Team 14 Verification and Code Review
Sprint 6 (November 21 – 25)

Jeffrey Li | Christian Sarran | Kelvin Duong | Zifan Gao

Code Review Strategy

Each team member is responsible for a part of code. We do code review regarding the code's functionality, syntax, style and architectures.

Code Review Checklist

- Is the code it easy to understand?
- Does the code do what it's supposed to do?
- Is there dead-code or debug statements?
- Are exceptions and errors handled appropriately?
- Is the documentation good enough?
- Are the naming conventions appropriate?
- Is there a lot of bad copy and pasted code?
- Are constants used when possible?
- Does the code correspond to our original design?
- Does the code follow SOLID principles?
- Consistent use of whitespace?

Code Review Summary

Christian

Zifan

Code Review Summary of:
src/main/java/com/devlopp/teq/reporting/GenerateReport.java

The class of GenerateReport created three more preset query functions that can generate different types of charts. The logic of the code is clear and easily understand. It is well functioned and can do what it supposed to do. The try and catch will avoid error generating and is the correct way to handle the functions.

However, one problem with this piece of code is some auto-generated catch block TODO comments are not removed. Overall the commenting did not include much detailed that it should include. This is the weak part that can be improved.

Jeffrey

Code Review Summary of:
src/main/java/ui/Main.java
src/main/java/ui/Controller.java

Despite the lack of commenting or documentation, the front-end code for the UI controller is easy to understand. Each method corresponds to a single button or action done by the user, so

it could be understood what a user action does to the UI. The code could be better improved by implementing some sort of observer pattern for all three UIs, and the controller class should be split up to correspond to each FXML file. Since there is a lot of if-statements and repetitive code, helper methods should be implemented to better reduce the amount of unnecessary lines. Commenting would better help on describing the context of each group of code in the controller.

Kelvin

Code Review Summary of:
src/main/java/com/devlopp/teq/databasehelper/DatabaseValidHelper.java

The code functions as expected and is easily understandable. No dead code or copy and pasted code. Exceptions are handled properly by a try and catch statement. Code is properly documented with docstrings, although there may be too many comments. For instance, the name for the constant “MIN_PASSWORD_LENGTH” already explains what it is for, but there are 3 lines of comments above it. Some functions have comments that are longer than the actual function itself. Constants are used near the top of the file. Consistent usage of white space and appropriate naming of functions.