**Software Construction Strategy**

We chose the programming language which most of our developers have experience using, Java (Update 8). Considering that all of the developers in our group only use either NetBeans or Eclipse for desktop (latest editor) as a programming platform, we chose to use whichever one each of us is comfortable with.

With regards to the prioritization of tasks, starting off with the logic of the system will be the main priority of our team. If ever we started with the user interface first, and the client would like to see the progress done so far, we would only be able to present a featureless skeleton consisting of panels and frames; but with the logic being worked on first, we would be able to show the main features and functionalities of the system currently being implemented as requested by the Client’s specifications.

Working on the system will be done per module/user story, implementing each of their respective functionalities. When most of the key features are done, we will then merge them into the main menu of the system. Building each module separately will save time when it comes to the overall development efficiency as it does not affect the functionality of the entirety of the system. With each of the module being individually built, they can be tested/debugged separately without any worries of affecting the other modules’ progress.

There will be periodical checks regarding the overall progress of the development of the system. Each developer will be assigned modules to work on. During the periodical checks, the lead Developer will then assess the respective modules currently being worked on by every developer, and whether the coding standards were followed or not. The QAs will then proceed with inspecting the lead Developer’s assessment, with the overall progress of the system, while giving critiques as according the SRS, the client’s requests, as well as the set coding standards.

During the first iteration/initial build of the system, error handling regarding user input will be first taken lightly. As the system gets further into the development, error handling will be applied to modules/features which require, but not exclusively, user input. User inputs involving data requirements or information that is frequently used in the system will be cross referenced to the system’s database.

A combination of both individual and pair programming will be implemented for the work delegation. Each developer will be assigned modules; there are however some modules that may require the cooperation of 2 or more developers. These modules will be assigned as pair work as well as the other modules marked with high priority. The remaining modules that are easy enough for one developer to code will be assigned individually.

GitHub will be used to store the latest version of the code being currently developed. Further uploads will be first cross referenced with the latest one to see any revisions or changes made. All revisions done will be based off the original code located at the main branch of the repository. The uploading of code will only be permitted after the periodical checks to avoid confusion or untimely revisions without the attention of the other developers.

As we are already using GitHub to store our codes and other documents, all we have to do is to create a branch separate from the master where we can upload our codes. Whenever an update regarding the progress of the system development is to be uploaded, all we have to do is upload it to the code branch, compare it to code located at the master branch (where the latest/ QA and DEV approved code is located), and then pull a request. We can also use the various compare functions available in GitHub to review any changes made into the code.

For overall reusability, non-standard language features will not be allowed. This is to avoid any further complications or problems we might encounter once importing the software to the client’s system. Rather than relying on third party resources, it would be better to just use the ones locally available.

Test cases can be done after the module has been made. Considering that test cases are not usually followed prior to the actual coding, checking for glitches or bugs along the way will be more time and work efficient. With the frequent periodical checks, developing code as according to test cases may lead to hurried or unfinished code. All of this considering that the developer knows exactly how a module should work; otherwise, test cases should be first implemented.

Programmers must first step through their code into their debuggers before checking in their code in the repository. This will save time in the long run when it comes to the overall system debugging and testing. This however considers the fact that the developers should first test their respective modules, and inspect if they work as according to the client’s specifications before uploading their code into the repository.

Programmers must also integration-test their code together with the other developers prior to the checking in. Before the initial upload of the code in GitHub, all of the developers will first have a copy of the code to their system locally. They will be then be able to test the modules they are currently working on if it can be integrated to code itself without any problems. As further uploads and revisions have been pulled to the code repository, the developers will then store the latest code locally for their respective testing.

Programmers will be able to check each other’s code for any further clarifications or critiques regarding the different processes currently being worked on. During the periodical checks, the lead Dev will first inspect the current progress of the code, checking if the coding standards were met. After the lead developer’s approval, a discussion among the programmers will be initiated, further discussing any clarifications or questions regarding the functionality or features of the system.