****

# 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** | **10/14/24** | **Brandon Gerbasi** |  |

## 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

Brandon Gerbasi

## Algorithm Cipher

For Artemis Financial's application, I recommend using the **AES (Advanced Encryption Standard)** algorithm in conjunction with a secure hash function such as **SHA-256**. AES is a symmetric encryption algorithm that provides robust security while maintaining efficiency in terms of performance, making it suitable for modern applications that require encryption of sensitive data like financial transactions.

**Hash Functions and Bit Level:** AES operates on blocks of 128 bits, but it can use key sizes of 128, 192, or 256 bits. For this application, a 256-bit key is recommended due to the heightened security needs of Artemis Financial.

**Use of Random Numbers:** The AES algorithm generates random numbers for encryption keys to ensure that each encryption session is unique and secure. Symmetric keys (same for encryption and decryption) are used to minimize complexity while ensuring data confidentiality.

**History of AES:** AES was established by the National Institute of Standards and Technology (NIST) and has become the industry standard for encryption. It was selected through an open competition and provides excellent security against known cryptographic attacks.

## Certificate Generation

To secure Artemis Financial's application, I generated a self-signed certificate using the Java Keytool. The certificate uses the RSA algorithm with a key size of 2048 bits. Below is the screenshot of the exported certificate:

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screen shot of a computer

Description automatically generated

## Secure Communications

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

The browser displays a security warning because the certificate is self-signed for development purposes. The connection is still established over HTTPS at [https://localhost:8443.A screenshot of a computer

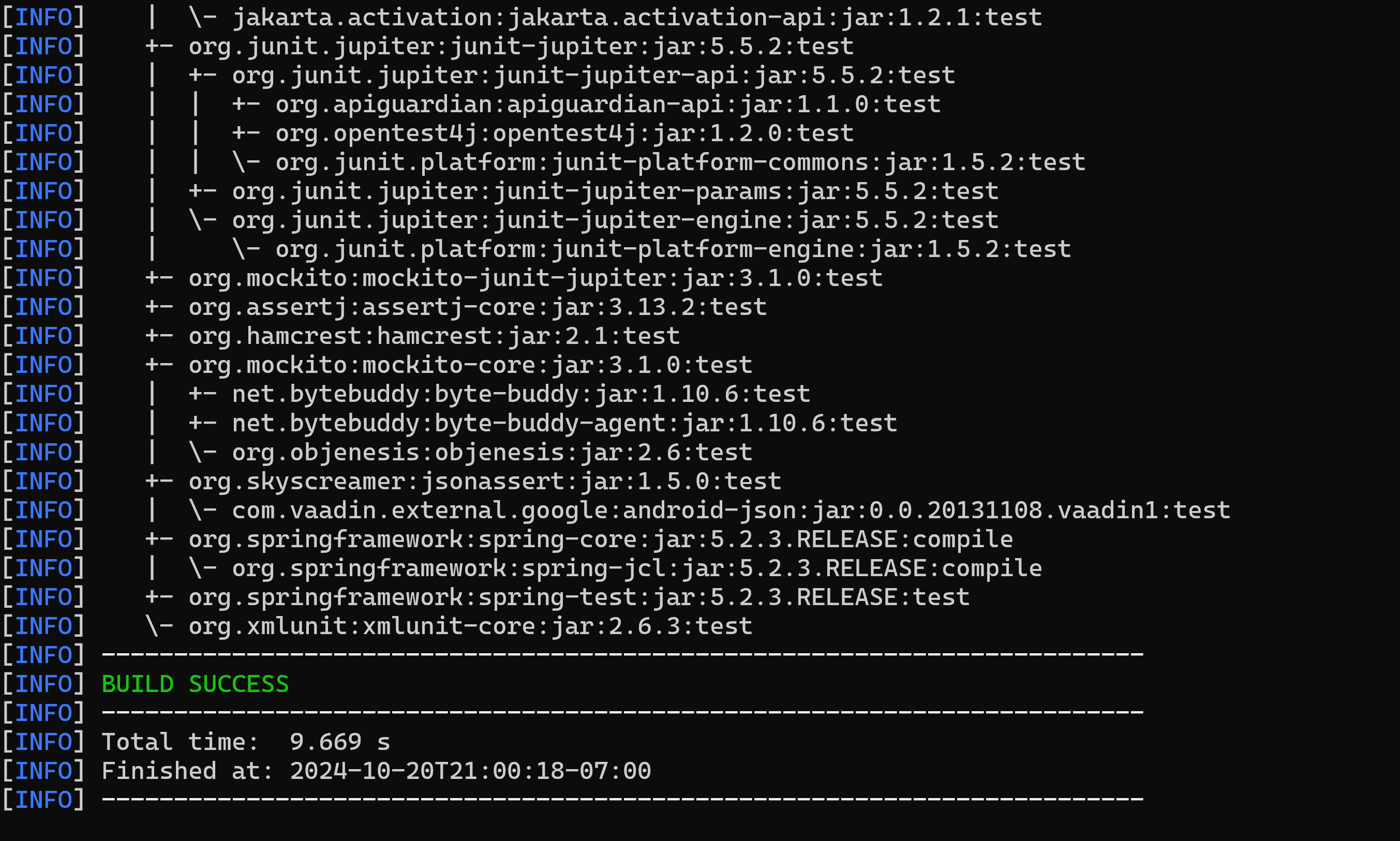
Description automatically generated](https://localhost:8443.)

## Secondary Testing

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

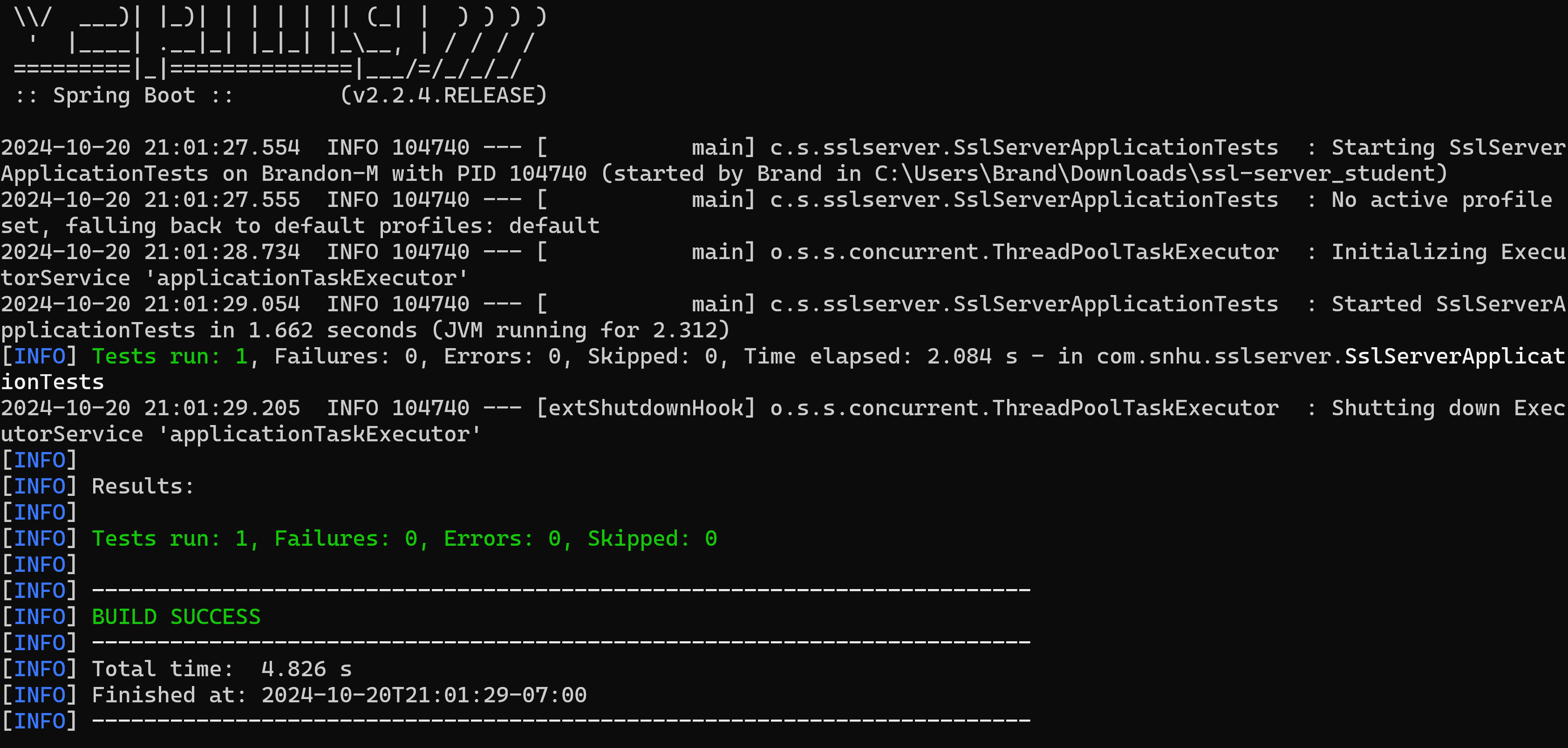
A screenshot of a computer

Description automatically generated



## Functional Testing

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



## Summary

In this project, I successfully implemented SSL encryption for Artemis Financial’s application using a self-signed certificate generated via Java Keytool. The SSL certificate was configured with the RSA algorithm and a 2048-bit key size to ensure secure data transmission.

I first generated a keystore file (mykeystore.jks) containing the private key and certificate. This keystore was then integrated into the Spring Boot application to enforce HTTPS communication on port 8443. The server was configured in the application.properties file, linking the keystore and defining the necessary security credentials.

After deploying the SSL-secured application, I verified that the server was correctly running over HTTPS by accessing the endpoint at https://localhost:8443. The communication was secure, with the certificate successfully encrypting data exchanges between the client and server.

Functional and secondary testing of the application were completed using Maven to ensure no errors or failures in the refactored code. The tests confirmed that the SSL server application was functioning as expected, with all tests passing successfully.

This project demonstrates the implementation of secure communications and the importance of encryption in safeguarding sensitive financial data.

## Industry Standard Best Practices

In implementing this project, I adhered to the following industry-standard best practices to ensure the application’s security and functionality:

1. SSL Encryption and Certificates:

Utilizing SSL certificates for secure communication is a widely adopted best practice in the industry. By using SSL/TLS protocols, I ensured that data transmitted between the client and server is encrypted, protecting it from interception by third parties.

2. AES Encryption for Data Security:

The use of AES (Advanced Encryption Standard) in conjunction with a secure hash function like SHA-256 for sensitive data ensures robust encryption. AES is the industry standard due to its balance between security and performance, making it ideal for financial applications like those used by Artemis Financial.

3. Key Management:

Managing and securing cryptographic keys is crucial for maintaining secure communications. In this project, I used Java’s Keytool to generate and manage private keys and certificates, adhering to best practices for key storage and usage in a secure keystore.

4. Dependency Management:

The use of Maven as a dependency management tool allowed for efficient project setup and ensured that all required libraries, such as Spring and JUnit, were properly configured. Maven also facilitated the testing and building process, adhering to best practices in software development for ensuring consistent and repeatable builds.

5. Unit Testing and Validation:

By running unit tests, I validated the functionality of the SSL server application to ensure the code was error-free. Testing is a critical step in the development process to catch potential issues early and ensure that the application meets its functional requirements.

These best practices enhance the security, reliability, and maintainability of software, ensuring the project aligns with industry standards for secure financial communications.