**Report for Team Commit**

This is a quality report for the project of Team Commit. This paper will review on the group’s GitHub, the group’s ATDD test cases, the codes used for the project, and the design patterns that were used.

**GitHub Review**

The group’s GitHub is great. The GitHub is organizable and readable. The folders are separated accordingly which helps in navigating through each files present in their GitHub. Furthermore, the name of the files and folders are easy to identify. The documentation provided is also great. The documentation is easy to understand since the instructions were clear and the separation of sections with headers were helpful. Even though the group’s GitHub and, there are still some parts that requires attention. One particular example is finding the executable file which is a minor issue. According to the repository’s documentation, the executable is in the bin/release folder. The issue is finding to the bin folder, which is within three directories labelled “HELLs FrontEnd”, so the documentation should at least mention this like “Hells FrontEnd”/“HELLs FrontEnd”/“HELLs FrontEnd”/bin/release/“HELLs FrontEnd.exe”. Another thing to note is having the three folders with the same name so it may be helpful to have a name that helps distinguish between the folders. Like what was said earlier, this is just a minor issue that does not necessarily affect everything since we did not have much problem finding the executable file but it is still one thing to note on. Another thing that requires attention is how to be able to work on the project. The documentation only describes on how to use the application but does not describe the programs we need to run the project or how can we make and see the changes to the project if a different group will pick up where the group left off. Although there are some issues that are present in the group’s GitHub, but overall, the group’s GitHub is great and the changes required can easily be added.

**ATDD Test Case Review**

The ATDD was relatively easy to find on GitHub. It was located in directory for milestone 6 which is their latest milestone. The ATDD test cases can help with the cleanliness of the code by ensuring the code works in a way that the developers had intended their software to behave.

In their ATDD test cases, while most of these test cases passed, some improvements are needed. For instance, there were issues involving the message prompts. This is apparent when logging in using incorrect credentials. In the ATDD test cases, it mentions how the error prompt should indicate which of the two fields, whether it is the username or password that is incorrect. Instead, the error prompt is the same among all the cases. This implies that the ATDD test case should be updated. In addition, another error prompt that needs to be corrected is the analyst approvals, where no message prompt is provided to the user when no software is requested for access.

There were also some missing ATDD test cases that should be tested. An example involves the escalation step where the analyst gives access to the approver to approve a software request. This step in the process is completely omitted, despite it is part of the process flow for the user to receive access to the software. Lastly, there appears to be a color coding system in this software that should be verified via ATDD. The ATDD does not mention anything relating to this color coding, yet this color coding does change with respect to the status of a particular software request.

**Code Review**

The code is very well structured, clean, concise. In this way it is possible to understand what is being done in each fragment of code even for those who do not have knowledge of the languages used, these being C# and PHP. Another positive point is the name of the variables and functions, which are meaningful, that makes the code easier to understand. Finally, the scope of each class is well determined, these are neither too long nor too short.

However, in our understanding, some things could be improved. First, the name of the UI components are with the default names provided by the framework. Like variables and functions, a name that describes the component in the context of its application would facilitate understanding. For instance, *Btn\_Submit* would be better than *Button1* since once you saw the code you would know that it is the actions to be performed when clicking the submit button.

Another point for improvement is the handling of user passwords on the login page, which is written in PHP. Currently you are using the *GET* method to return the password, thus the content of the variable is displayed in the *url* and is visible for all. For this reason the recommendation is to use the *POST* method, which makes the value of the field invisible for all.

The last point we would like to point out is not about the code itself, but about the organization of the project. You have files in different folders with the same name. In industry there are several naming patterns to differentiate the files and make explicit what makes the code they contain. For example, files that end with the acronym DAO, which stands for data access object, as the name suggests are responsible for accessing the database. If we think then the analyst, we could have the *analyst.cs* file to describe the class and contain the getters and setters for the encapsulation and we could have the class *analystDAO.cs* that would be responsible for integrating with the database for this class.

**Design Pattern Review**

The usage of design patterns will simplify and organize the code during the development of the entire project. Three design patterns are applied to this project, which are the factory pattern, singleton pattern and reflection pattern. The connection and adjustment for each pattern are good enough and all of these patterns are used properly.

In terms of the factory pattern, they considered the situation that different users may have different styles about their accounts. If they create different classes for each users, it would consume lots of time. In this case, they create a factory class called “HttpClientFactory”. If people try to sign up a new account, the “WebClient” class will use this factory to create a new account that will satisfy all the requirements that clients have. This pattern also helps developers avoid duplicated code and improve the quality of the project a lot.

Next one is singleton pattern, the application of this pattern is to ensure a class has only one instance, and provide a global point of access to it. More clear is that, clients can not log in the same account to edit the same file in order to avoid the confusion and chaos. In the project, they use this method during the sign in page, and only one user can log in the same account at the same time. If this project want to further develop in the future, this method will also same the memory usage for the system.

Last but not least, reflection pattern is also applied to change the type of session at run time. Reflection pattern is usually used to examine or modify the behavior of methods, classes etc. For the project code, they tried to get the information of users and set the login sessions during the run time. However, the code is hard to find where they applied this pattern. All we can find is that, they tried to use the information of uses from the database. The data may be accessed through an easier way.