****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/14/2023** | **Jose Perla** |  |

## 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

Jose Perla

## Algorithm Cipher

An algorithm cypher that would work best for this company would be Advance Encryption Standard (AES) which is a block cypher that can come in 128, 192, or 256 bit length. The higher the bit length the stronger the encryption is and the more difficult it is to crack. However, a caveat of this is that higher bit length can increase decryption time which would slow down the process of retrieving information or sending communication. The National Institute of Standards and Technology(NIST) includes AES-128 as an acceptable bit length for encryption so we can use this since it still provides maximum security while not slowing down the program.

To understand more why AES is a good choice for security we must understand more about its functions and components. One thing to consider is hash functions which is a means to verify the validity of data it works by taking an input and compressing it to an output of a fixed length. Hash functions are helpful to ensure data integrity and can indicate if the data has been altered which is especially useful against man in the middle type of attacks.

Another component of encryption is random numbers which add an extra layer of security that encrypts plaintext with a different cypher every time, this makes it more difficult to solve since there is no set pattern on how a message is being encrypted, for example maybe a simple encryption would be making A = 1 every time, intercepting a couple of messages would lead to this message being cracked. However, using randoms numbers every time would make it harder for hackers to crack. Normally messages are encrypted using symmetric or asymmetric keys. Symmetric keys are good for encryption because they scramble the message and can only be encrypted and decrypted using the same secret key. Asymmetric keys are better for signing because the sender holds a private key while the public key is known by anyone and validates the signature.

## Certificate Generation

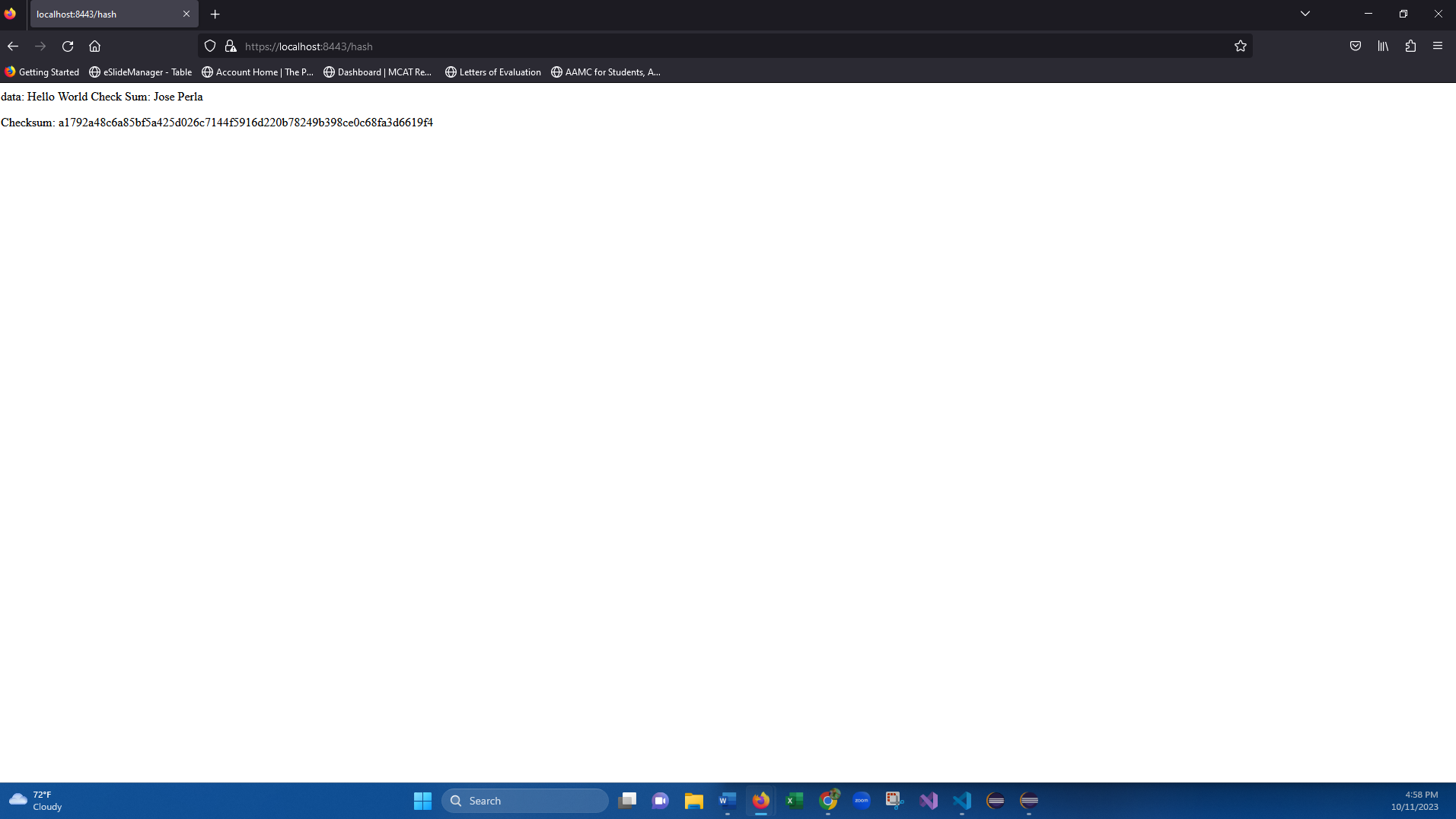
Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

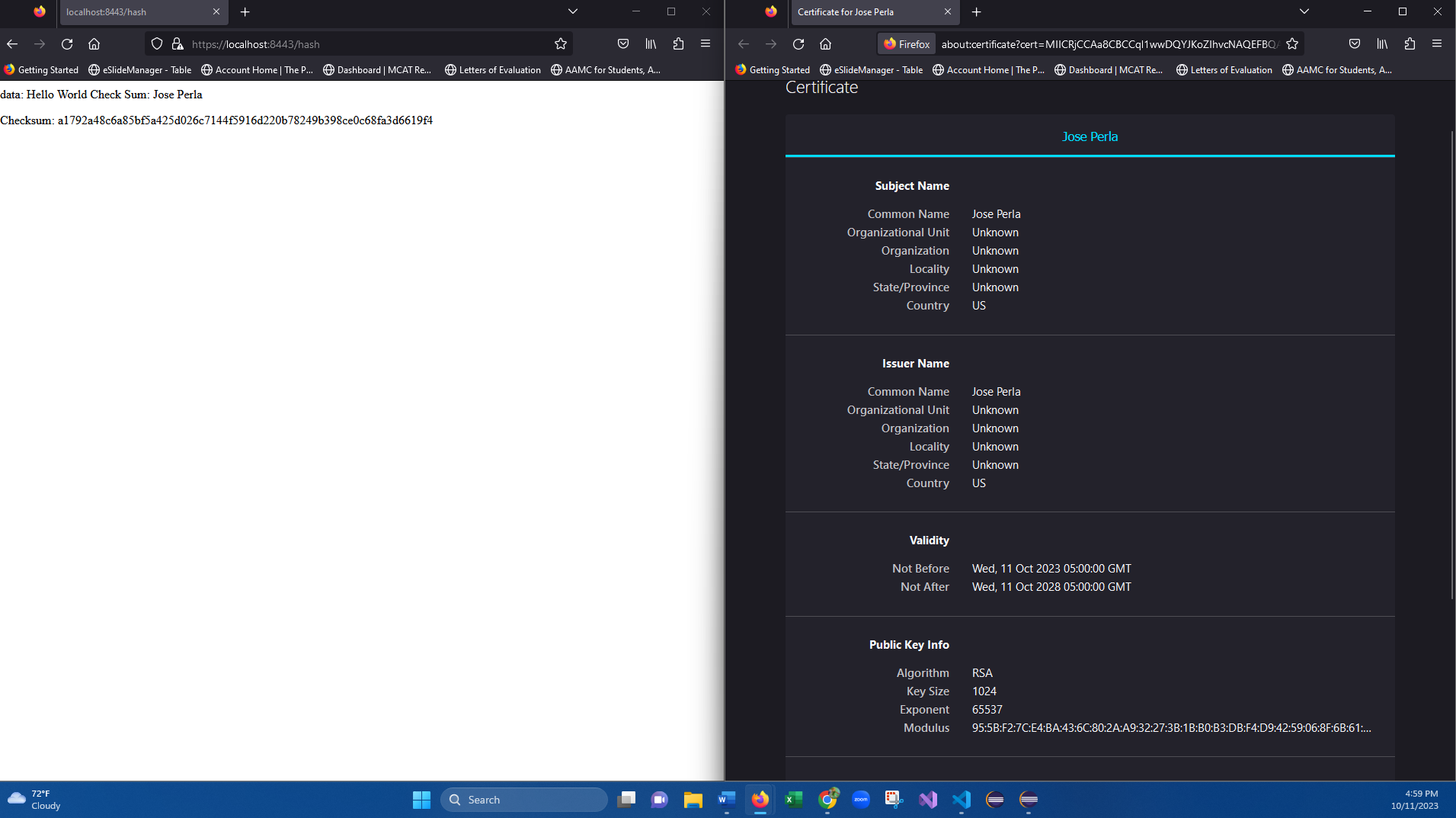
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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



## Secondary Testing

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

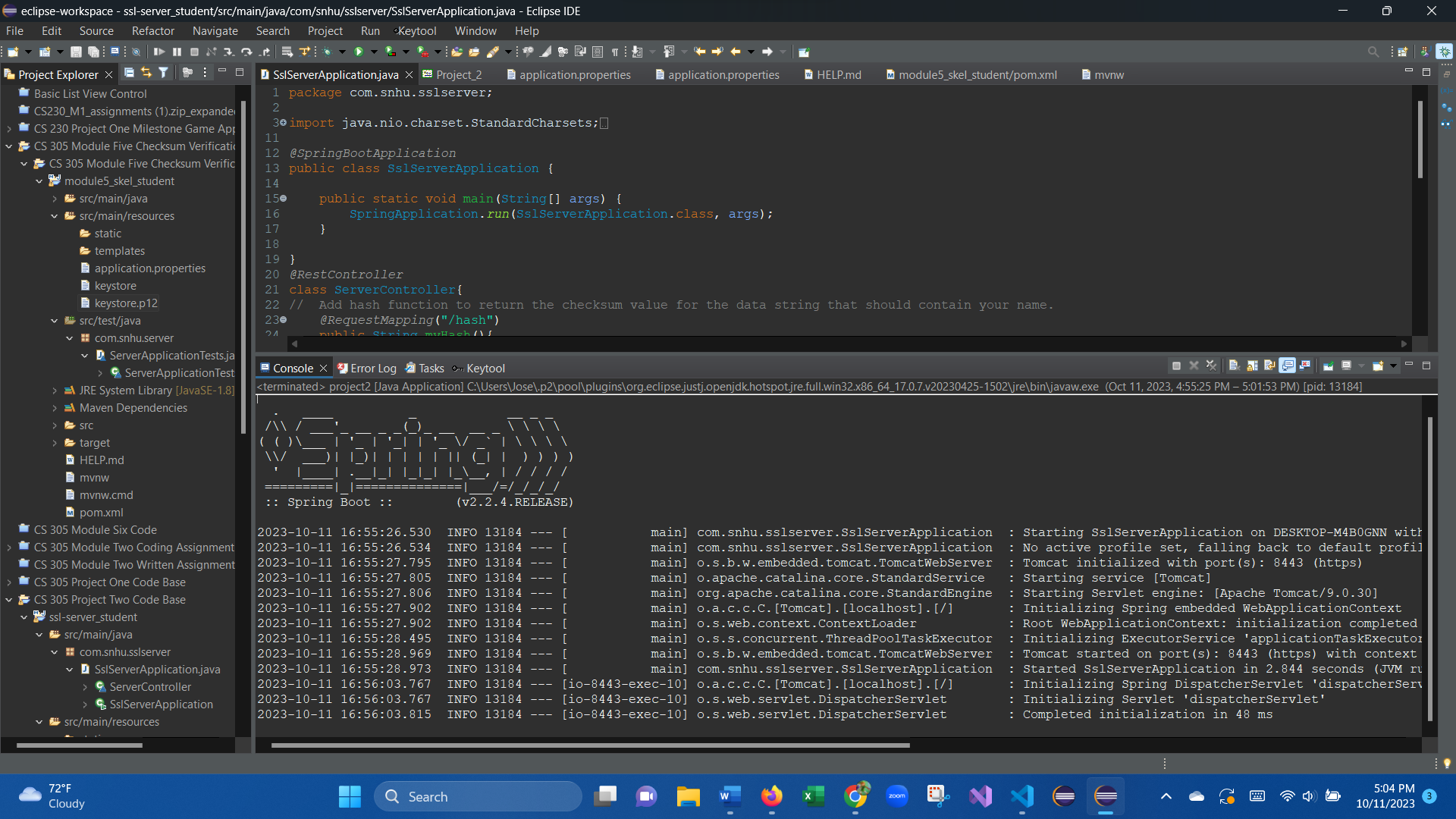
A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

## Functional Testing

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



## Summary

When it came to addressing our primary security concerns in the current iteration of the program our first step was to create a secure RestController by enabling usage of HTTPS with our self-signed certificate. We did this by creating a certificate with a secure key that will allow users to know that connection to the site was secure and validated. We also created a secure hashing function using SHA-256 creating a heavy encryption and we validated this encryption by utilizing a checksum that scrambles a message with our algorithm cypher. We also updated the Maven dependency check to the latest 8.4.0 version to ensure the most up to date check will be available so that we know what vulnerabilities exist in our current code so we can mitigate future vulnerabilities.

## Industry Standard Best Practices

When refactoring our code, we had to do a vulnerability assessment to understand what potential issues our program would have and steps we can take to combat these issues. Thus far we have created a secure API by creating our self-signed certificate that validate our site to the uses, we created a hash cypher that allows for encryption to prevent tampering of messaging as well as enabling secure communication protocols like HTTPS help with secure authentication. Those were the few examples we have used so far but there are other things to consider as well. Techniques such as input validation would ensure we are protected against injection-based attacks, implementing error handling would also prevent data leaks and maintaining logs of those who have access to the system so we can detect security breaches. Utilizing industry best practices not only protects company sensitive information but also helps protect client’s information. This in turn increases our responsibility and need to create secure code that prevents unauthorized access to our system and provides customers piece of mind knowing that the sites they are visiting are secure.