****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/15/2021** | **Michael Richards** |  |

## 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

Michael Richards

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The algorithm cipher, the client needs is AES, while using SHA3-256 as the hash function. AES is widely used by banking institutions, the US government and used to secure messaging applications. AES was approved as the federal standard in 2002. It uses symmetric keys for the encryption and decryption of data.

The has function SHA3-256 has a bit digest of 256, which means that there are 2256 possible hash combinations. Which means that it helps to block brute force attacks and it is also unlikely to create collisions. One of the main differences with SHA3-256 is the performance speed. It is fast with the hardware but slower with the software. But according to Jeff Atwood, “But hashes aren't designed for speed. In fact, quite the opposite: hashes, **when used for security, need to be slow**. The faster you can calculate the hash, the more viable it is to use brute force to mount attacks.” That is why I choose to use SHA3-256, it may be slower by default but that is also a good thing when it comes to security.

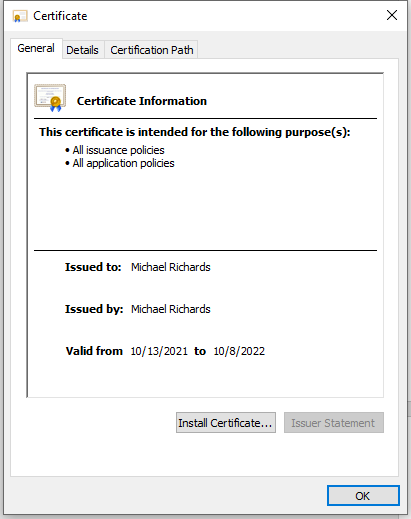
Non-symmetric keys use one key to encrypt and another to decrypt, when this key is used there is often a public and private key. Symmetric keys use the same key to encrypt and decrypt the data, so the key must remain protected.

AES became part of the US government in the 1990’s when the previous encryption algorithm DES(Data Encryption Standard) was cracked, which the US government still use today. I believe that AES would be a huge asset to Artemis Financial, to help secure personal data as well as any classified data that must be secured.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

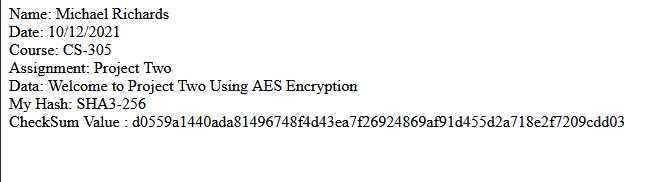
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

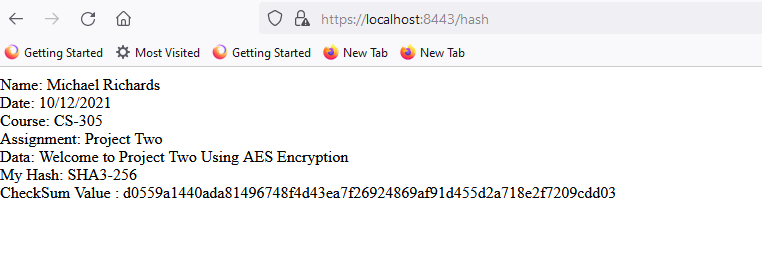
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

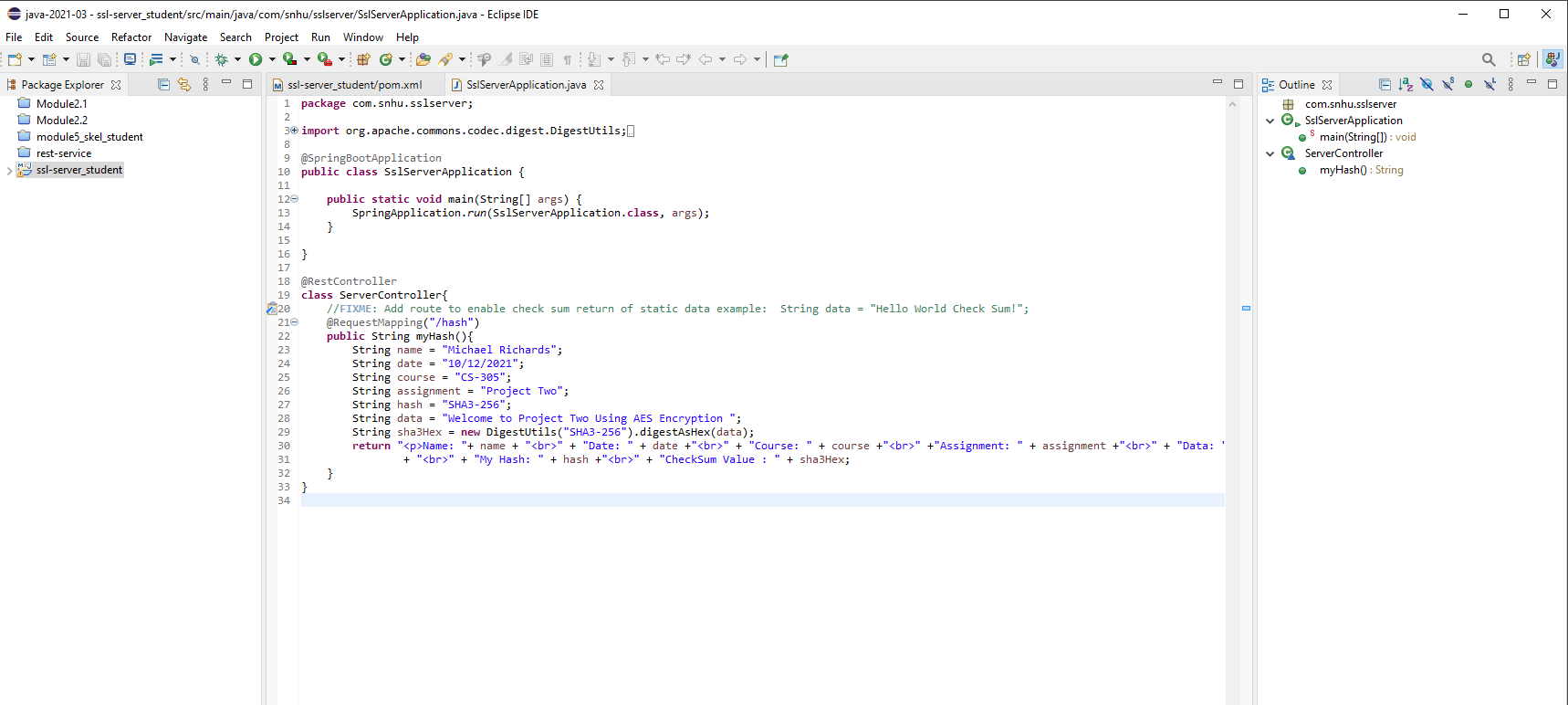
* Insert a screenshot below of the web browser that shows a secure webpage.

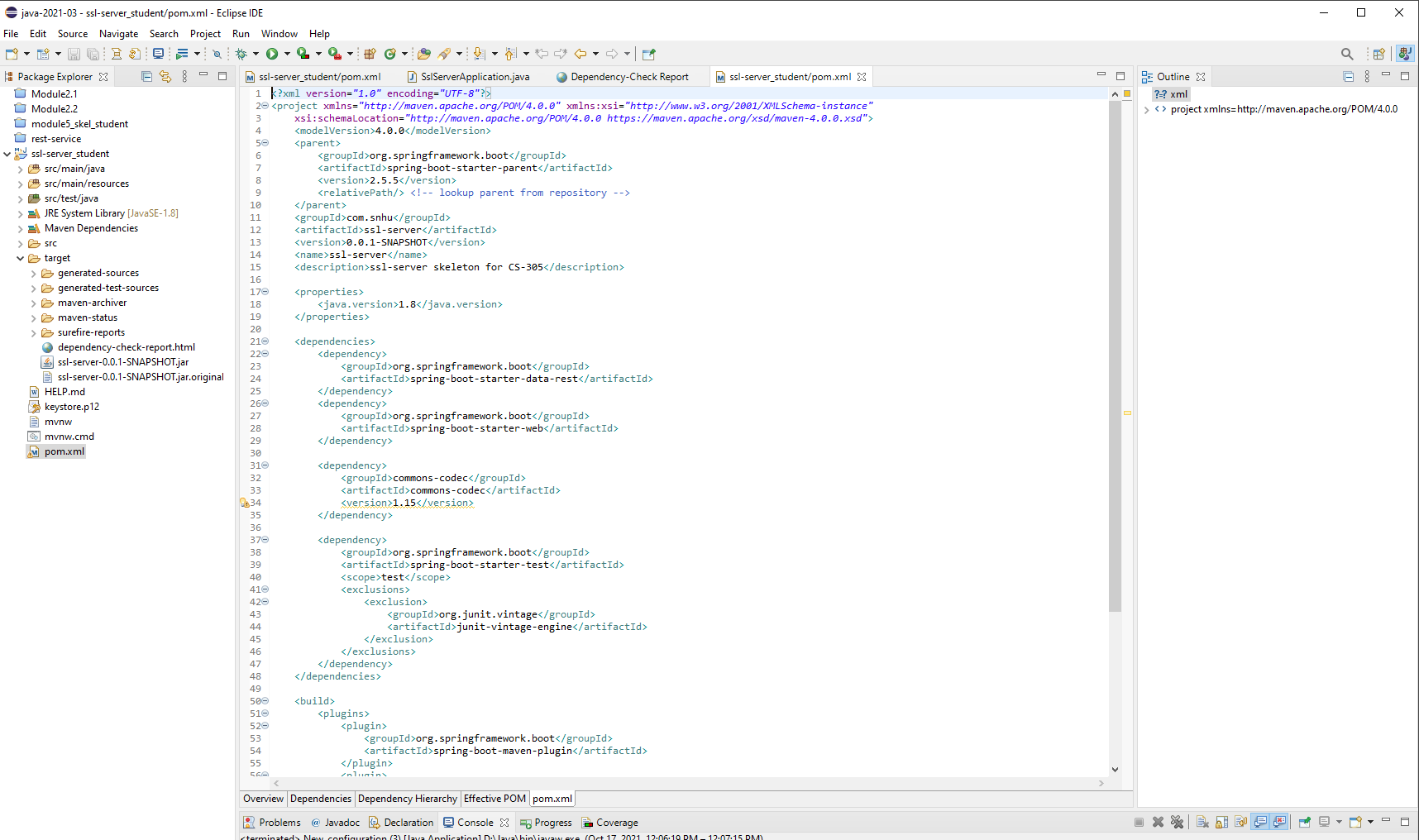


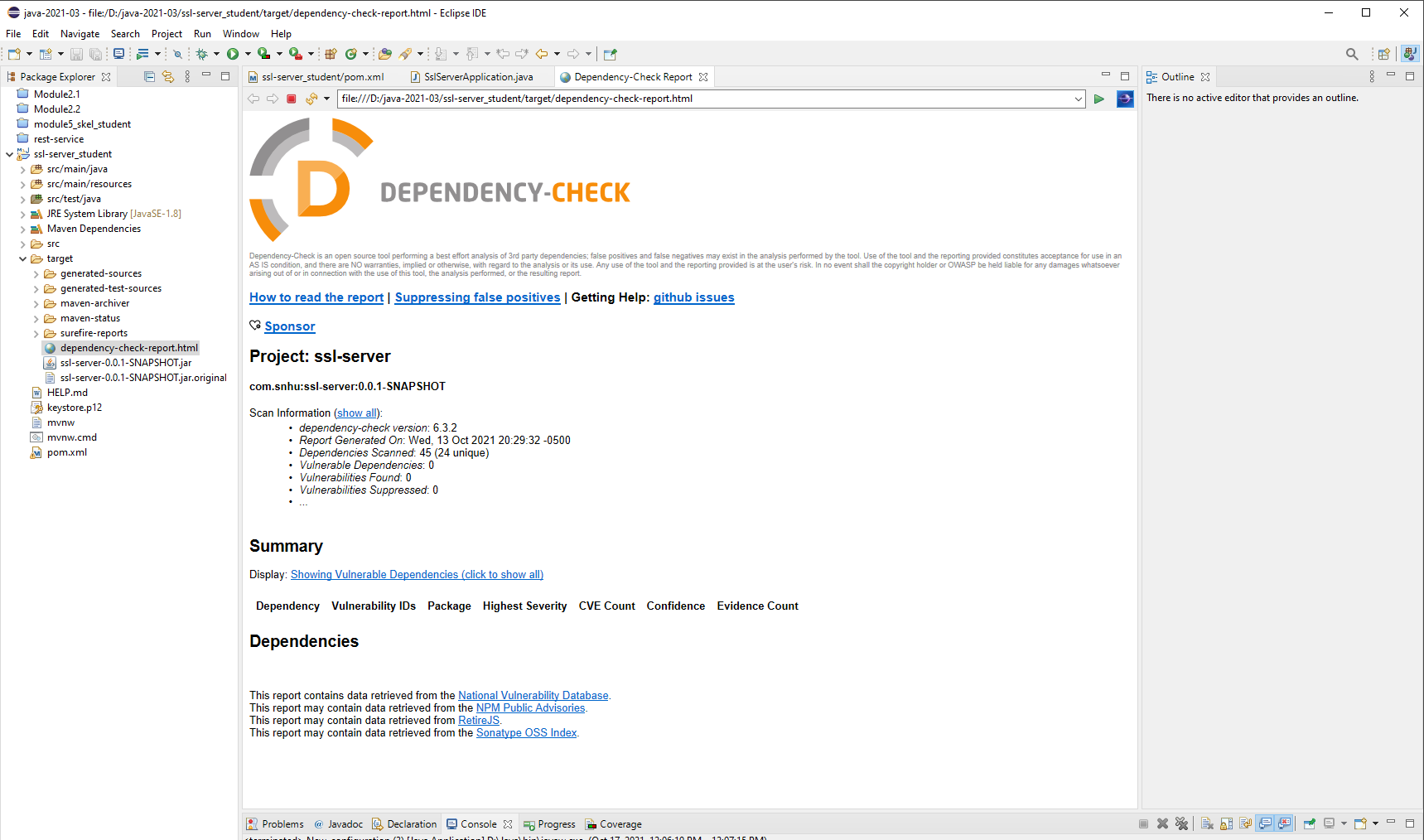
## 5. Secondary Testing

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.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report



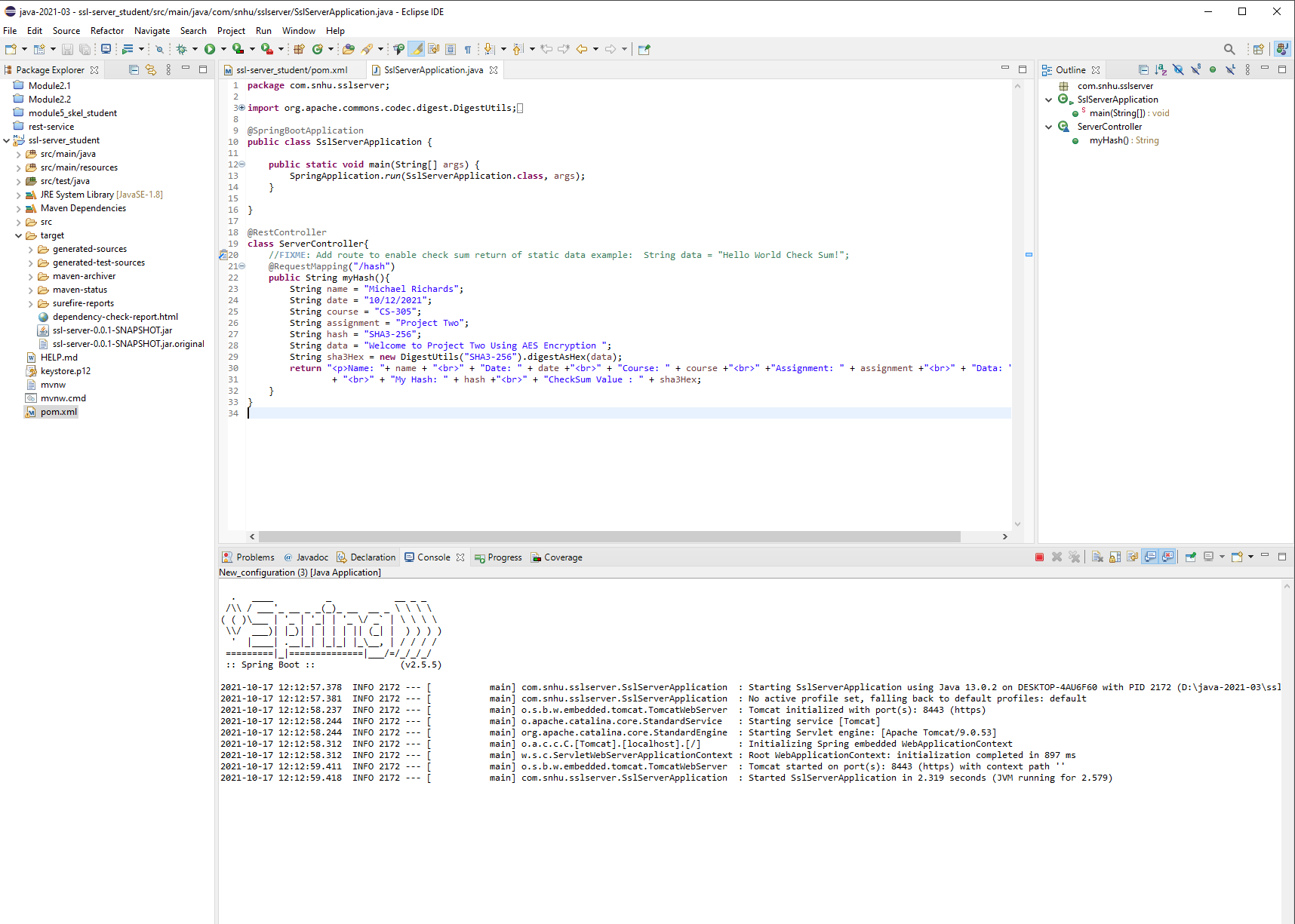




## 6. Functional Testing

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.



Runs with no errors.

## 7. Summary

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

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The first thing that needed to be refactored was the area of security, to include a hash function to encrypt data. Create a RESTful API to make it more secure. Also improved was the client/server security by adding a certificate. I also ensured that the latest versions of tomcat and spring-boot-parent were being used to ensure that the application is protected against known vulnerabilities. The best way to keep the application secure is for the dependencies to be checked for new vulnerabilities and updated to newer versions in the dependency-check tool if and when new vulnerabilities are found.