****

# 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** | **12/13/2020** | **Peta Clarke** | **Final Update** |

## 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

Peta Clarke

## 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.

After reviewing the various encryption techniques, I would most likely recommend **RSA** for Artemis Financials’ needs. As it would provide the level of security that would be needed for a task of this nature.

Justification:

**Hash functions** are mathematical functions that can turn pieces of data into a bit string of a fixed length. These bit strings are called hash values and help add complexity to data security.

For example, when working with passwords, the bit levels provide a way of increasing the amount of time that it would take a brute force hacker to crack a password by increasing the amount of password possibilities.

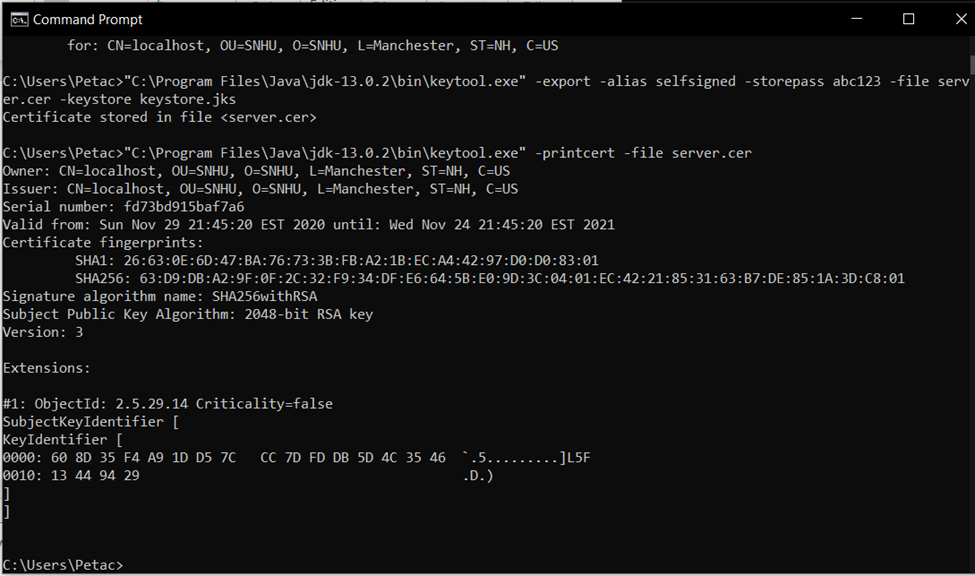
**Symmetric encryption** uses a single key that must be shared between the users that need to get the message. The key can be numbers, words, or random letters. **Asymmetrical encryption** uses a pair of public and private keys. It is a relatively new method and involves encrypting a message with a public key but then it can only be decrypted by using the private key. This method boosts the security over any data being transmitted. RSA uses this key encryption method.

Many encryption algorithms have been put out of practical use due to the ability to crack easily. Although a couple are difficult to crack, it really depends on how far technology moves forward and if there are any breakthroughs in mathematics. For example, RSA adds a great deal of security, but it is advised that by 2030, that may no longer be the case. This is true with all encryption algorithms, the faster that our computers become, the faster it is for hackers to crack a hash value.

## 2. Certificate Generation

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

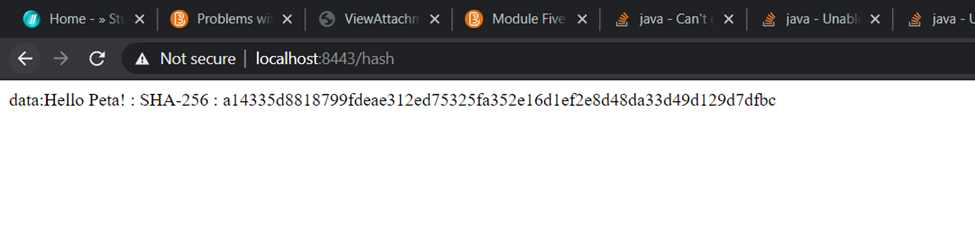
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

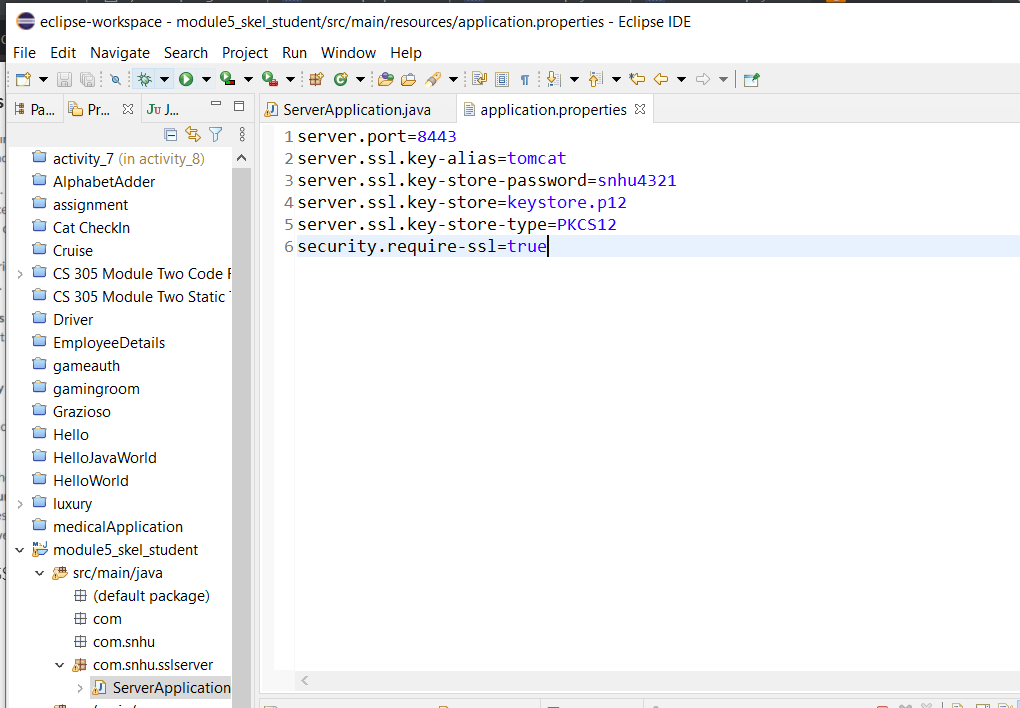
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

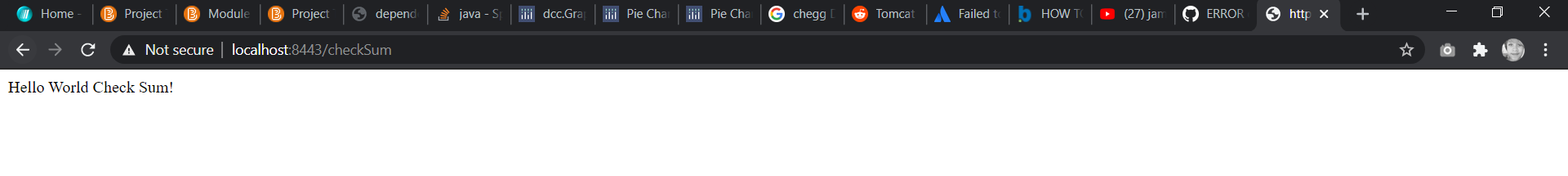


## 4. Secure Communications

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.

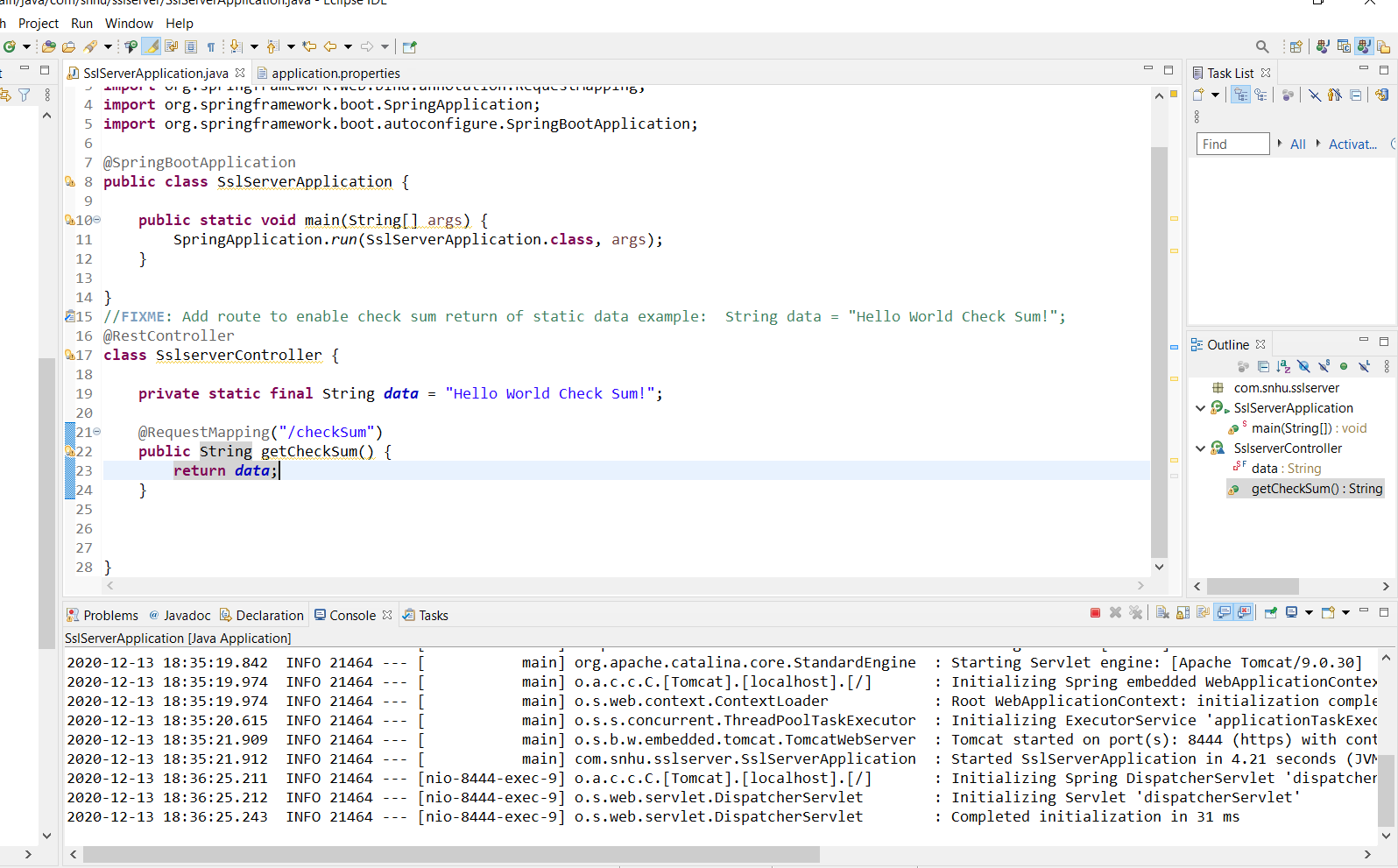
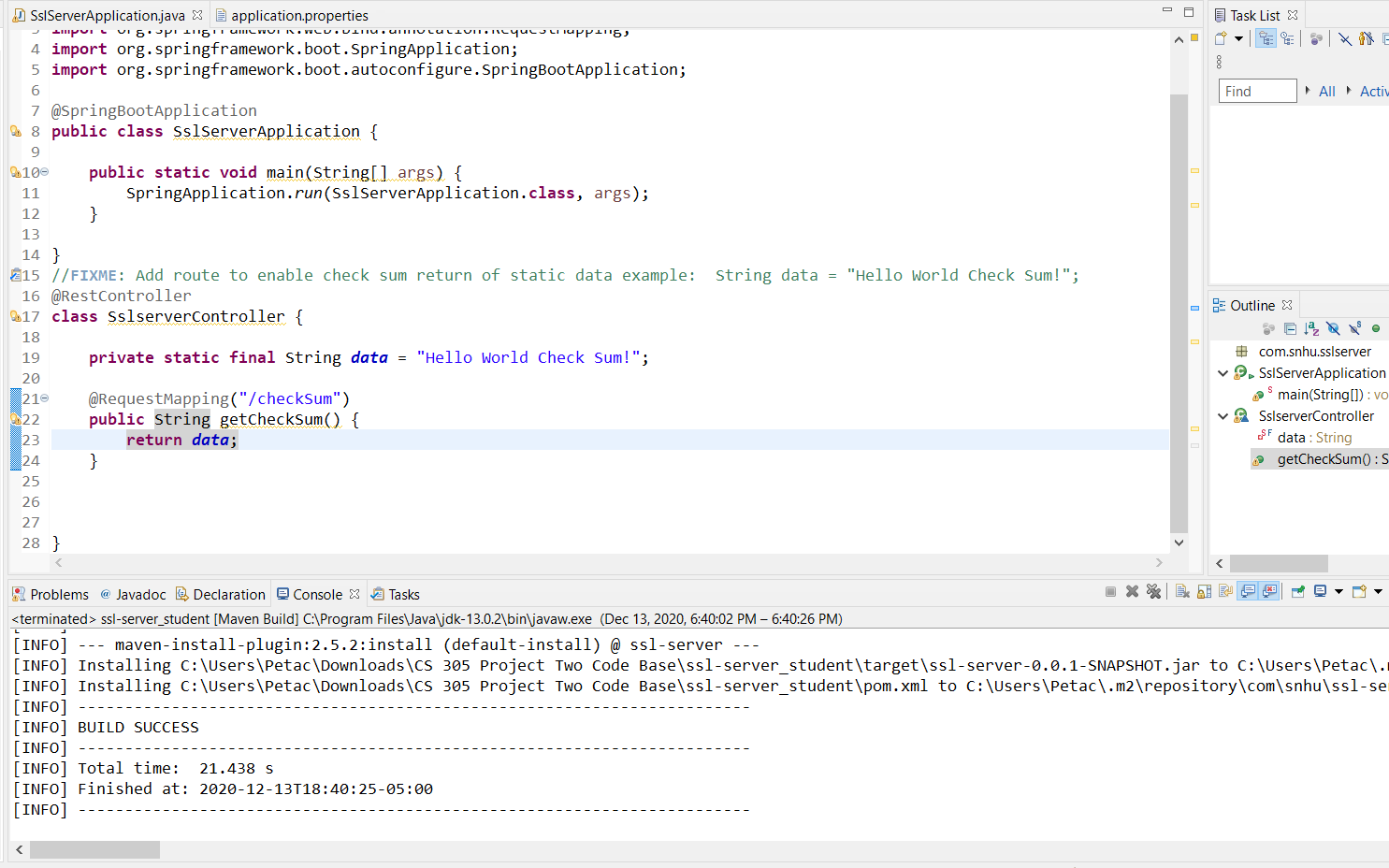
* Insert a screenshot below of the web browser that shows a secure webpage.

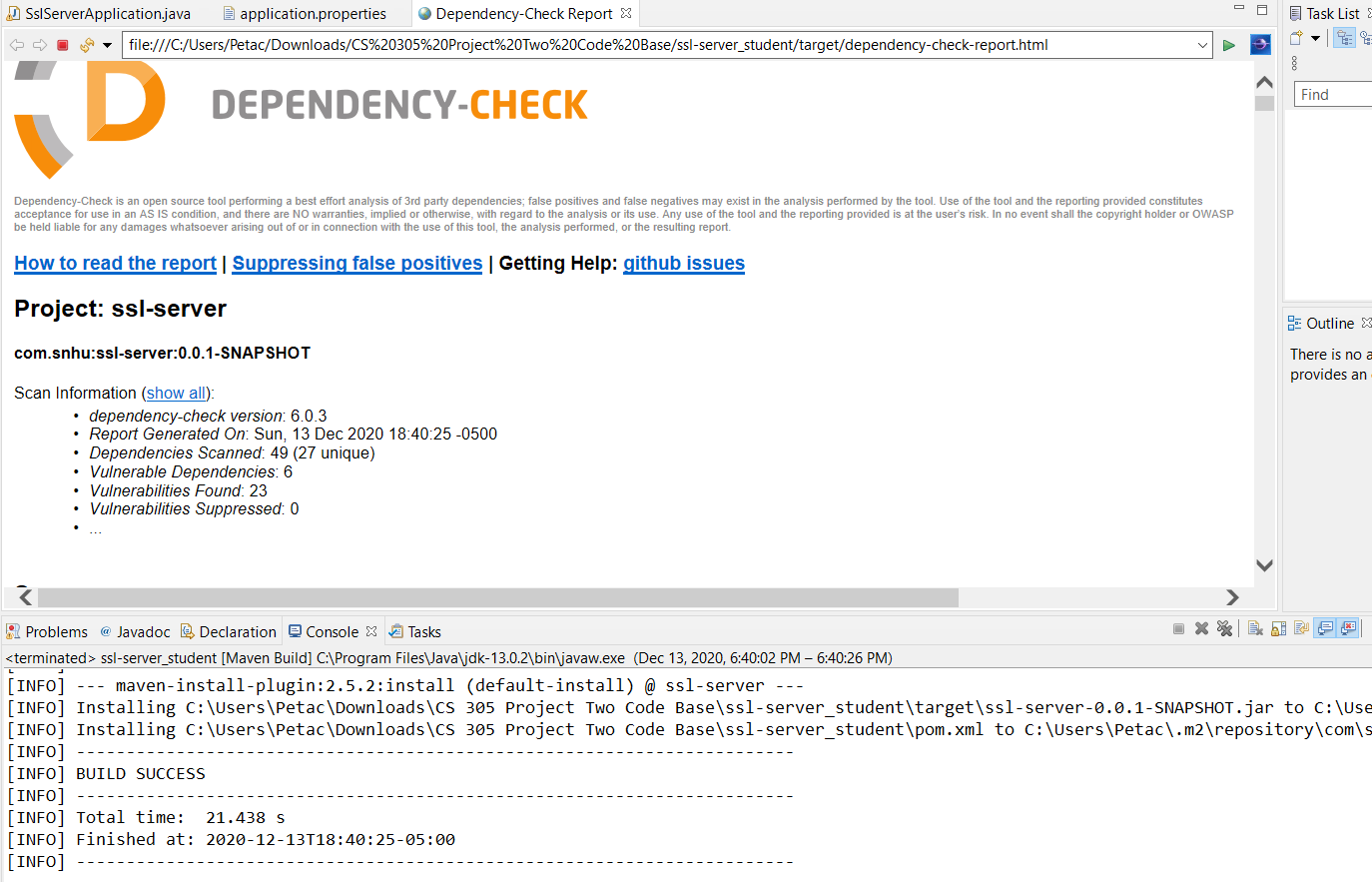




## 5. Secondary Testing

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.

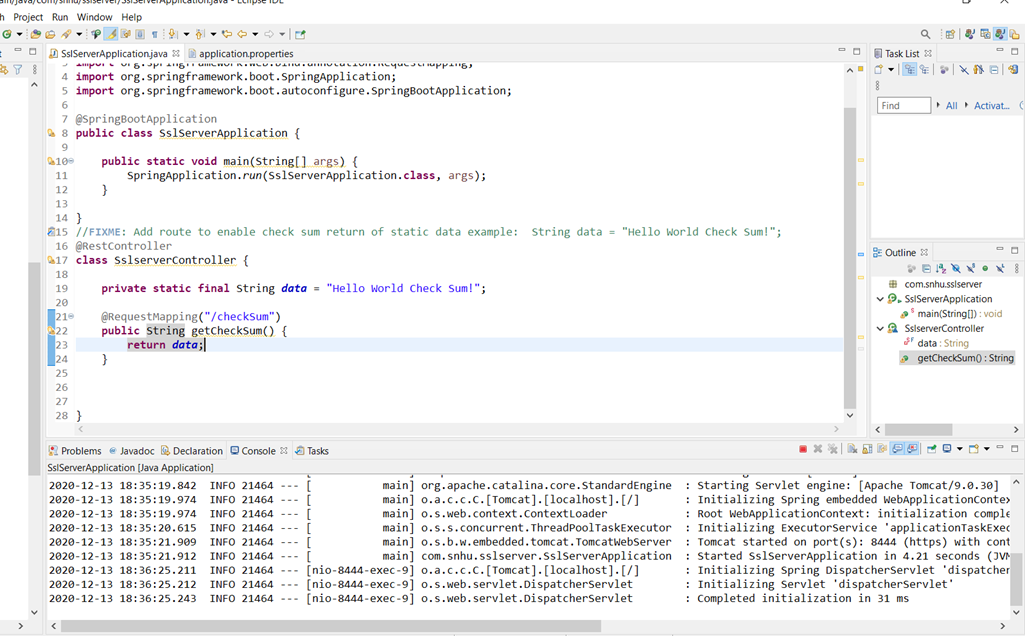
* Include the following below:
  + A screenshot of the refactored code executed without errors
  + 
  + 
  + A screenshot of the dependency check report



## 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.

[Insert screenshot(s) here.]

## 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 code has been refactored to enable a more secure application through https instead of http. By adding to the code in the application.properties file, I was able to create an extra layer of protection to the application. By adding this layer of protection, we are boosting the app’s security against hackers who might be after sensitive information. Although this adds more security, it does not mean that the application is secure from all hacks. It is always vital to keep up with any security updates as technology advances and computers become faster and more efficient.