

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History[3](#_Toc33111302)](#_Toc33111302)

[Client[3](#_Toc33111303)](#_Toc33111303)

[Instructions[3](#_Toc33111304)](#_Toc33111304)

[Developer[4](#_Toc33111305)](#_Toc33111305)

[1. Algorithm Cipher[4](#_Toc33111306)](#_Toc33111306)

[2. Certificate Generation[4](#_Toc33111307)](#_Toc33111307)

[3. Deploy Cipher[4](#_Toc33111308)](#_Toc33111308)

[4. Secure Communications[5](#_Toc33111309)](#_Toc33111309)

[5. Secondary Testing[5](#_Toc33111310)](#_Toc33111310)

[6. Functional Testing[6](#_Toc33111311)](#_Toc33111311)

[7. Summary[7](#_Toc33111312)](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/14/22** | **Samantha Godwin** |  |

## 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

Samantha Godwin

## 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.

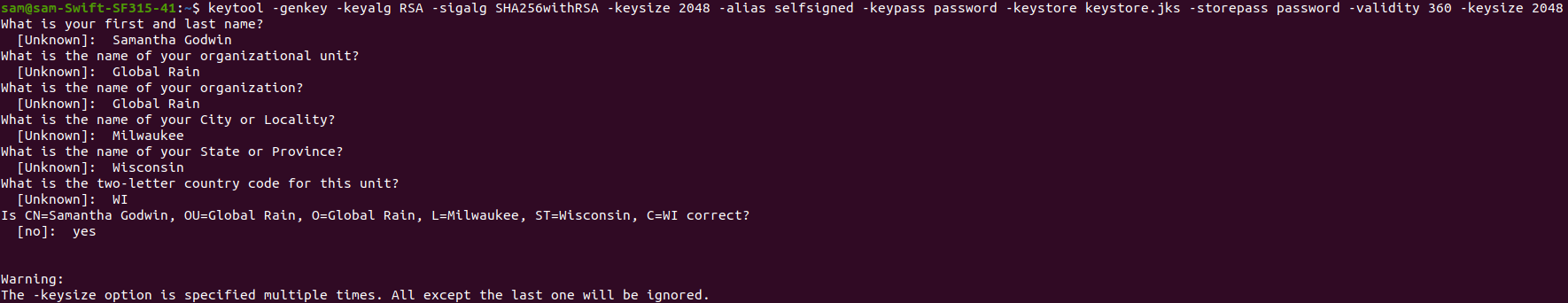
Because the business needs of Artemis Financial require that there is both speed and security during exchanges of data, the hashing algorithm SHA-256 would be a good fit. Short for Secure Hashing Algorithm, SHA-256 produces a 256 bit hash digest value. This value is not reversible, meaning it is not possible to figure out the cleartext fed to the algorithm based on the hash that is created. SHA-256 is faster than older methods while still maintaining security. This secure hash can then be used to compare to an expected hash value in order to ensure that there were no changes to the file, as any changes to the file would change the hash. The algorithm is also commonly used with the SSL encryption protocol. The protocol is symmetric, which is not detrimental due to how the information would be shared in this system.

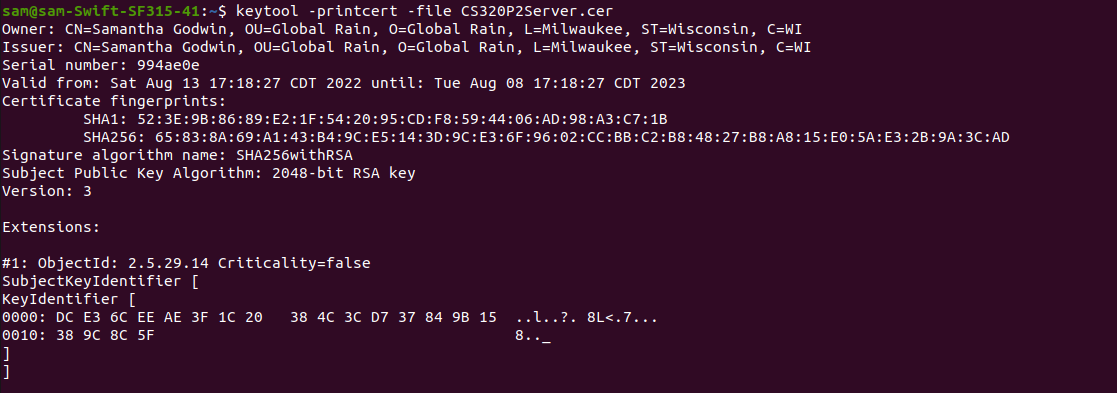
Random numbers are used when generating keys. Alongside this, the RSA-2048 signing algorithm can be used in order to ensure proper encryption.

## 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.

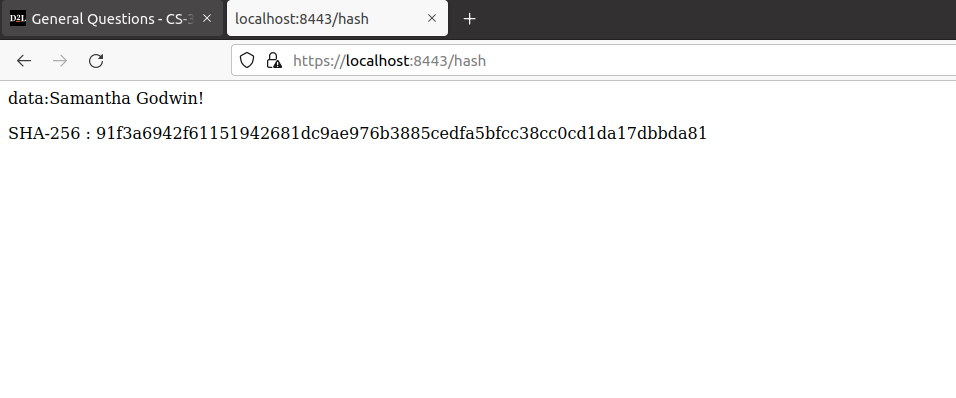




## 3. Deploy Cipher

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.

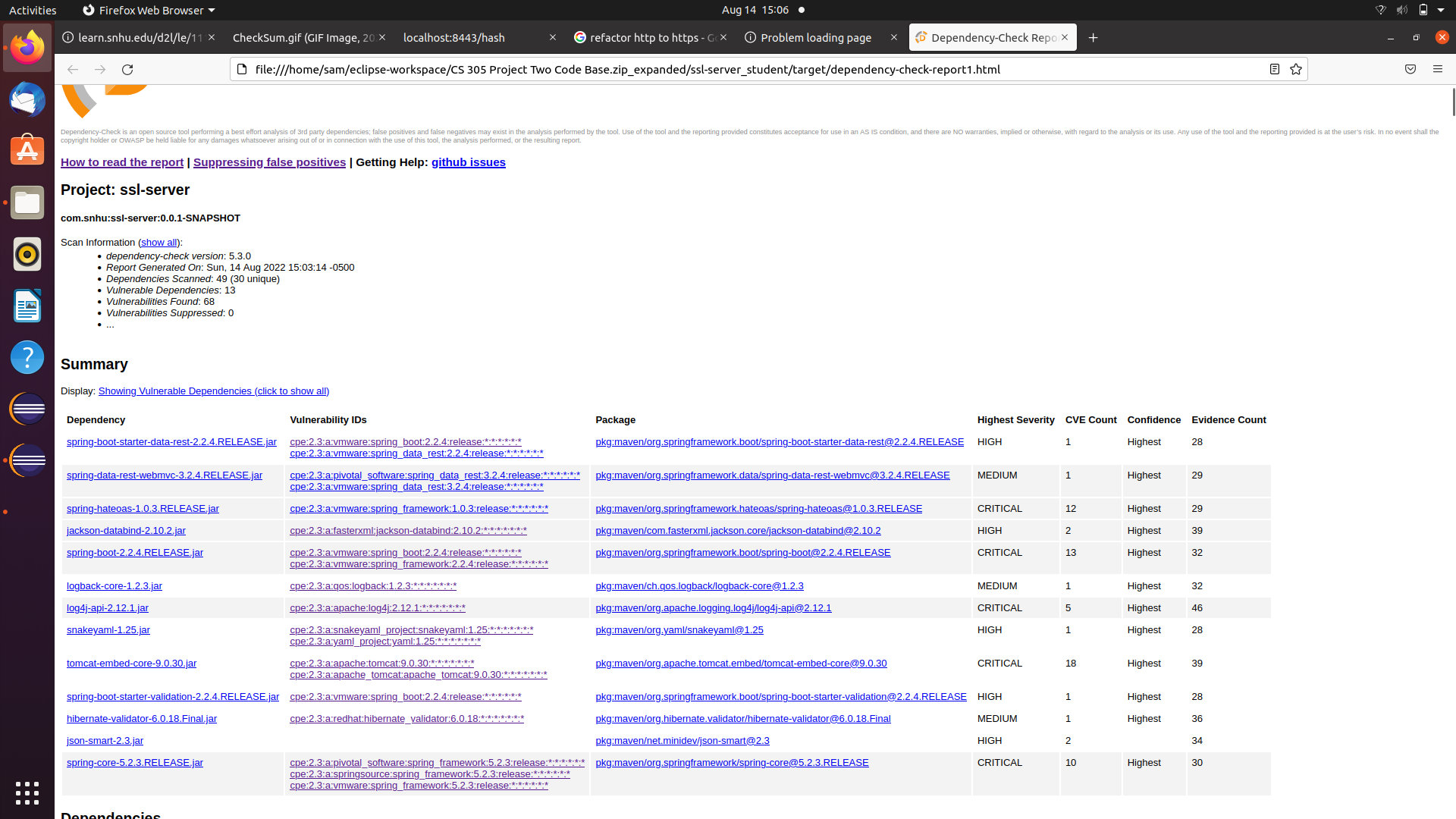
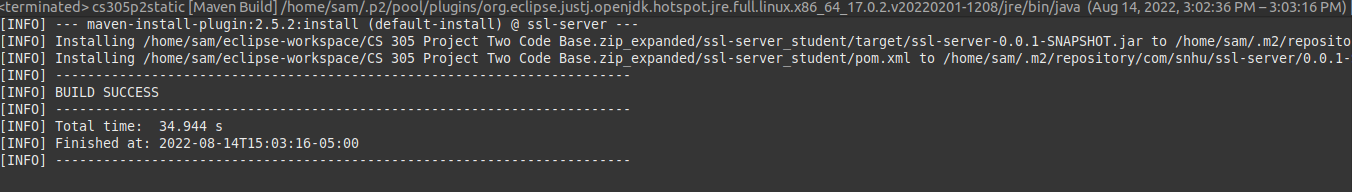
* Insert a screenshot below of the web browser that shows a secure webpage.

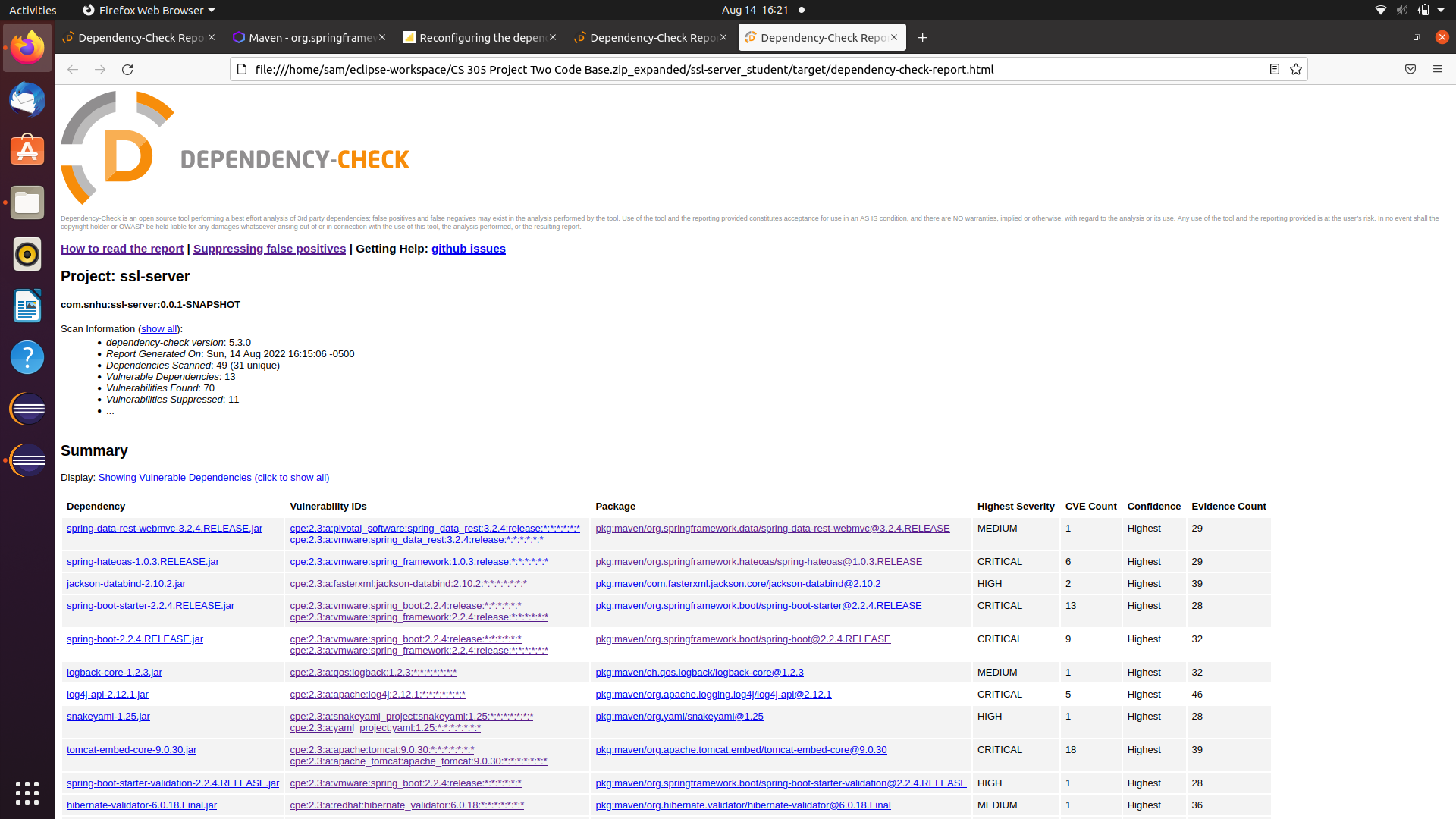
## 5. Secondary Testing

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

[Insert screenshots here.]

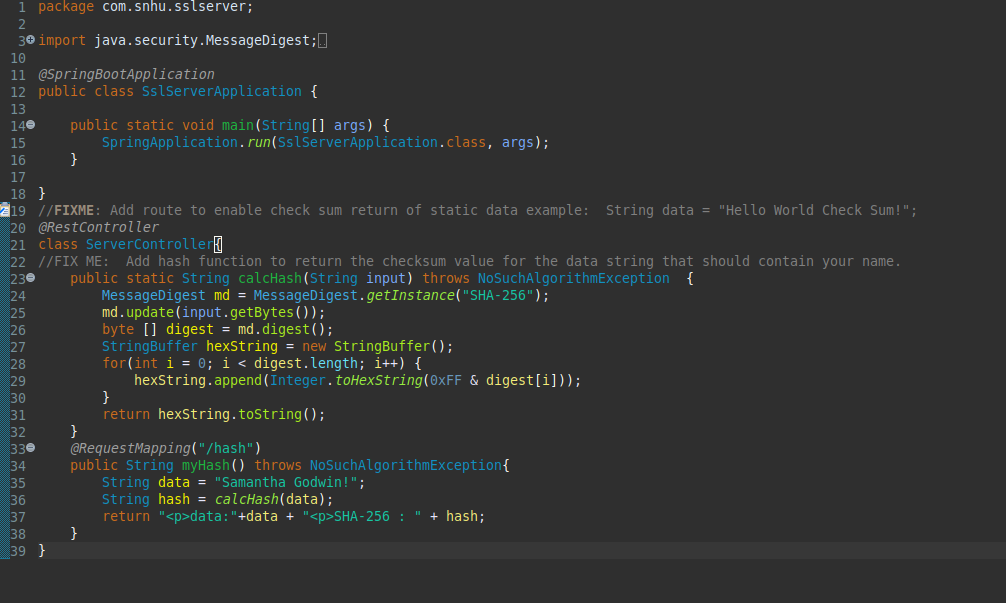




## 6. Functional Testing

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.

In order to refactor the code to comply with security testing protocols, I began by writing a Rest Controller which would ensure that the secure coding section of the vulnerablity assessment process flow diagram. I selected the hashing algorithm SHA-256 in order to satisfy the cryptography category. Used best secure coding practices by using REST formatting, and contained all of the code in a class in order to ensure proper encapsulation. Finally, I updated the version of the dependency check report to 7.1.1 in order to have the most up to date information about potential vulnerabilities.

In order to maintain the system, the dependency check report should be run regularly, as well as after any changes or additions to the system. This will help provide proper warning if any new vulnerabilities are found. The dependencies found within the program should be updated whenever possible in order to eliminate vulnerabilities as they come up.