3. Review

# 3.1 Readability

The code demonstrates strong readability with consistent naming logic throughout. Function and argument names are appropriate, making the code easily understandable. While lacking comments, this was intentional, as we were asked to write code aimed to be self-explanatory. Overall, navigating and modifying the code was easy and didn't require a lot of time to understand what was happening.

## 3.2 Structure

The project structure, integral to my class, was based on an n-tier architecture:

* **Data Layer:**
  + Handles communication with the data sources (MySQL, and the socket)
* **Domain/Model Layer:**
  + Comprises POJOs representing entities from the data source.
* **Services Layer:**
  + Encompasses non-UI logic, acting as an intermediary between the data layer and UI.
* **User Interface Layer:**
  + Responsible for displaying information on screen.
  + Captures user input for further processing.

The code shows good design, providing excellent functionality for users. UI changes are sensible, maintaining an aesthetically pleasing appearance. Parallel programming is implemented safely, and the code avoids unnecessary complexity or future-proofing.

The services layer facilitates segregation between code/user and data, enhancing security measures.

Though lacking sufficient unit tests initially, additional tests were incorporated during the review.

## 3.3 Security

The following good security/coding practices were implemented:

### 3.3.1 Secure SQL Connection:

The SQL classes were already covering the security advice provided in class. Usernames/passwords are **encrypted** and not used as plaintext. Encryption files can be found in /util, and the configuration file for the database is in resource/config/config.properties.

A screen shot of a computer program

Description automatically generated

### 3.3.2 Use of prepared statements:

Prepared statements and stmt.setString were used to protect the data and prevent potential SQL injections.A screen shot of a computer

Description automatically generated

### 3.3.3 Logging:

Logging was implemented for the entire project. No additional implementation was needed.

### 3.3.4 Input validation

The input validation wasn't present in this project.

I have added input validation on the first part of the project but as the socket wasn't working, the 2nd page couldn't be tested and would also need input validation; for searching movies and adding reviews.

**Input validation to add:**

The MovieScreenController is responsible for the 2nd page of the program, which is showing movies retrieved from the server.

* The following picture is the function that allows a user to search a movie: Input validation should be added here.

A screen shot of a computer program

Description automatically generated

* This part of the code is responsible for adding a review on a movie. This review is added to my local SQL database. Input validation should be added here too.

A computer screen shot of text

Description automatically generated

## 3.4 Reflection

My final thought after finishing the review/Writeup is that this project was already good. The different secure measures taught during the Secure Programming were mostly already implemented. The code was well-written, clear, and well-structured.

This project was lacking **testing**. The use of static analysis also showed a lot of **useless imports**. Having added the different badges and **automated** **tests** really enhance the general quality of the code and help the quick review of the project quality.

**3.4.1 Should be added.**

One thing that I haven't done and should be considered is the review of the *socket code*. As the server used for the project is not running anymore, it's not possible for me to verify if the code is working, and I decided not to implement the different fixes.

However, things to consider would be:

* Not using the IP address in clear text like this (As advised on the PMD report)
* Creating tests for testing the connection with the socket and retrieving information.
* A screen shot of a computer program

  Description automatically generatedEnsuring to close the connection when using **ObjectOutputStream** and

The use of FXML might pose a security problem, as shown during the class, the XML is not secure.

Some CVEs found regarding FXML/JavaFX:

* [CVE-2023-34104](https://nvd.nist.gov/vuln/detail/CVE-2023-34104)
* [CVE-2021-3522](https://security.archlinux.org/CVE-2021-3522)
* [UI Discrepancy for Security Feature](https://security.snyk.io/vuln/SNYK-JAVA-ORGOPENJFX-5788302)
* [Oracle » JavaFX: Security Vulnerabilities](https://www.cvedetails.com/vulnerability-list/vendor_id-93/product_id-21383/Oracle-Javafx.html)