# Team 3 Code Review Summary

Team 3 Code Review Video: https://www.youtube.com/watch?v=zsoqbUJpsxY

| Team Member Name | Code inspection summary |
|---|---|
| Zhili Pan | Assigned code: NeedAssessRefChecker.java, TemplateChecker.java<br><br>The code overall is concise. There is no repeated code, no late exit, no unused code, not too many dependencies and no hard-code values. The code style is good.<br><br>TemplateChecker is an interface for all concrete checkers and is used in other places. NeedAssessRefChecker implements TemplateChecker and is not directly referenced as a type in other places. That makes the code more extendable. The TemplateChecker provides stateless services. It has been made as a Java Bean (singleton), which follows the creational design pattern.<br><br>However, the code doesn't follow the template method design pattern (guideline  8). The checkNotNull method is almost completely independent from other templates. However, our current design requires developers to re-implement that code when they want to add a new checker. So, we can make an abstract class for all concrete template checkers so that the code can be reused.<br><br>The test cases is created for the classes |
| Jacob Buol | Reviewing: TemplateReader.java<br><br>For my review, I am giving recommendations on the TemplateReader.java file. This code has the main logic for reading the client's provided templates and saving the data to our database. For the most part, the code follows good programming practices and it efficiently does what it is meant to do. However, there are two things that we could improve on this code. Below are the details.<br><br>The first thing that I would recommend is adding comments into this code. There are no comments in the code so it would take time before a new developer can understand what is going on in the code. This falls under our guideline number 7 which talks about following correct programming conventions.<br><br>The second recommendation is that it would be much better to |

| | |
|---|---|
| | declare a specific variable once then assign it as many times as possible within a loop instead of declaring it each time.<br>An example in the code is the line<br>"XSSFRow row = mySheet.getRow(i)" inside the for loop. This can be improved by having the variable 'row' declared outside the loop, and then performing the assignment within the loop. Based on our guideline number 1, we should avoid repetition in the code wherever it is unnecessary.<br><br>Other than the two recommendations above, the code is good overall. |
| Michael Sun | Associated Code: DashBoardController.java<br><br>For this code review, I will be looking at the DashBoardController class, more specifically, the data method that queries the MongoDB database, which fetches data and creates CSV file.<br><br>From the guideline 6, we can see that on line 102 we are checking if the template does not equal NeedAssessRef. This doesn't follow the open/closed principle from SOLID as this will make it very tedious when we add several more template types. A good change here would be to create a set of strings with different templates and simply check if the string template is in the set.<br><br>Following guideline 7, although nitpicky, there are several empty lines in the try catch blocks that make it harder to read the code.<br><br>I'd also recommend a bit of commenting as well. It doesn't have to be for every line but just for sections of code to explain generally what is happening. For instance, lines 172-181 makes the user download the CSV file from the application to get the data. |
| Gurpreet Gill | Based on our code review guidelines, there are a few things that I noticed during our code review. Specifically, for the part I was assigned (dashboard.jsp), I noticed that guidelines 6 (Hard-coded values) and 7 (Language style and Coding conventions) were violated.<br><br>For guideline 6, I noticed that in the file there are hard-coded values for checking user-permissions. For example, the line "`<c:when test="${loginUser.role=='UTSC'}">`" can be modified so that instead of checking for the string value 'UTSC', it would instead have a case for the variable itself rather than the value of the variable. This way, the code becomes more maintainable for future use in the event that the role names change, as developers would only need to change the value of the variable |

| | |
|---|---|
| | rather than changing multiple instances of the string in the code.<br><br>For guideline 7, I noticed that in the file there was no separation of style and content (consider the navigation bar, for example). In the case of the front-end, we could instead use external CSS style-sheets in order to make easier modifications to the styles.<br><br>Besides these recommendations, I noticed that there were a few instances where the codebase followed good practices (i.e. I noticed that we follow guidelines 5 very closely by writing variables to be private and only increasing the scope when necessary). |