****

# 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** | **10/24/21** | **Michael Tekin** |  |

## Client



## Developer

Michael Tekin

## 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.

As Artemis Financial is seeking to add a file verification step to their web application, what is needed is to accomplish that goal is a cipher that can simply be used to verify that data is authentic and has not been corrupted intentionally or otherwise.

Ciphers can be separated into two main categories - ciphers that create ciphertext which can be decrypted revealing the initial data before it was encrypted and ciphers that create a sequence of fixed-length characters called a hash. We can refer to these categories as encryption algorithms and hash algorithms respectively.

Encryption algorithms tend to be more complicated which means they tend to be slower and more resource intensive than hash algorithms. For this reason, given that file verification can be done using a hash algorithm, that is the category of ciphers that should be considered.

Characteristics that must be considered when choosing a hash algorithm include performance and security. Modern hash algorithms include MD5, SHA-2, and SHA-3. When focusing on security, SHA-2 and SHA-3 are generally accepted to be more secure than MD5. They have longer hash sizes and therefore have a higher collision resistance. For general purpose processors, SHA-2 is faster than SHA-3 and given that SHA-2 does not yet have any known structural weaknesses that compromise its security, SHA-2 would be the recommended hash algorithm for Artemis Financials purposes.

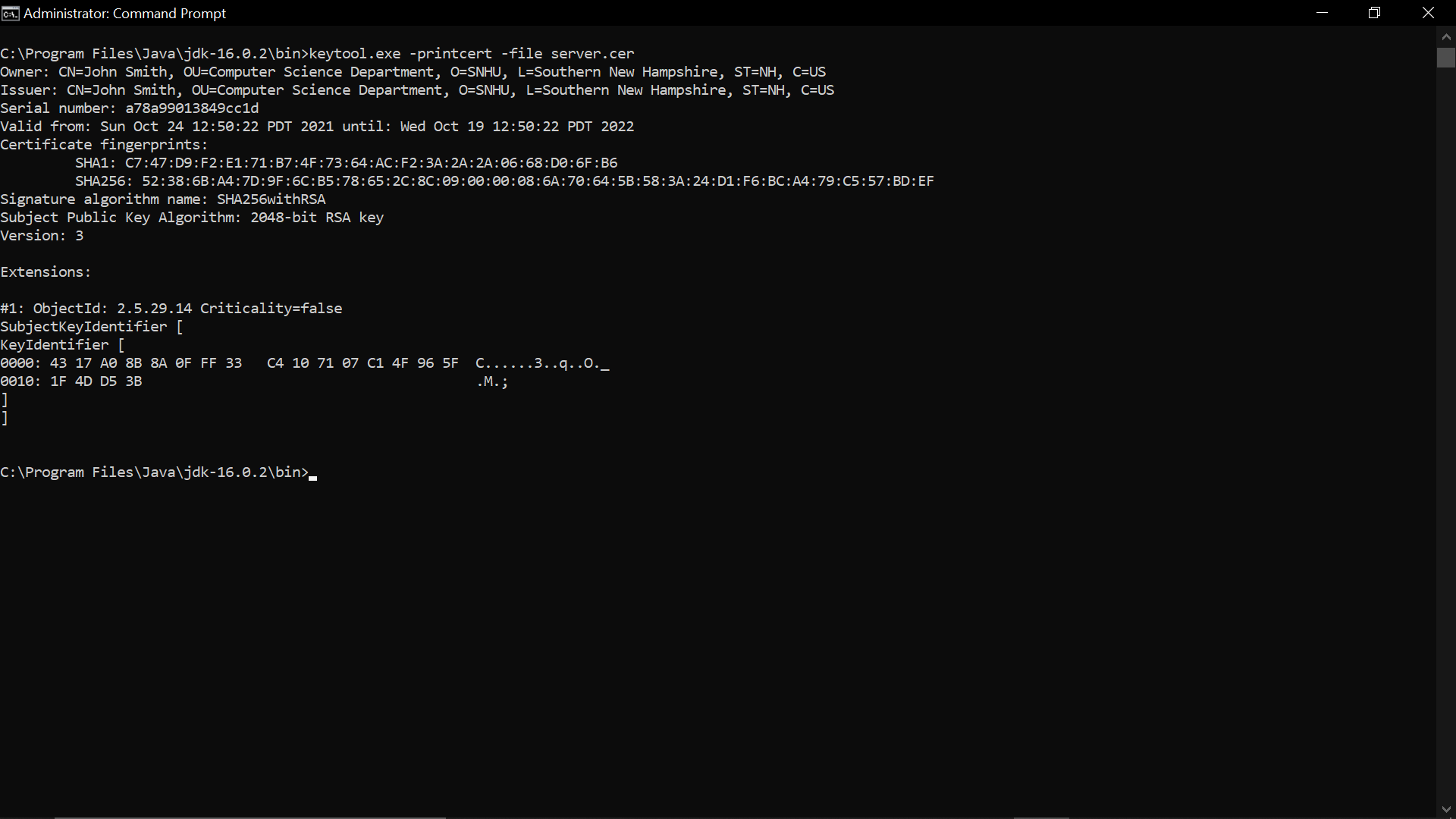
SHA-2 is a family of hash functions that are available in a range of bit levels. The most common variant being SHA-256 (256 refers to the bit length of the resulting hash or digest produced by the SHA-256 hash function) and this is the variant that Artemis Financial should use to verify data in secure communications.

As SHA is a hash function, there are no keys for “decrypting” the digest. If Artemis Financial also needed to encrypt the data they were sending back and forth to customers, they should consider using AES to encrypt the data as it’s one of the most secure and widely used encryption algorithms.

## 2. Certificate Generation

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

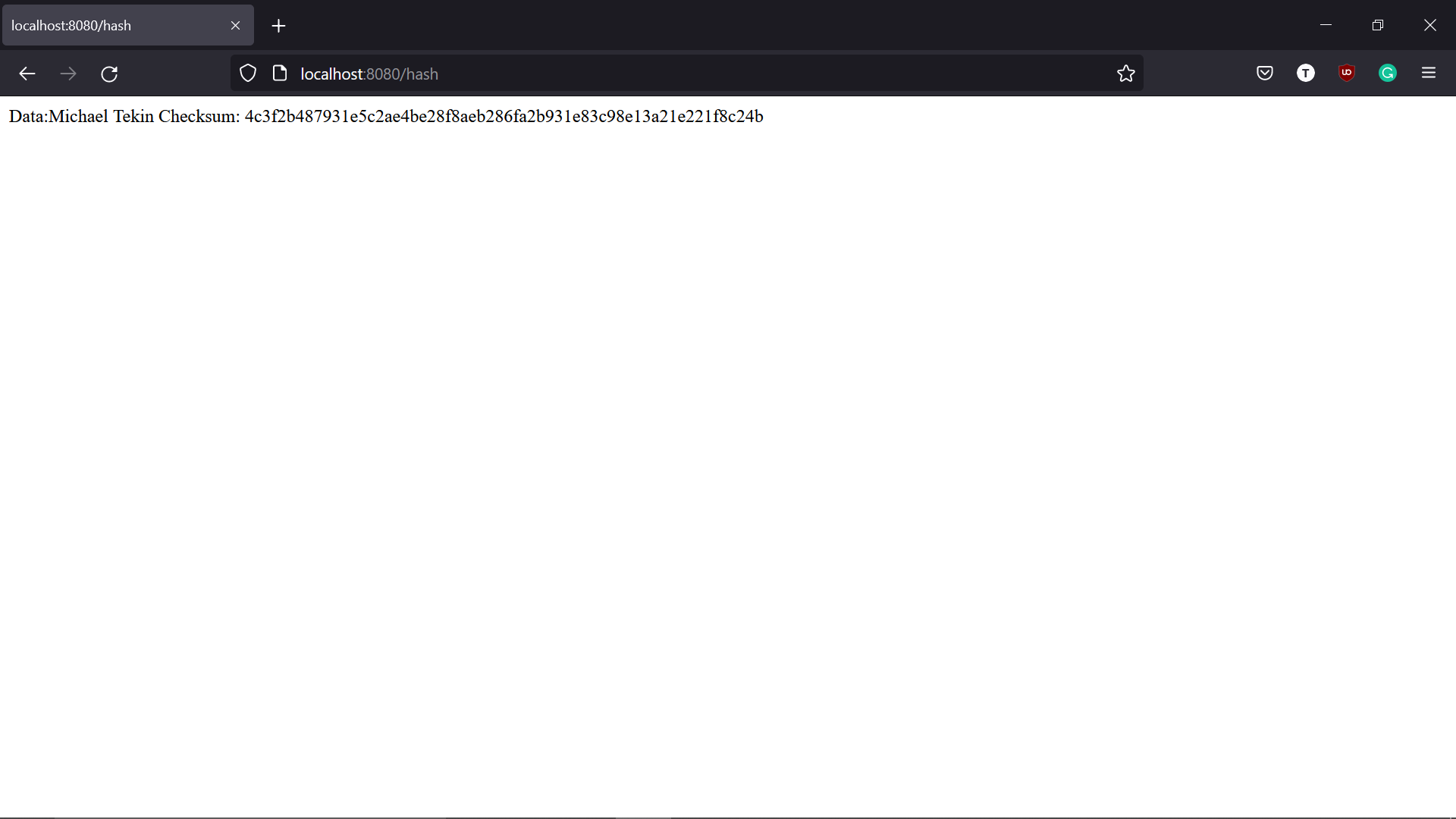
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

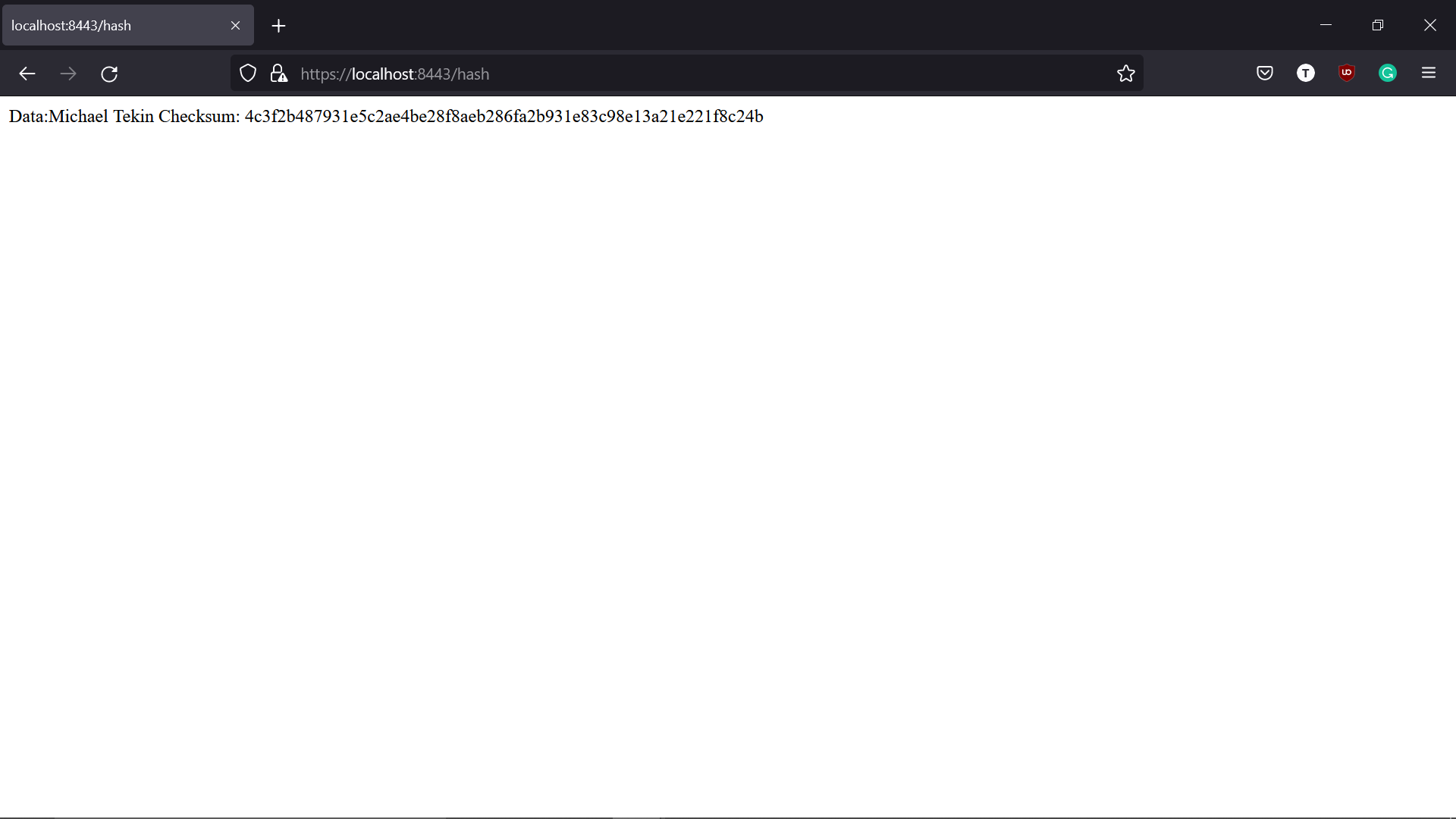
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

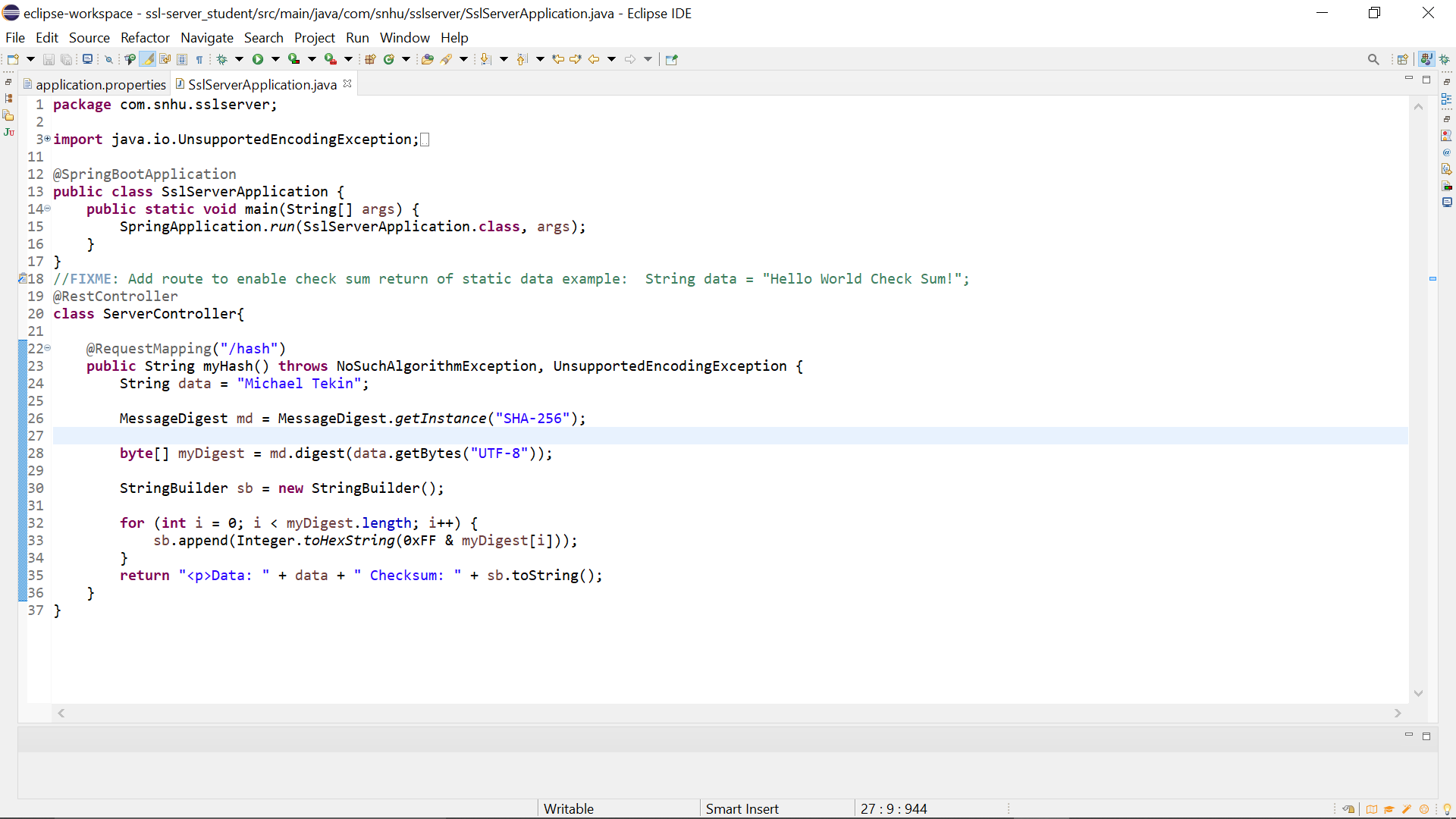
* Insert a screenshot below of the web browser that shows a secure webpage.



## 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

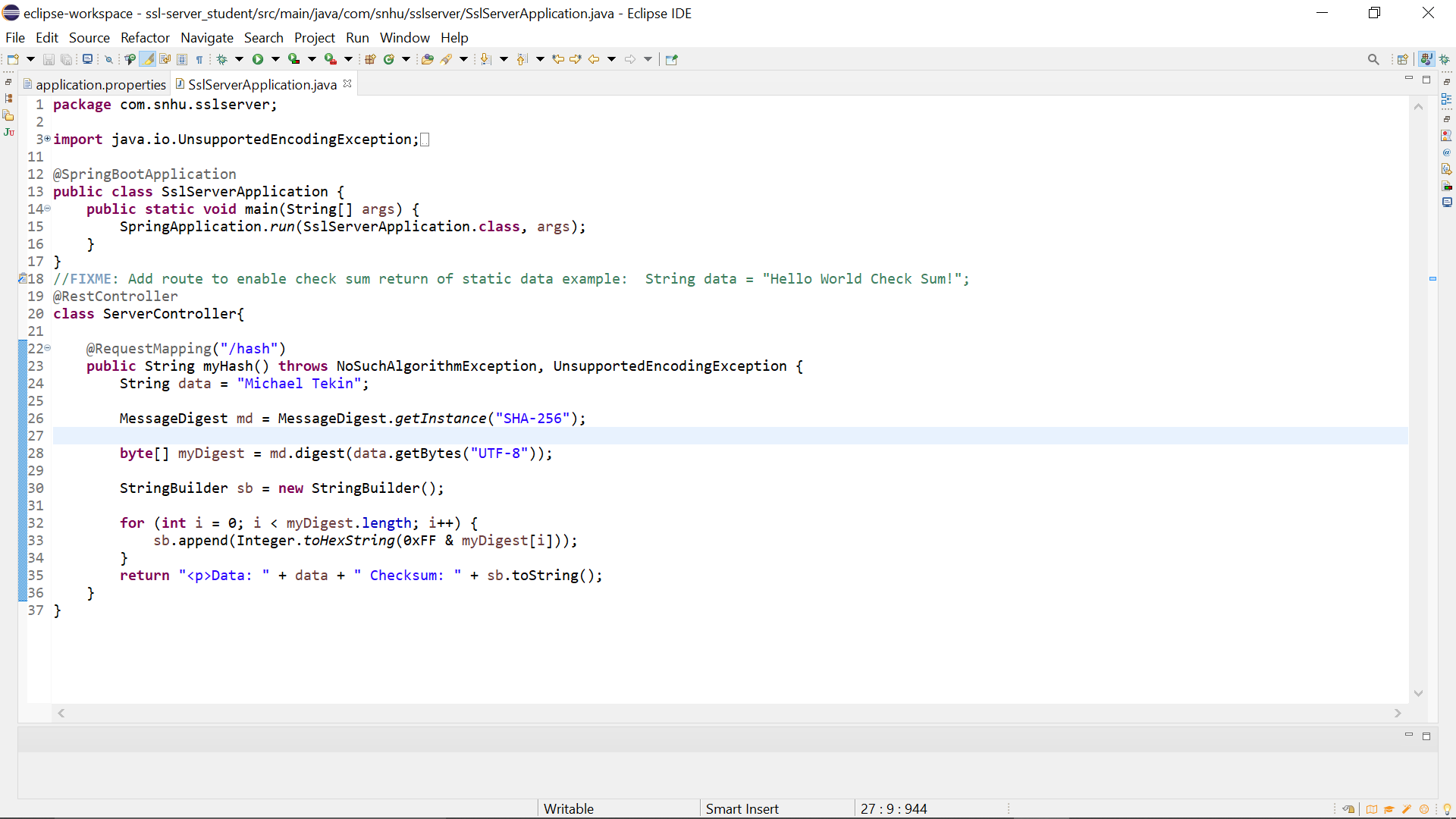




## 6. Functional Testing

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.



## 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.

By refactoring the code, we have addressed cryptography and APIs. We added layers of security with our file verification and with our signed certificate. Now that users are able to access the website over https and are able to view a checksum for data they are able to be more confident that communications are secure. The checksum can help users confirm that data they have received has not been corrupted or altered in any way. The signed certificate allows users to confirm that they are connected to an authenticated website. Best practices for maintaining current security will involve ensuring all dependencies are kept up to date, that certificates are renewed before they expire, and being on the lookout for any news about changes in encryption cipher reliability.

The refactored code is included in the attachment “CS 305 Project Two Code Base.zip”

**References**

*What's the difference between SHA and AES encryption?* Stack Overflow. (1957, September 1). Retrieved October 25, 2021, from https://stackoverflow.com/questions/990705/whats-the-difference-between-sha-and-aes-encryption.

Nohe, P. (2021, March 24). *The difference between encryption, hashing and salting*. Hashed Out by The SSL Store™. Retrieved October 25, 2021, from https://www.thesslstore.com/blog/difference-encryption-hashing-salting/.

*Why isn't sha-3 in wider use?* Cryptography Stack Exchange. (1967, November 1). Retrieved October 25, 2021, from https://crypto.stackexchange.com/questions/72507/why-isn-t-sha-3-in-wider-use.

Java security standard algorithm names. (n.d.). Retrieved October 25, 2021, from https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names.

*What is the difference between SHA-3 and SHA-256?* Cryptography Stack Exchange. (1967, June 1). Retrieved October 25, 2021, from https://crypto.stackexchange.com/questions/68307/what-is-the-difference-between-sha-3-and-sha-256.

Cypher Research Laboratories. (n.d.). Retrieved October 25, 2021, from http://www.cypher.com.au/crypto\_history.htm.