# CS 305 Module Two Written Assignment Mike Brown

## Instructions

Replace the bracketed text with the relevant information in your own words. If you choose to include images or supporting materials, make certain to insert them in all the relevant locations in the document.

## Areas of Security

The main areas of security for this Spring web app are:

* Input validation – all query and path inputs must be checked and constrained.
* Architecture – review use of Spring Data REST and SpEL, since both add risk.
* Error handling – stop stack traces or system errors from leaking.
* Coding practices – avoid unsafe APIs, use least privilege.
* Cryptography – enforce HTTPS and secure key storage.
* API interactions – ensure REST endpoints require proper authentication and rate limits.
* Layered code review – controllers, services, and repositories all need consistent checks.

## Areas of Security Justification

* Input validation is critical because the app accepts untrusted data (name, ID).
* Architecture matters since version 2.6.5 of Spring Data REST has known exploits.
* Error handling prevents array errors or parsing issues from exposing stack traces.
* Secure coding is needed because the app currently evaluates user input directly.
* Cryptography ensures secure communication and the safe handling of secrets.
* API interactions must be secured if repository endpoints are exposed.
* Layered review ensures that no layer reintroduces insecure behavior.

## Code Review Summary

* SpEL injection risk: User input is passed straight into the expression parser and evaluated. This can let attackers run system calls.
* Array index error: /number/{id} does not check bounds, so invalid input crashes the app.
* Unsafe logging: User data is printed to the console without checks.
* Outdated dependency: Spring Data REST v2.6.5 is known to have RCE and DoS issues.

## Mitigation Plan

* Remove SpEL evaluation of user input. Build greetings with simple string handling.
* Add input checks. Limit name to safe characters/length. Validate that the ID is within the array range.
* Centralize error handling. Use @ControllerAdvice to catch exceptions and return clean messages.
* Use proper logging. Replace System.out.println with SLF4J and sanitize inputs.
* Upgrade dependencies. Upgrade Spring Data REST to a secure version (≥2.6.9) or disable risky features, such as PATCH.
* Secure APIs. Require authentication/authorization on any exposed endpoints and add rate limiting.