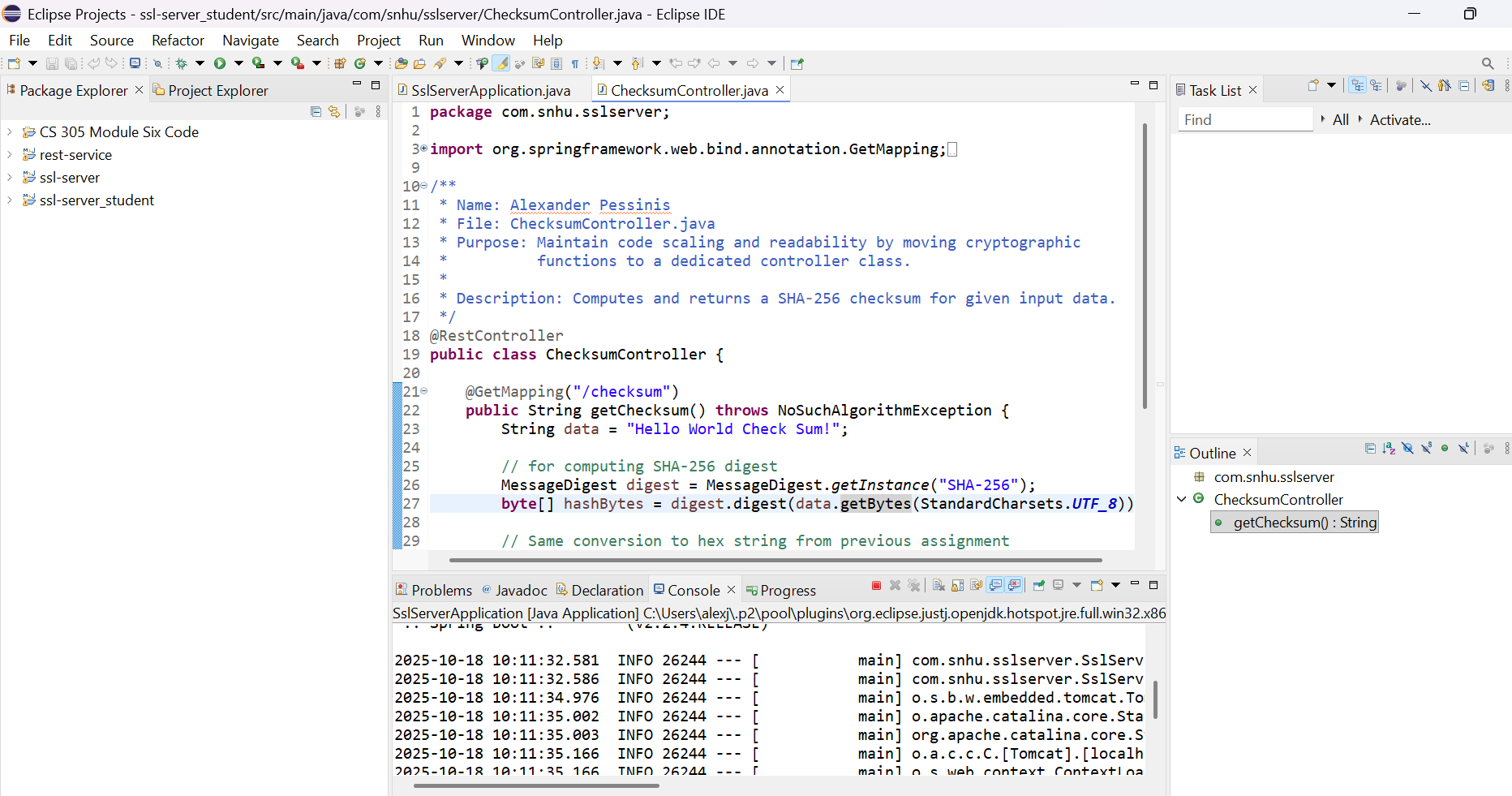
Insert a screenshot below of the web browser that shows a secure webpage.

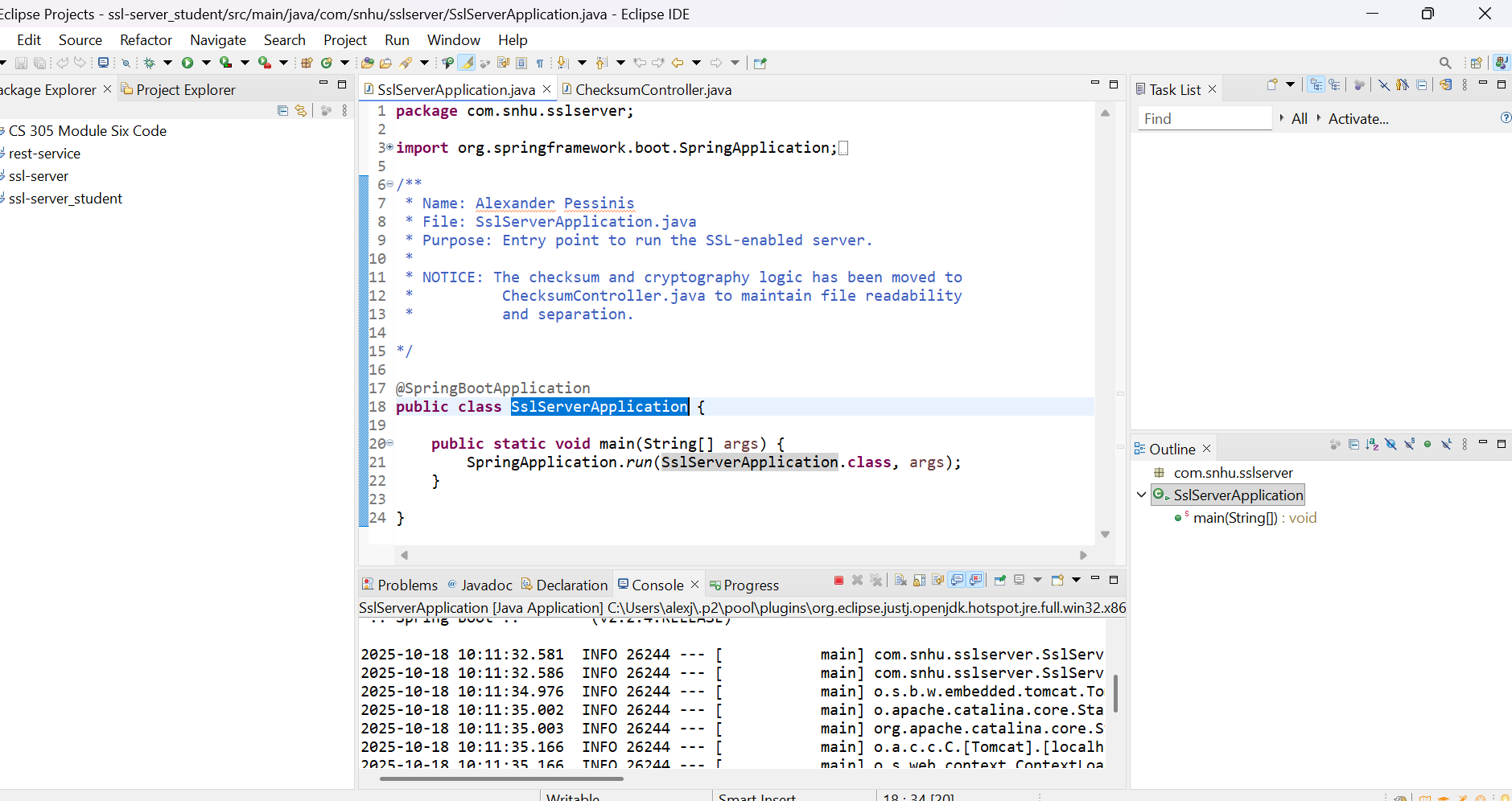
**Note:** It is https, but since the cert is self-signed, the browser still warns.



## Secondary Testing

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



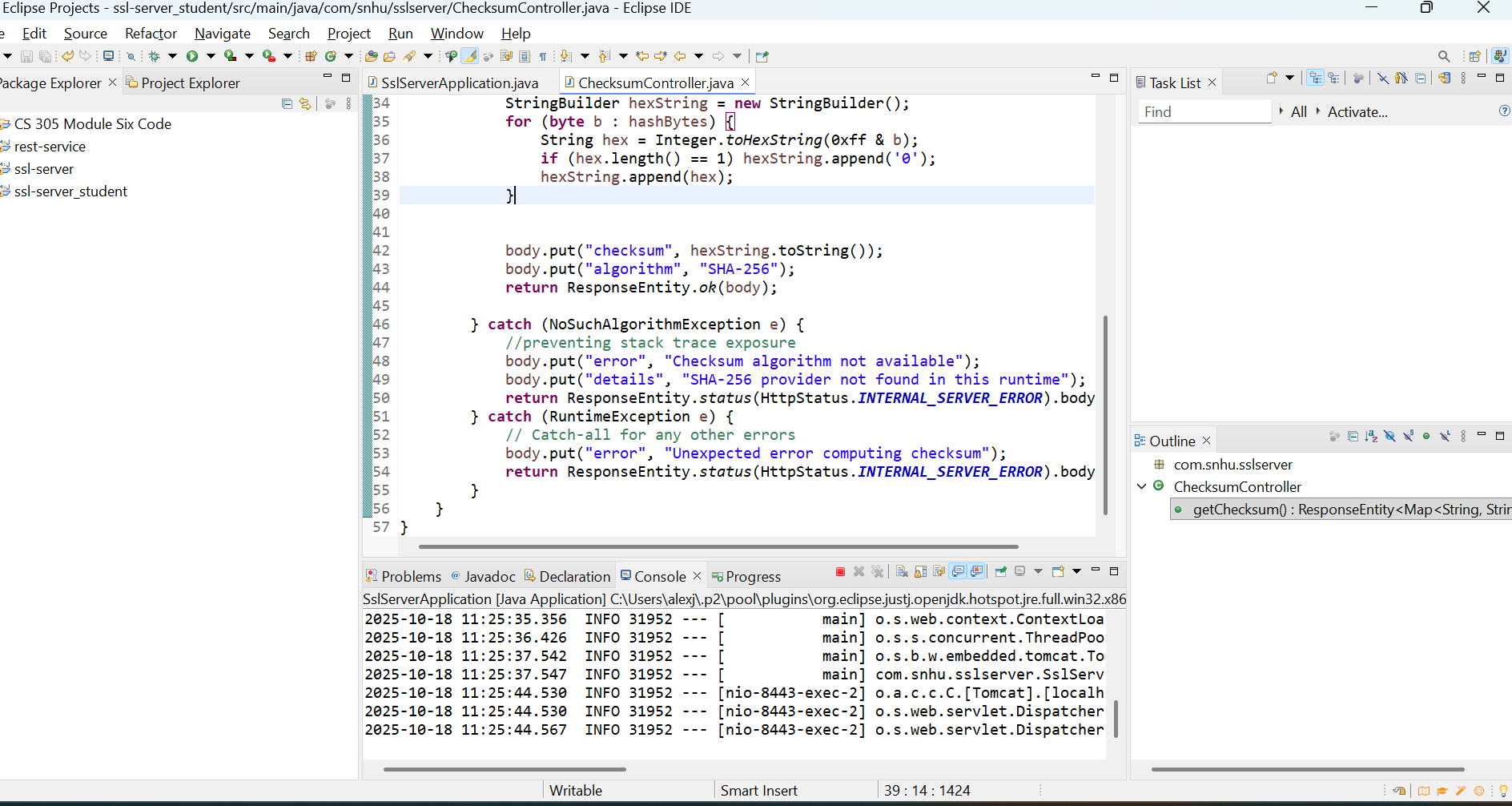




## Functional Testing

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

**Note:** There were no extra vulnerabilities found, however, I refactored my exception handling, just in case the unlikely event of exposing the stack trace, using references I found online for local exception handling.



## 

## Summary

The code was reviewed in accordance with the vulnerability assessment. **Cryptography** was strengthened by using SHA-256 for checksums, which complies with my own as well as NISTs recommendations. Potential **code errors** were mitigated with appropriate exception handling to guarantee that the stack trace does not get exposed. This also helps with code quality and encapsulation. An additional layer of security was added by utilizing HTTPS and generating a certificate, helping to guarantee encrypted communication between the client and the server. While no explicit or immediate vulnerabilities were identified, many potential vulnerabilities in the code were mitigated through these methods.

## Industry Standard Best Practices

To maintain the application’s security and to provide the highest level of security for this project, industry standard coding and security practices were employed, such as guaranteeing HTTPS connection, using SHA-256 for cryptography, and employing effective exception handling to prevent abuse of vulnerabilities. These measures align with the industry’s best practices for input validation, cryptography, and error management. This helps guarantee that the application is resistant to attempts at exploitation.

The value and advantages of using these practices extends far beyond mere compliance with the industry. By utilizing these standardized and proven strategies for mitigating vulnerability, the company vastly reduces the risk of data breaches, or any of the negatives that may come with a successful exploitation of software, such as legal trouble or reputational harm. The use of industry best practices helps the company and its clients resist potential cyber attacks, and supports customer trust.