Team 3 Code Review 2 Summary

Team Member Name	Code inspection summary
Zhili Pan	Assigned code: dashboard.jsp. DashBoardController.java (addAccount method) The code for adding new account feature works properly, but there are some code can be improved. For dashboard.jsp, there is an useless text section at the bottom of the page, which contains a overall description about the system. This section is unrelated to the new account feature and user experience. The information is needed in the welcome page. After user login, the information may be misleading since some features are not available for some users. So developer can consider to remove the unused code (guideline 4). For the java code (addAccount method), the current solution doesn't use the created exceptions, which makes the exceptions to be "unused code" (guideline 4). Besides, there is no need to explicitly interact with session attributes (guideline 5). Developer can create exception handlers for the exceptions and use http redirect to display corresponding pages. By doing that, the code can have less dependency and get rid of unused code. Last but not least, developer should create UAT test cases for adding new account feature (guideline 9).
Jacob Buol	Assigned Code: LoginController.java In this review, I am giving recommendations on the code in the file LoginController.java. This class handles authentication for active users of our system. The code works fine and authenticates users as expected, however, based on the programming techniques we learned during the lecture on code smells, we can improve on this class. The main thing that we can improve on in this code is getting rid of some redundant comments. Some lines of code are self-explanatory and may not need any comments to explain what is being done (guideline 7). For instance, a line such as 'session.setMaxInactiveInterval(60*30)' can be well understood and does not need comments.We would therefore remove such small inconsistencies to improve the quality of our code.
Michael Sun	Associated Code: DashBoardController.java

	For this code review, I will be looking at the DashBoardController class again and the overall design of the class. In our controller package we have several java classes to handle the functionality of the website with things such as displaying certain pages, logging into an account, logging out, etc. The dashboard controller displays the dashboard page but it has too many responsibilities and can be refactored into different classes. It is responsible for displaying the dashboard page, uploading a template, retrieving data, and adding a new account. This can be improved on and is not consistent with the other classes in this package. Particularly the data method in DashBoardController makes a query to our database but this is not done with our file upload as it makes the query outside of the controller package so there is an inconsistency here with our design.
Gurpreet Gill	Assigned code: NeedAssessRef.java For this code review, I looked at NeedAssessRef.java, the file that contains the getters and setters for the template. Specifically, the length of the file raises maintainability concerns for future templates. In the context of the features of our project, this file would not be a big concern. However, in the context of future extensibility, this file poses a major problem as we are manually getting and setting each field of the template. Some templates have hundreds of fields to get/set, and this file alone is 626 lines for just one template. This violates guideline 8, where the architecture and design of the class could not be easily re-used. Instead, we could look into alternate methods of handling templates based on client needs and consider making the product more dynamic rather than static.