****

# 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** | **4/15/2023** | **Ryne Williams** |  |

## Client



## Instructions

Submit these 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

Ryne Williams

## Algorithm Cipher

After reviewing the requirements and security needs of Artemis Financial, I recommend that this software use the SHA-256 cipher algorithm. SHA-256 is a member of the SHA-2, or “Secure Hash Algorithm”, family that is a cipher algorithm created by the National Security Agency to replace SHA-1. It uses a hash function to convert data into a 256-bit string. This type of hash function can be used on data of any type and length and will always produce a hashed string of 256 bits. This makes the data unreadable and can only be verified by a key given to the user to ensure that the data has not been changed. In encryption, random numbers are used to generate keys for certificates and in encrypting data in a way that makes it impossible to get the original data. These keys are used in both symmetric and asymmetric encryption. Symmetric encryption uses the same key to both encrypt and decrypt the data being secured. Alternatively, asymmetric encryption uses a pair of keys, a public and private key. The public key is used to encrypt the data, while the private key is given to the receiver of the data for decryption. When encryption was first created during World War 2, the idea was much simpler than it is now. It involved using rotating disks that encrypted the message and was changed daily. It was implemented in computer technology in the 1980’s to secure the data of users for IBM. However, now encryption has a much larger scope and is regulated by the government to ensure that the algorithms in use are safe to use in all applications, including to encrypt classified data for the government.

## Certificate Generation

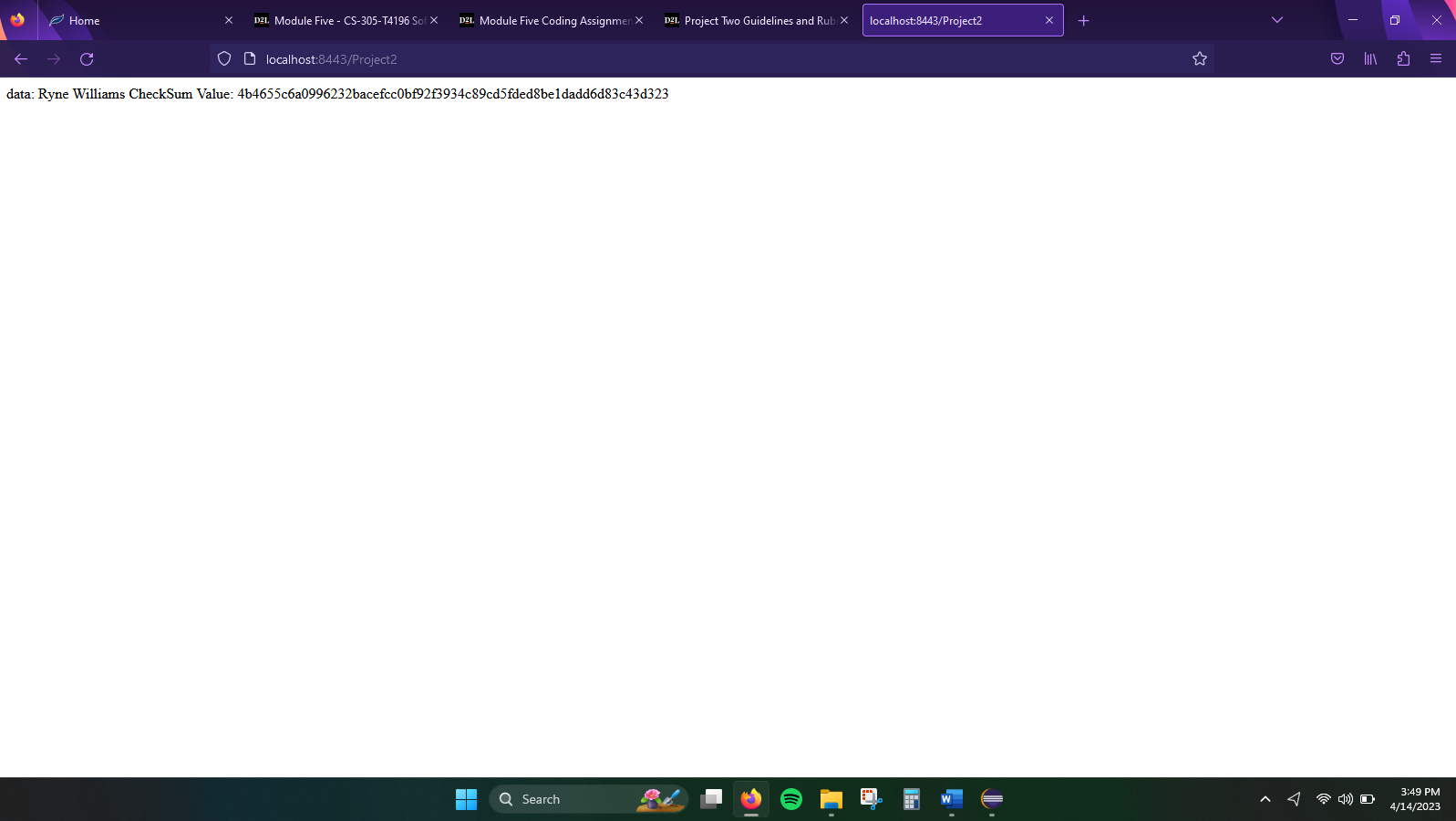
Insert a screenshot below of the CER file.

Text

Description automatically generated

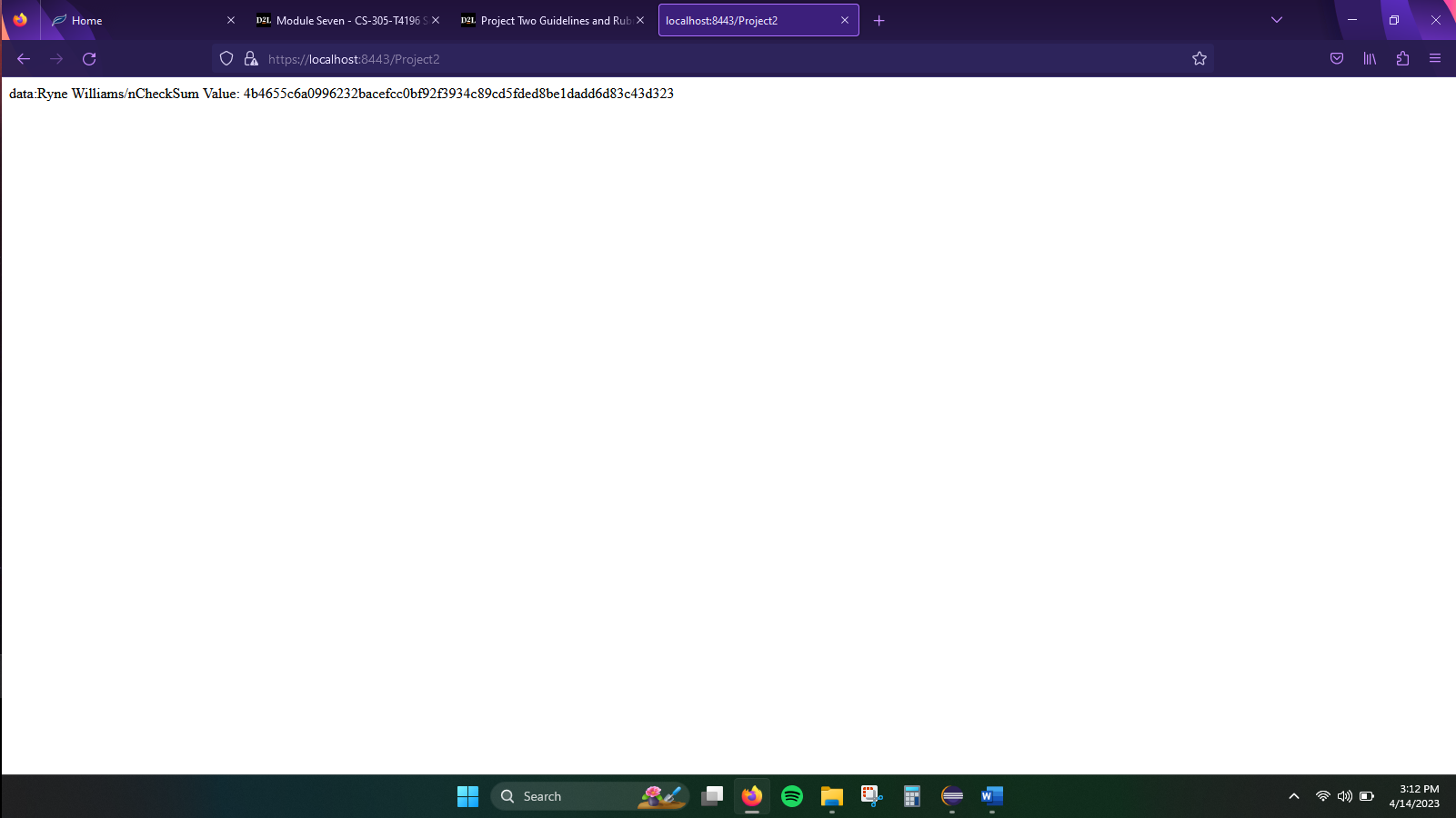
## Deploy Cipher

Insert a screenshot below of the checksum verification.



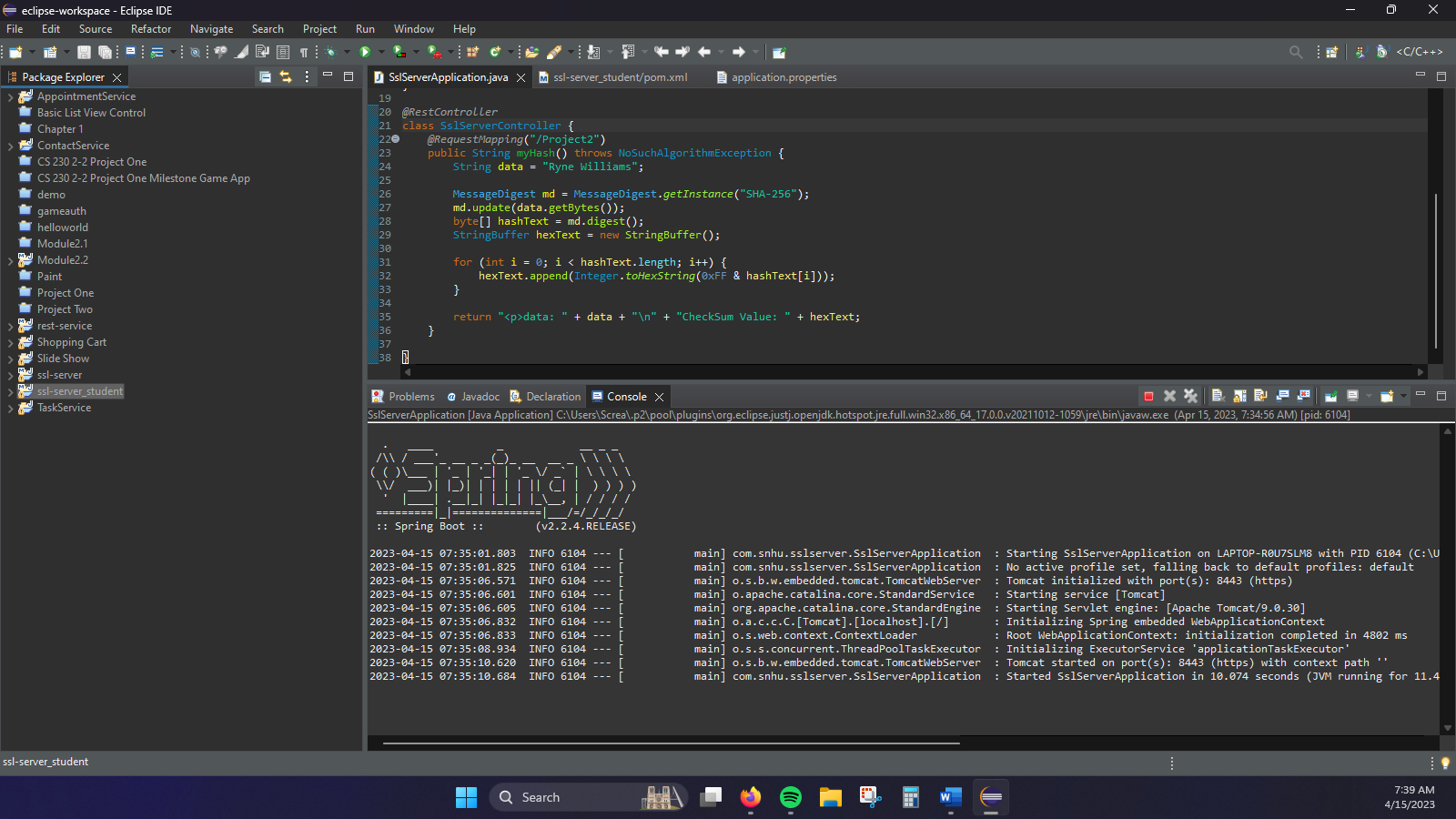
## Secure Communications

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

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

## Functional Testing

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

Text

Description automatically generated

## Summary

This program had various security vulnerabilities that needed to be addressed. Some of these include APIs, Cryptography, and Code Quality. In an effort to increase the security of the code based on the known vulnerabilities I implemented various security protocols. Using RESTful API calls, such as “@RequestMapping” and “@RestController”, allows the program to connect securely to the web through the RESTful API. Also, adding HTTPS security with a self-signed certificate will help to ensure that the data being transferred is secure and unchanged from the original sender. This, paired with encrypting the data through a cipher algorithm, will help to ensure that no attackers can gain access to the data held within the system.

## Industry Standard Best Practices

This code used standard best practices to ensure the security of the program. A secure cryptographic hashing algorithm that has been tested has been implemented in the code to ensure that secure communication is maintained while the system is in use. This helps with data protection by not allowing attackers to gain access to the data. The program also employs the use of a RESTful API platform in order to create a connection to the web that will operate in a secure manner. This paired with a certificate authority adds extra layers of security to the system. Also, checking the vulnerabilities that are present in the dependencies being used allows developers to mitigate potential risks before they can be exploited by attackers. Software security is one of the most important focuses a developer should have when creating code for software. When it is an after thought the risks could go unchecked or unresolved, leading to intrusions in to the system that could steal data or cause the system to operate unexpectedly, if not stop operating altogether. It is vital that software security is built into the system from the beginning to ensure that such risks and attacks are minimized as much as possible and caught before the system is subject to an attack.