**Software Construction Strategy**

**Language, Compiler, and Editor**

The team members have decided to use Java and MySQL as the coding language for this Software Solution. Seeing that the problem is mainly on tracking inventory, orders, and sales, we see it fit that a database should be used to solve this problem. Creating the system with Java would allow the system to run on different Operating Systems, and members are more familiar connecting SQL with Java.

The team has decided to use MySQL Workbench for the database of the system. Majority of the members are already adept and familiar with MySQL Workbench and its environment and MySQL Workbench also allows us to create a database easily. As for Java, the team has decided to use Eclipse because it is more user friendly in terms of setting up the user interface. Eclipse also provides plug-ins that can help the members.

**Construction Strategy**

Logic defines the behaviour and rules of the application which is why we think that it is important to implement it first before the user interface.

In order to have a clear view of what needs to be done, the team has decided to define all the functions needed first before proceeding to the main menu. Programmers are to define how functions and classes should behave which would help lessen the complicatedness of the program. Also defining functions first would also let us to work on separate features, then connect them later with the Main Menu.

Programmers could work in pairs or individually, depending on the load of the feature they are working on.

Programmers are discouraged to use non-standard language features as other programmers who are not familiar with the non standard language features will be forced to be familiar with it. This, of course, could be overridden if the whole group is to decide on a non-standard language feature.

**Revision and Version Control**

When adding a new feature or editing a new feature, programmers should fork from the existing branch. They should edit and add new features in the newly created branch, and test it if it works with the system before pushing it to the repository.

Before the branch can be merged to the parent branch, QAs should have tested the branch before merging it to the master branch to see if the system still follows the criteria and is bug-free as much as possible. Members of the group should check on the Github repositories for revisions. Members should also notify other members about revisions in the code through Facebook or text.

**Code Review**

Fellow programmers should consciously and systematically check each other’s code for mistakes and bugs. It is to make sure that the code that gets into the system is well thought of and has gone through the brain of more than one person. This way, refactoring is done on a regular basis, keeping the code neat and more correct. This would also allow the programmers to understand the code and provide feedbacks.

Programmers should inform the author of the code about bugs, violations in the coding standards, and errors so that the author can edit the code.

**Testing and Debugging**

For our project to be as bug free as possible, programmers should test and debug their new features if the feature works with other existing features using a separate test case. The code is then passed to the QA for debugging the feature using the test cases provided by the QAs and the Developers. This should be also true for integration-testing. Programmers are recommended to integration-test their code before pushing to the repository as this can minimize greater problems such as tracing where the problem is residing.

**Error Handling**

In Java, errors are handled by an exception object. Using exceptions will allow the application to recover gracefully from errors and other exceptional events. Another way to recover from errors is by error logging. This captures the data that was being processed when the application crashed. It give the programmers the information necessary to fix them quickly.