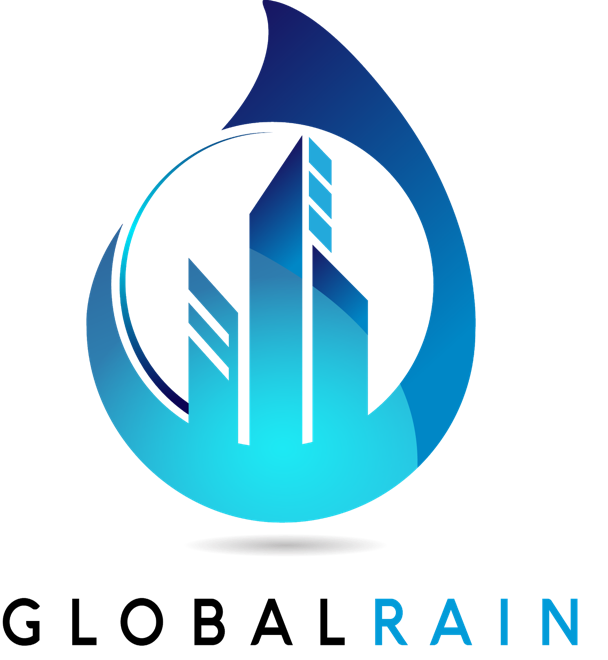
****

# CS 305 Project Two

**Practices for Secure Software Report**

**Table of Contents**

[Document Revision History 3](#_1fob9te)

[Client 3](#_3znysh7)

[Instructions 3](#_2et92p0)

[Developer 4](#_tyjcwt)

[1. Algorithm Cipher 4](#_3dy6vkm)

[2. Certificate Generation 4](#_1t3h5sf)

[3. Deploy Cipher 4](#_4d34og8)

[4. Secure Communications 4](#_2s8eyo1)

[5. Secondary Testing 4](#_17dp8vu)

[6. Functional Testing 5](#_3rdcrjn)

[7. Summary 5](#_26in1rg)

## Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **4/15/2022** | **Nahom Mekonen** | **Security Report** |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

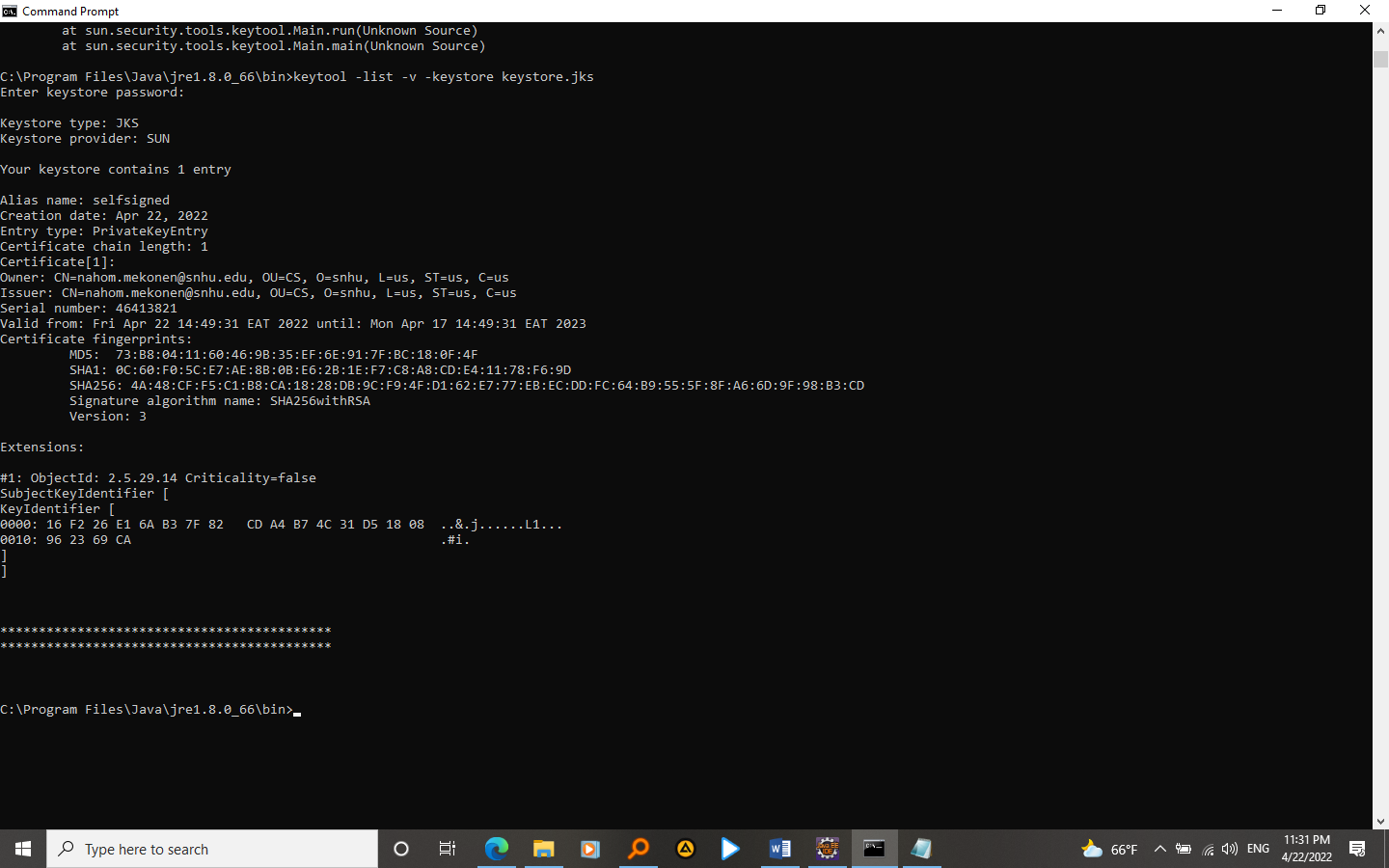
## Developer

Nahom Mekonen

## 1. Algorithm Cipher

The algorithm cipher I suggest for this project is AES-256. AES (Advanced Encryption Standard) is quickly becoming the go-to encryption algorithm for most applications as it is easy to implement on both a software and hardware level. AES has also been proven to withstand all attacks except brute force, and given a properly complex byte count (128, 192, or 256), the risk of falling to brute force drops significantly. “AES is a symmetric encryption algorithm. It was intended to be easy to implement in hardware and software, as well as in restricted environments and offer good defenses against various attack techniques.” (HowToDoInJava, 2020). AES is symmetric, meaning the same key is used to both encrypt and decrypt data. Since this algorithm only accepts byte levels of 129, 192, or 256 bits, random numbers are only useful in generating a sufficiently complex key for encryption/decryption. Encryption algorithms have played a large part in shaping the current state of the Computing industry. Most notably, their use during times of war (i.e. the Enigma machine in WWII) to send encrypted messages, and more substantially to decrypt enemy messages forever shaped the industry. Currently, encryption algorithms are commonly implemented to verify files, protect secure data on servers, and ensure trusted and valid connections to a host of web applications.

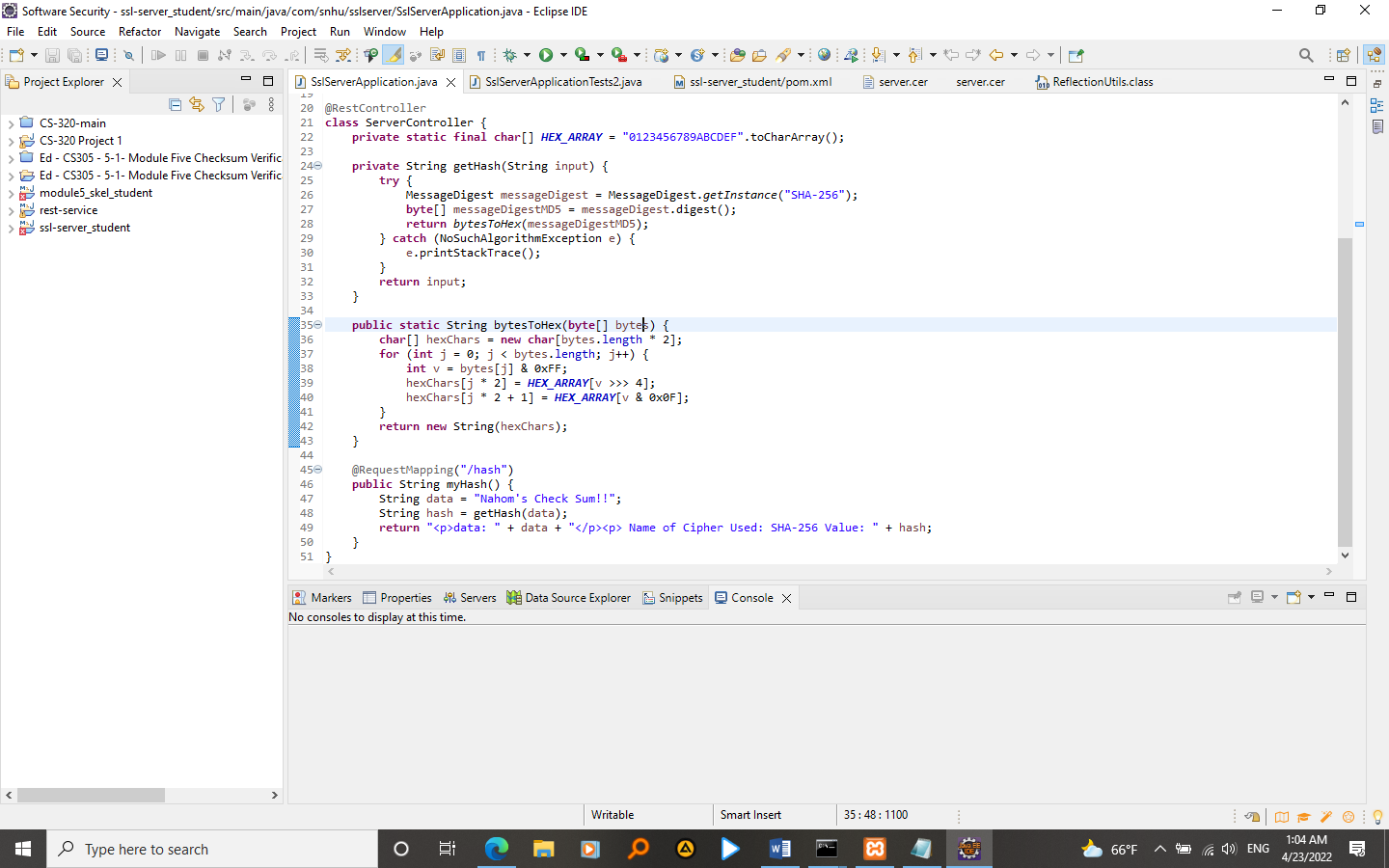
## 2. Certificate Generation



## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 4. Secure Communications

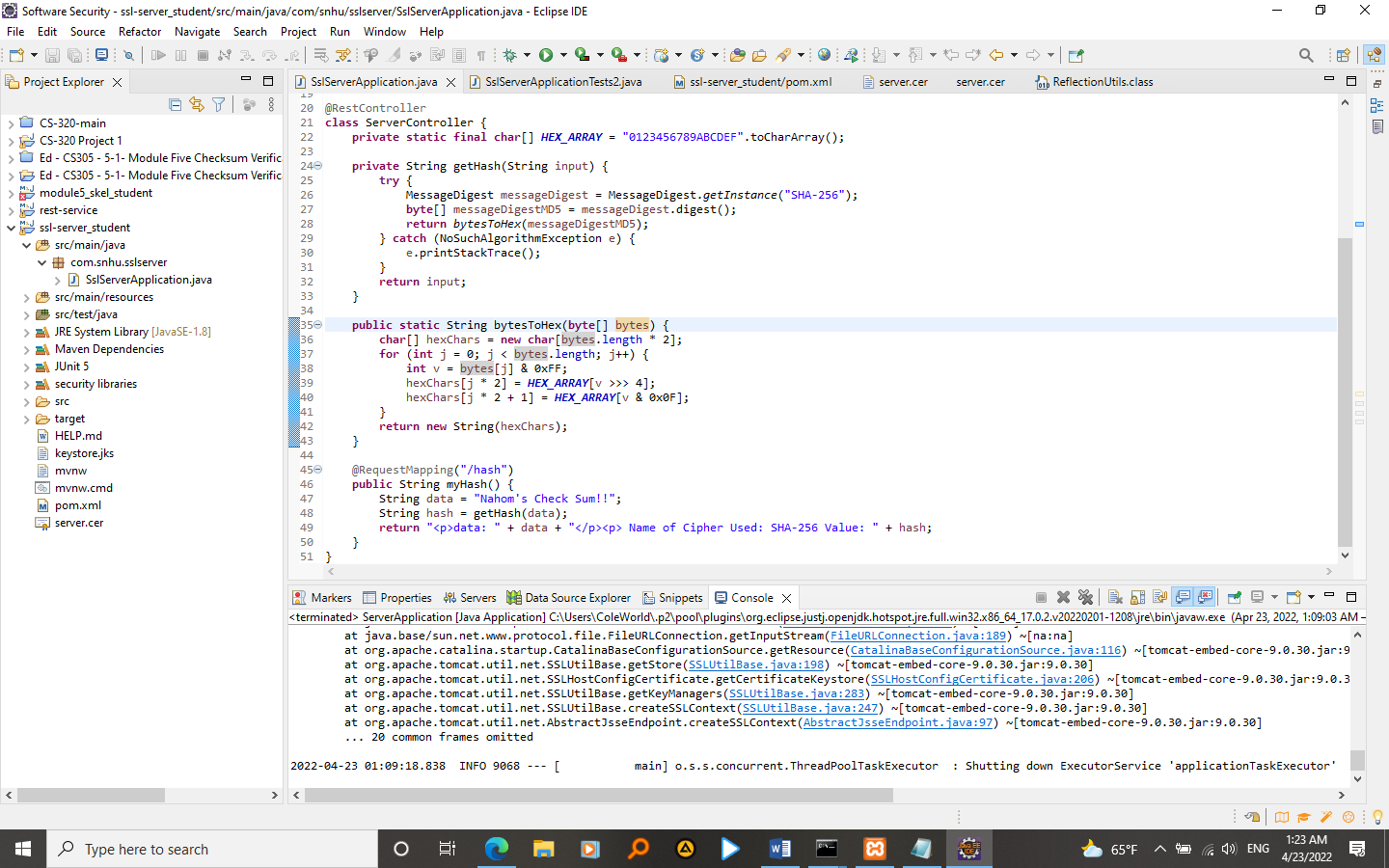
Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

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

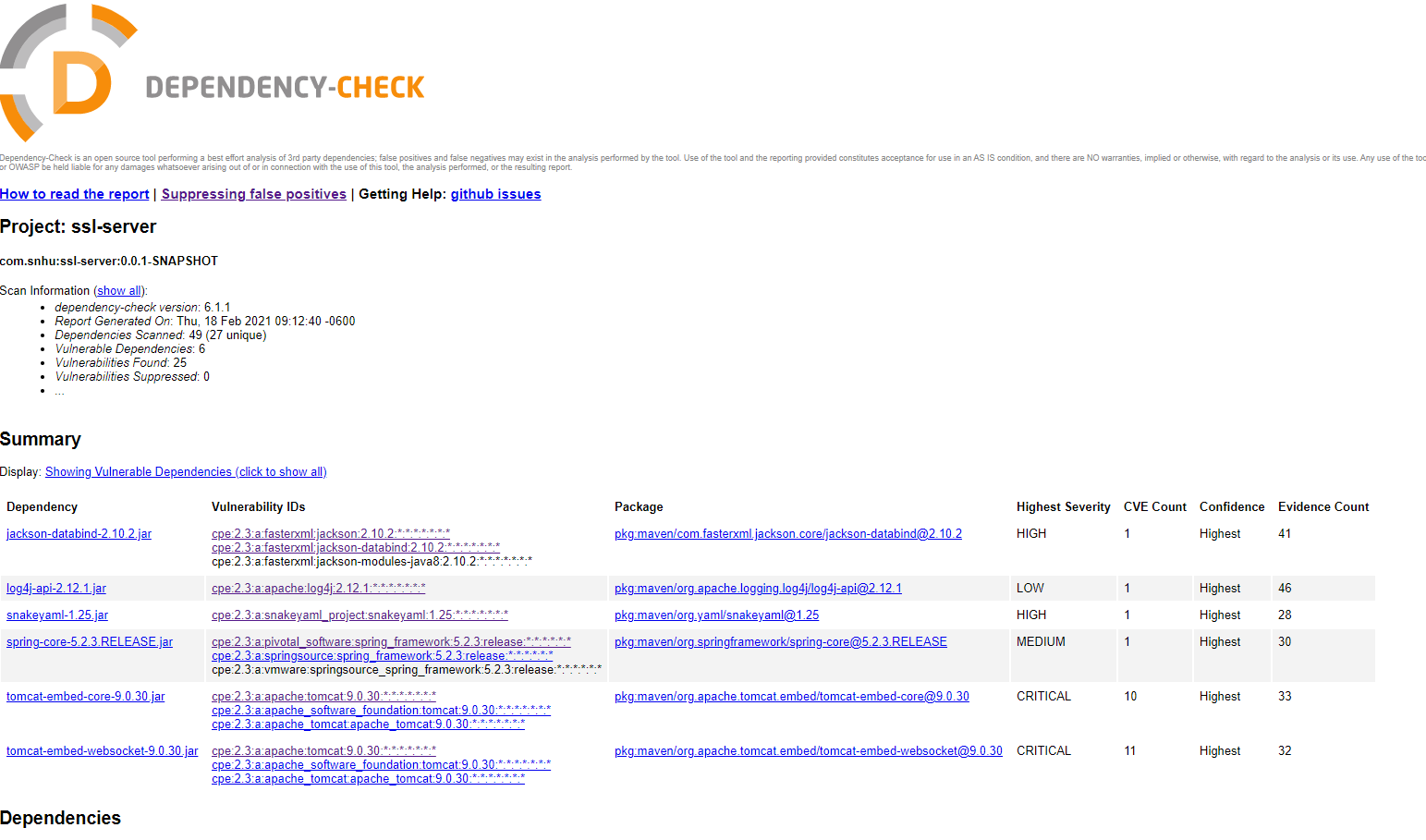
*Note: Web Browser used is Chrome, which doesn’t seem to “trust” the certification generated in previous steps.*

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.



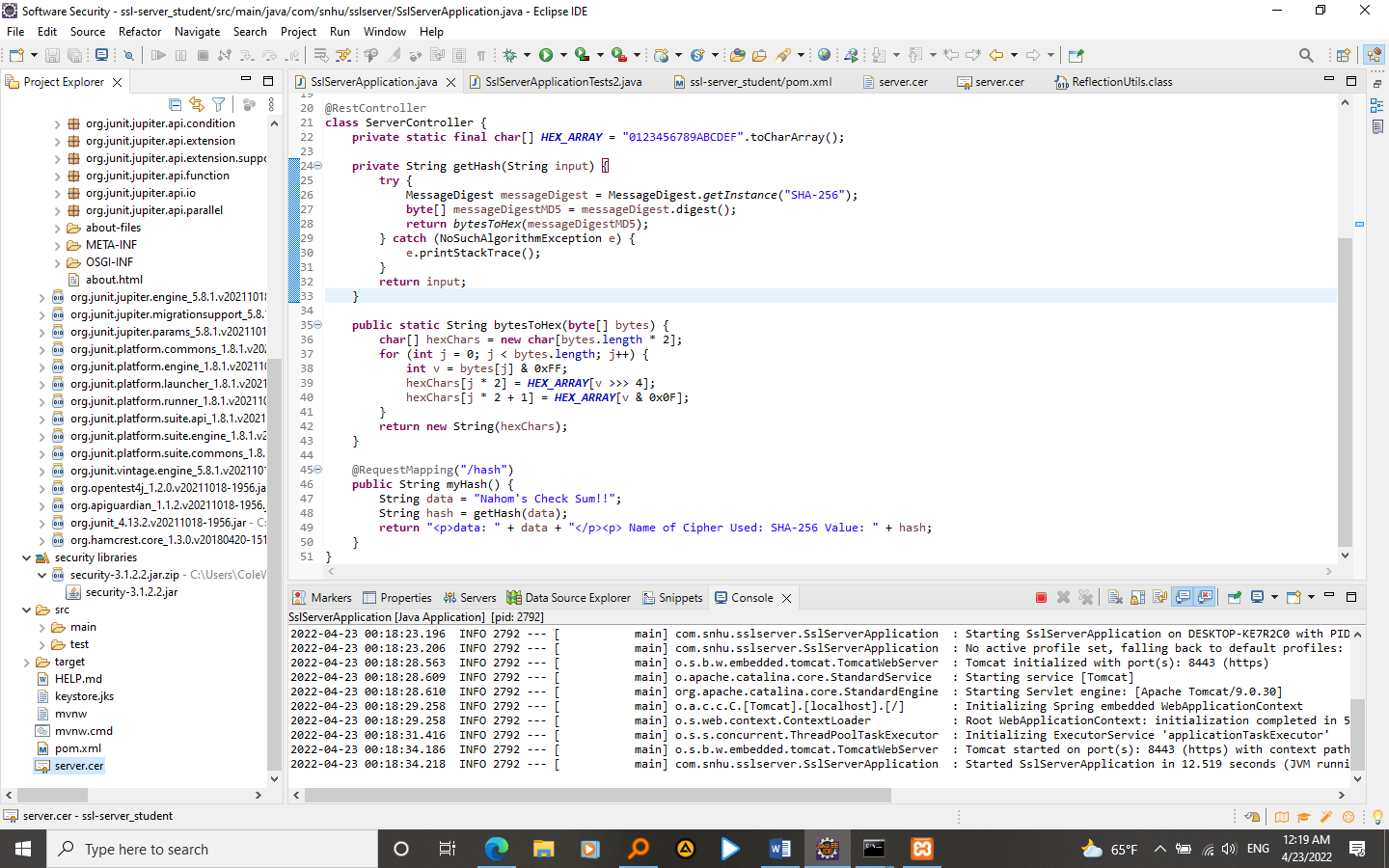
Refactored code running successfully (see Terminal at bottom of screen)



## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.



## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

The basic Spring web application has been modified with a hashing function. In its basic form, this function takes a string variable and encodes it using AES encryption. A secondary function also generates a checksum value that the server can display to the user, which in turn can verify the contents of the string passed through. This application can be extended to cover and encrypt any other data types (such as file verification, user login checks, etc.) in the future as needed. Additional security, especially in an age of web development, is extremely important; ensuring not only users have a safe and as-intended experience using the software, but also securing the integrity of the application and company itself. Maintaining such security should be a primary focus, including making changes to the encryption algorithm used if the current is compromised, and keeping system dependencies up-to-date to avoid dependency vulnerabilities.

**Bibliography**

HowToDoInJava. (2020, May 7). *Java AES 256 Encryption Decryption Example*. HowToDoInJava. https://howtodoinjava.com/java/java-security/aes-256-encryption-decryption/