****

# 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** | **[Date]** | **[Your Name]** |  |

## 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

Gray Lott

## Algorithm Cipher

Encryption is a way of translating information into a secure format that only authorized parties can understand. For Artemis Financial, given the sensitivity of client plans, we need encryption that is resistant to both current and future attacks. Considering these needs, my recommendation is to deploy the AES algorithm, specifically AES-256. Let me break this down.

AES is a symmetric encryption algorithm, meaning it uses the same key for both encrypting and decrypting data. It’s widely considered the gold standard for secure communication due to its balance of speed and security. AES became the U.S. federal encryption standard in 2001.

AES encrypts data in blocks of 128 bits. These blocks are transformed through a series of operations including substitution, permutation, and mixing, ensuring the data is well-scrambled.

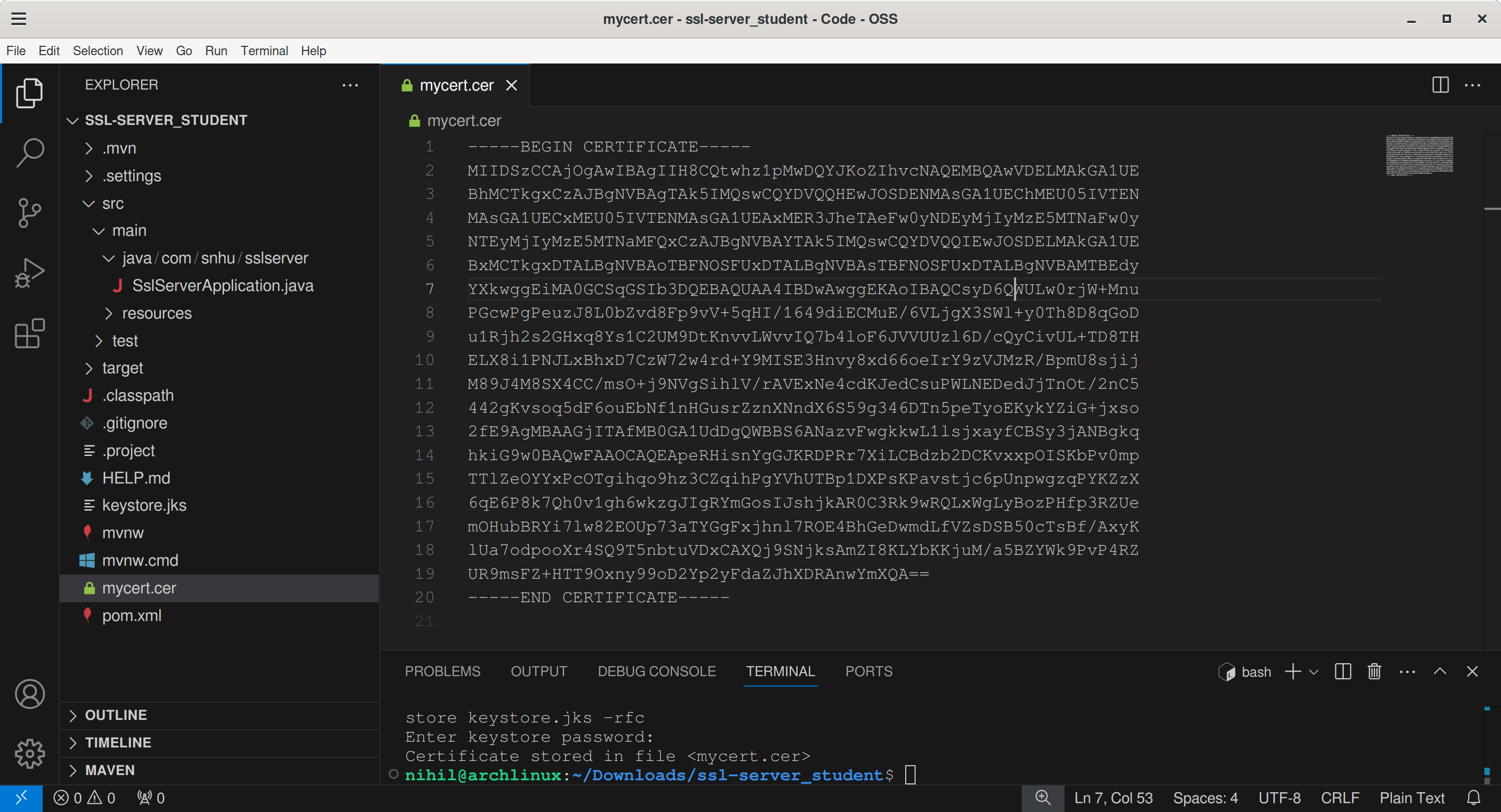
While AES is for encrypting, hash functions are another pillar. A checksum verification is implemented with a cryptographic hash function such as SHA-256. Hashing creates a fixed-length output from any input, and even a minor change in the input results in a completely different hash. This ensures data integrity.

Randomness is critical for encryption. For Artemis Financial, we need a secure random number generator to generate keys. This prevents predictable patterns in encrypted data, a vulnerability attackers could exploit.

Encryption has evolved significantly since the days of simple substitution ciphers. DES, a once-dominant algorithm, fell to bruteforce attacks due to its short key length. AES emerged as a response to these vulnerabilities and remains unbroken when implemented correctly. However, quantum computing threatens to upend the field. For now, AES-256 is resistant.

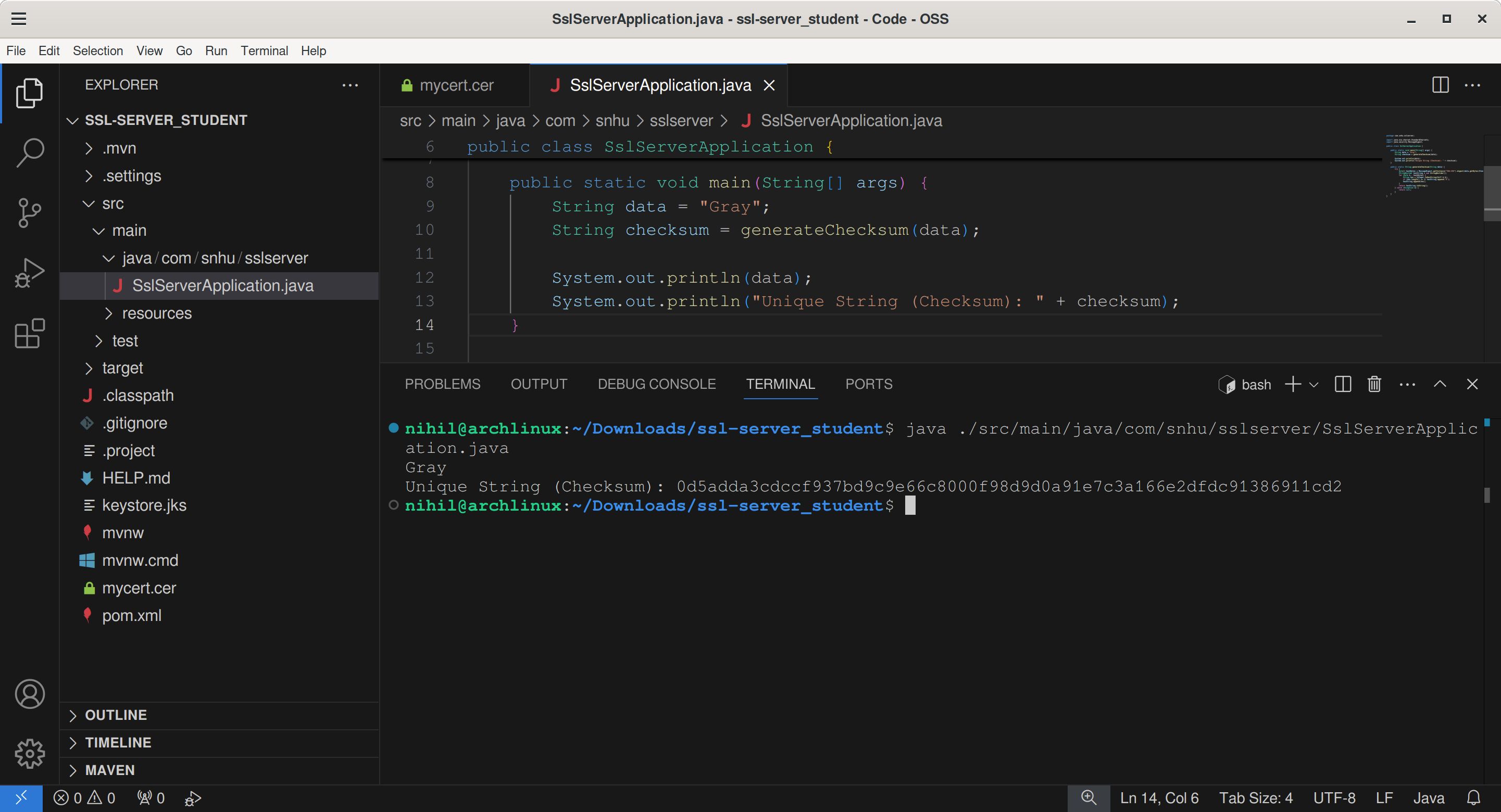
## Certificate Generation

Insert a screenshot below of the CER file.



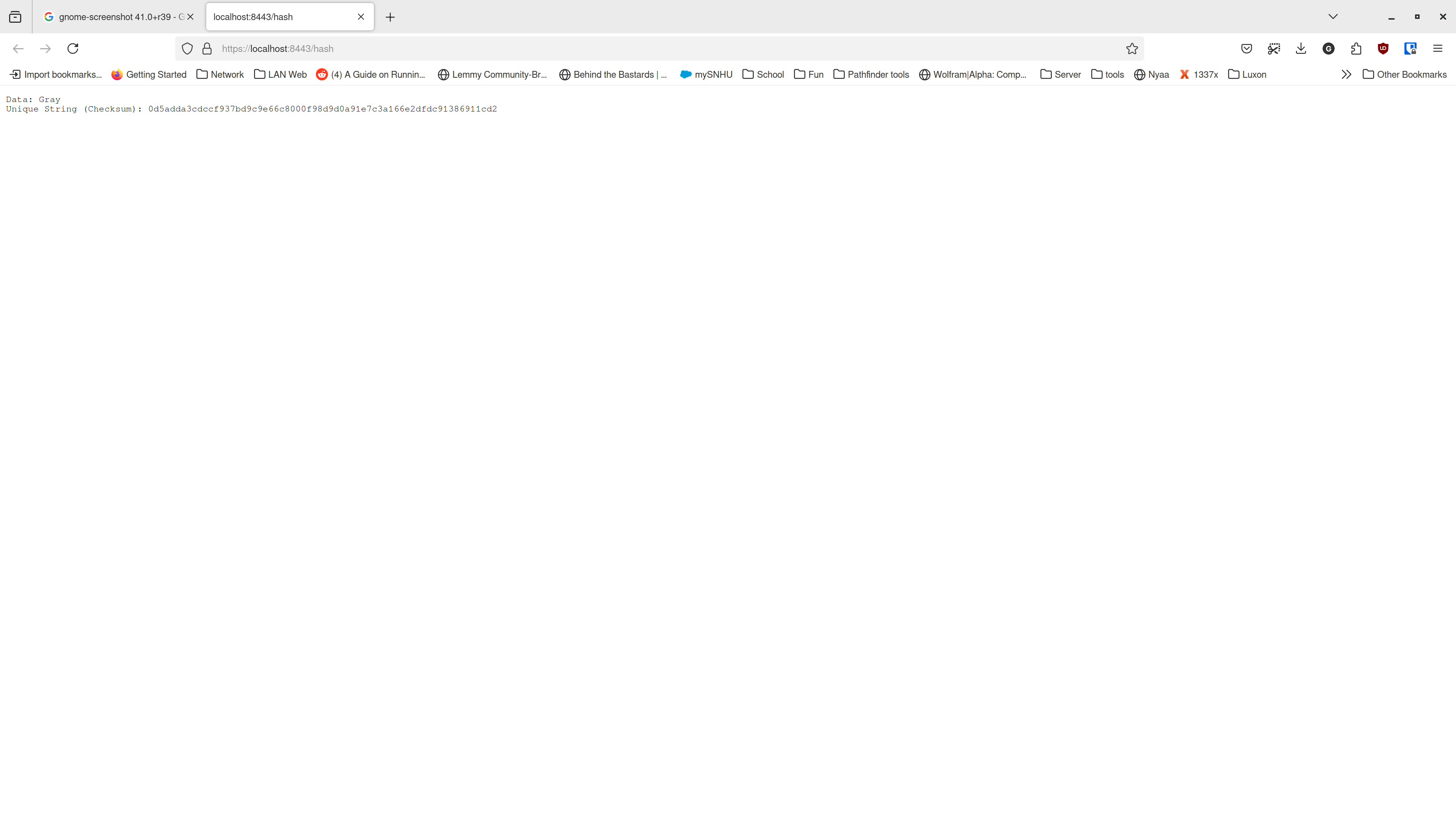
## Deploy Cipher

Insert a screenshot below of the checksum verification.



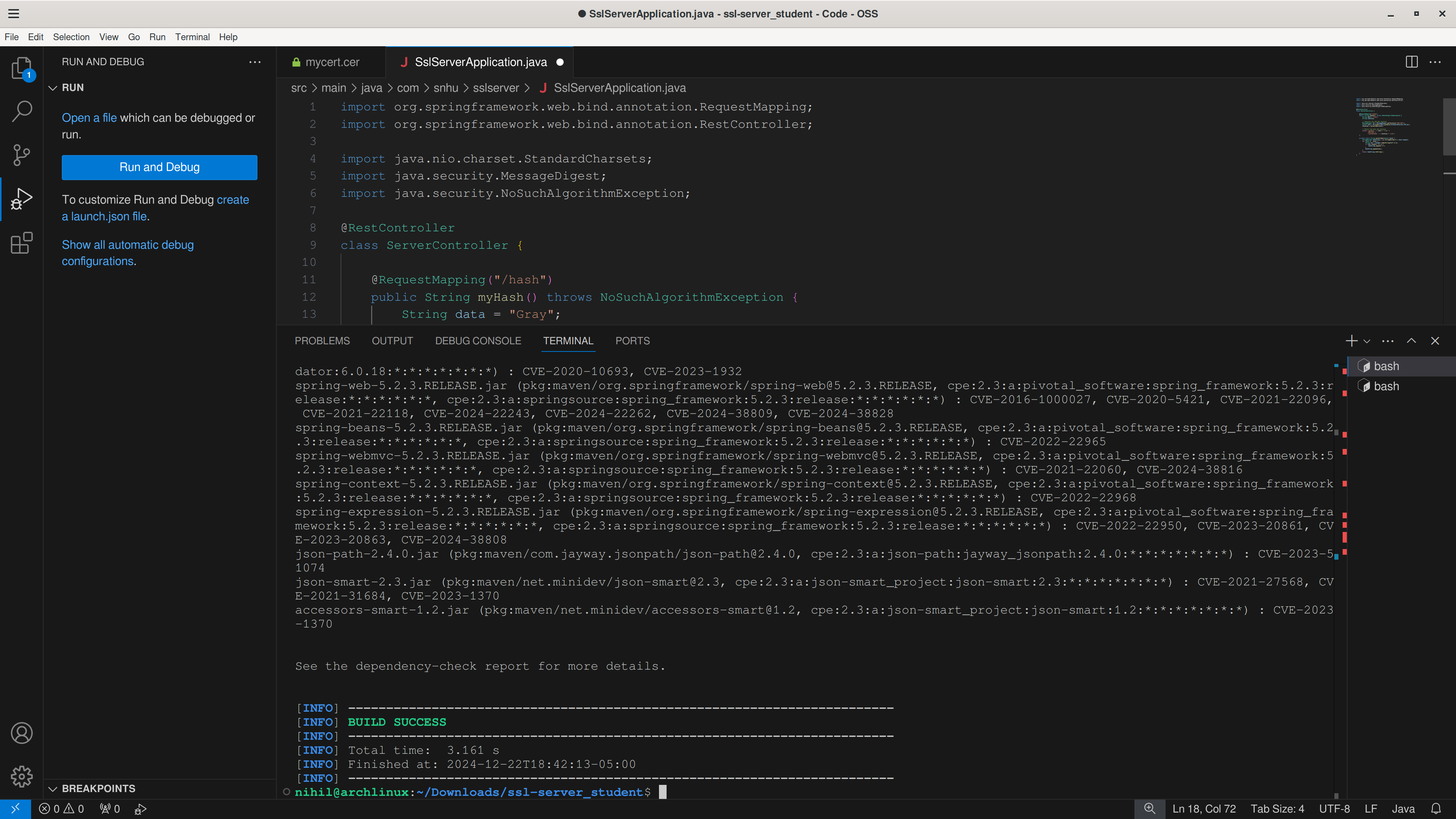
## Secure Communications

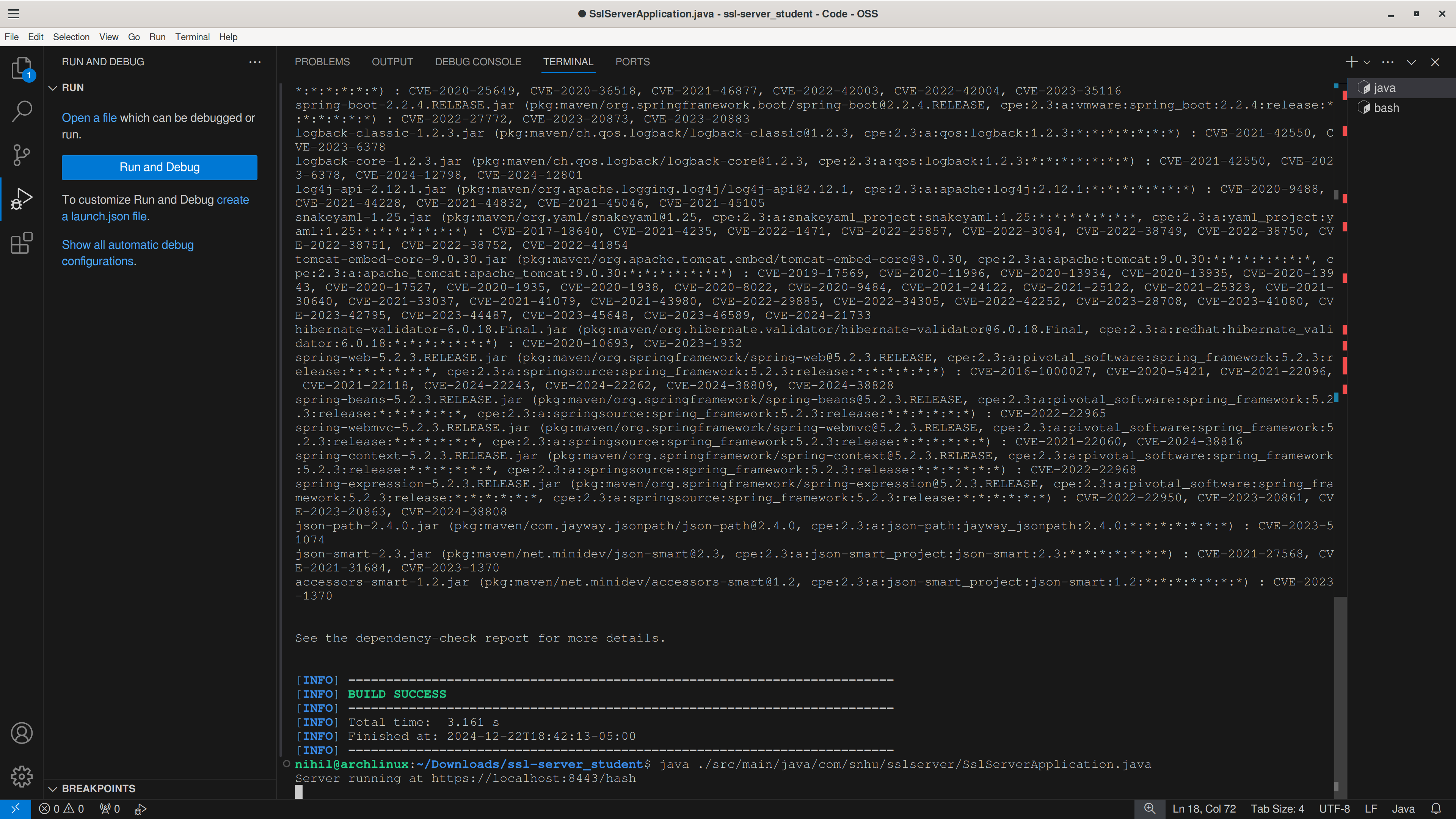
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

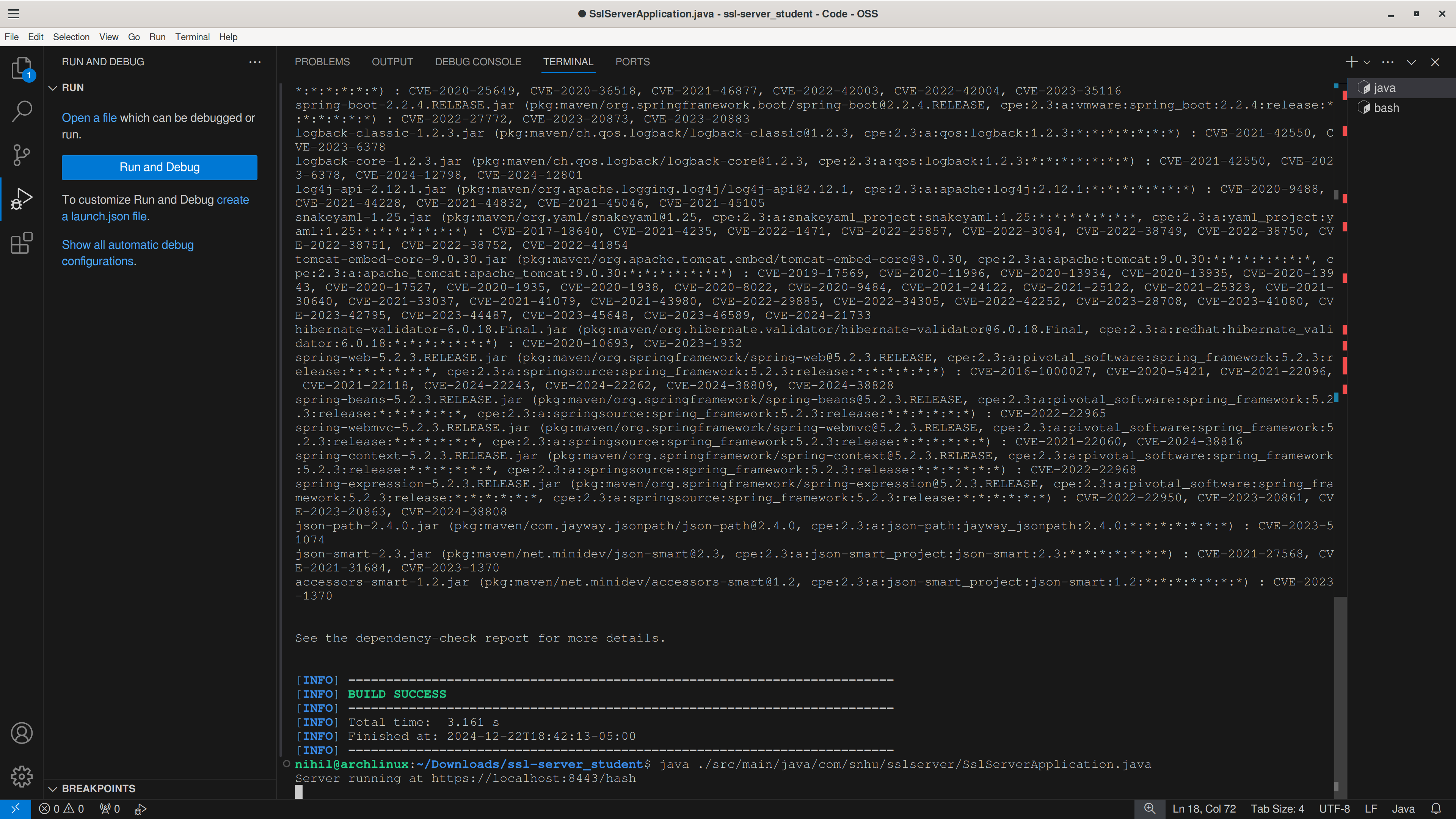
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

The code underwent a review to ensure adherence to secure coding standards. By using the Spring framework, the architecture is robust against common vulnerabilities.

The endpoint /hash ensures minimal exposure by limiting interaction to a single, predictable function. No user-provided data is processed, reducing the risk of injection or improper API usage.

The refactoring enforces cryptographic practices by using SHA-256, a strong hashing algorithm. The method avoids vulnerabilities associated with weaker algorithms

Encapsulation was respected by limiting responsibilities and keeping methods ie bytesToHex private. This minimizes interactions with sensitive data transformations.

## Industry Standard Best Practices

In applying best practices to secure coding, I adhered to established best practices such as using cryptographic standards, enforcing encapsulation, and leveraging the Spring framework.

1. Secure Hashing: SHA-256, a widely recognized cryptographic hashing algorithm, ensures that data cannot be reverse-engineered. By avoiding weaker algorithms, the application remains aligned with current cryptographic standards.

2. Encapsulation: Private methods like bytesToHex limit exposure and potential misuse of internal functionality

3. Framework Utilization: The Spring framework provides a foundation of secure default configurations, reducing common risks like injection vulnerabilities.

Secure coding practices contribute to the company’s overall well-being by preventing breaches and vulnerabilities, thus the company demonstrates its commitment to protecting user data, fostering customer confidence. Adherence to industry standards (e.g. OWASP) ensures compliance with regulatory requirements, protecting the company from fines

These practices not only maintain the integrity of the application but also protect the company's reputation and reduce risk.