# CS 305 Module Two Written Assignment

## Areas of Security

Based on the files **Application.java**, **Greeting.java**, **GreetingController.java**, and the **pom.xml**, the following areas of security are critical for assessment:

* **Input Validation**: The app takes user inputs through the controller, which could lead to attacks like code injection if inputs are not checked carefully.
* **Secure Input and Representations**: Using Spring Expression Language (SpEL) can cause security issues if inputs aren’t handled securely, possibly letting attackers run harmful code.
* **Cryptography**: Encryption isn't directly in these files, but any sensitive data sent through the API should be encrypted to protect it.
* **Secure Error Handling**: If errors happen we must make sure error messages don't give away too much info about the system.
* **API Security**: The app uses the Spring framework for web applications, so it's important to make sure APIs are secure and only allow access to authorized users.
* **Architecture Review**: The structure of the app should be checked to ensure there’s proper separation between components and that data flows securely.

## Areas of Security Justification

* **Input Validation**: The @RequestParam in GreetingController.java takes user inputs that need to be checked to stop injection attacks.
* **SpEL Security**: In Application.java, SpEL (Spring Expression Language) can be used to run code. If not handled safely, this could let attackers run harmful commands.
* **API Security**: In pom.xml, the spring-data-rest-webmvc dependency shows that the app uses REST APIs, which need to be secured with proper authentication and authorization.

## Code Review Summary

* **Application.java**: The SpelExpressionParser() is used here to evaluate expressions. If these expressions aren’t properly checked, attackers could use them to run harmful code. We need to make sure inputs are safe.
* **Greeting.java**: This file mostly handles data (like greeting messages). There aren’t any direct security risks here unless user input isn’t checked before being displayed.
* **GreetingController.java**: This file manages user inputs, and without proper validation, those inputs could be used to attack the system. We need to check the inputs from @RequestParam and @PathVariable carefully to prevent any code injection

## Mitigation Plan

* **SpEL Injection**: In Application.java, data going into the SpEL parser should be checked and cleaned up to avoid attacks. It’s best not to use SpEL for user inputs unless absolutely necessary.
* **Solution**: Use a whitelist of safe inputs or expressions and check everything before it gets processed.
* **Input Validation**: In GreetingController.java, we need to validate user inputs to stop injection attacks. Use Spring's built-in validation tools, like @Valid annotations, to check inputs.
* **Solution**: Set up input validation using @Valid in Spring and tie validation rules to the inputs that the controller takes.
* **API Security**: The API endpoints should be protected with industry-standard security methods like OAuth 2.0 or JWT tokens to prevent unauthorized access.
* **Solution:** Use Spring Security to make sure only authorized users can get to the API endpoints and keep sensitive information safe.