Team 14 Verification and Code Review
Sprint 5 (November 14 – 18)

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

Code Review Summary of:
src/main/java/com/devlopp/teq/databasepreset/DatabasePresetQuery.java

Code is easy to understand and requires little commenting and is commented where needed. Doc string is complete and well done mostly, some of the more complicated variables like serviceType, should be better name or documented to better represent what you want as we have a value in the DB with the same name representing something different. The code does with is needed, except some function need to be able to be more specific, for example get age range, should be able to be specified based on their location or some other factor.

There is no bad use of copy and pasting code nor magic numbers, Formatting and indentation is appropriate. There is no dead code and white space is used appropriately. Code follows SOLID well, except perhaps splitting some queries based on single responsibility principle.

Zifan

Code Review Summary of:
src/main/java/com/devlopp/teq/databasepreset/DatabasePresetQueryHelper.java

The code is well formatted, meeting indentation and whitespace standards and free from parse errors. The code did what is supposed to do. It includes helper functions to help the implementations in *DatabasePresetQuery.java* effectively. These helper functions are necessarily included in this class. It is nicely commented and can easily understand. I think overall this part of code is good because they are correct and effective solutions for the project requirements at hand.

Jeffrey

Code Review Summary of:
src/main/java/com/devlopp/teq/parser/TemplateParser
src/main/java/com/devlopp/teq/parser/ClientProfileParser
src/main/java/com/devlopp/teq/parser/CourseSetupParser
src/main/java/com/devlopp/teq/parser/ServiceParser

The purpose of these classes is to parse the data read from the iCARE excel files into data objects that can be inserted into the TEQ database. The use of an interface follows SOLID design, as that would allow the parser package to be built-on for newer template types. The use of the FieldParser helper class and the builder patterns do allow the code to be more readable, but a major source of code smells comes from the use of the allData HashMap object. There is an enormous amount of repeated code lines that simply map a String to a new ArrayList in the constructors of the implemented parsers. To resolve this, either a for-loop or a container class must be implemented to drastically lower the amount of code in the parsers.

Kelvin

Code Review Summary of:
src/main/java/com/devlopp/teq/security/PasswordHelper.java
src/test/java/com/devlopp/teq/service/MockService.java
src/test/java/com/devlopp/teq/service/TestService.java

The code is easily understandable, performs as expected and commented well. The names of the functions explain what they do. Exceptions are handled properly by a try and catch statement. Docstrings explain the function of the code and functions are small (less than 10 lines). Good use of mock objects for testing. No dead code other than one unused import statement in TestService.java. No copy and pasted code, and spacing is consistent throughout. Follows SOLID principles where a class only performs a single function. Performs as required according to our original design.