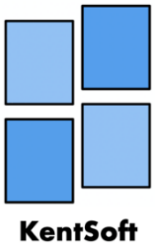
# Code Review




While developing this application at KentSoft, our developers followed the guidelines set in the Developer Guidelines document. We believe that the document was key to keeping a maintainable development environment which has code that members can easily read and understand.

We used Java, which was the language specified, to build the application and programmed in the same coding style specified (camel case naming, etc). However, we ran into a few software problems while using our chosen IDE, IntelliJ. This issue was resolved by our skilled versatile team of developers choosing to comfortably migrate to a different IDE called Eclipse which allowed them to complete the project with ease.

They used different elements of object orientated programming such as encapsulation and inheritance to avoid code duplication and make the code easier to maintain for future development.

We used test driven development to provide a robust application our client. This agile development pattern has allowed us to build applications efficiently but also in a way that the number of bugs are reduced. Our main form of testing was using JUnit to write test classes for our code and making sure they pass the tests.

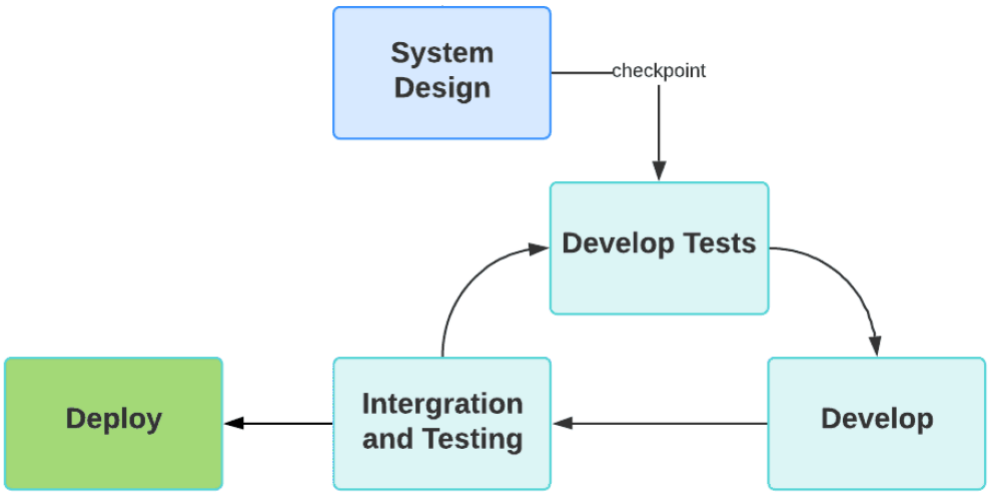
We followed the 6-step process in our Developer Guidelines document on how to implement JUnit testing properly so that we are writing tests which actually help us find bugs.

Try and Catch was imperative in the success of this project and was used very frequently. This ensures error handling so that we know why our application crashes and to also keep running in situations where otherwise it would crash.

Method comments were also used so that each member will understand what the code is doing in the future as humans can forget easily. This ensures that members can always quickly grasp what the code written is doing so that development can happen faster.

We used GitLab's built in feature of Issues to track and fix a wide variety of issues we had over the course of Stage 5. This was very useful as we could delegate and assign different issues to members in out team so that they can be solved. Knowing which issues, we had to solve and who was solving them was imperative to the success of solving many bugs due to the fact that we could constantly check the progress and be reminded of them. Using labels when submitting issues help sort them according to what type of issues they were.

We also decided to divide each stage into milestones on GitLab which helped us clear the backlog of Stage 4. This kept us organised as we knew what tasks we had to do for each stage and not get mixed between which task belongs to which stage.




During the stages between Develop Tests, Develop and Integration and Testing we as a team will carry out reviews and update the UML class diagram as a result of the system design being changed. For example, after the first cycle of integration we review the outcome as a pair. We review our developed tests and change them according to how the application is progressing.

Our design methodology dictates that are workflow is as seamless as possible. Our main repository contains a sub module. This sub module has our Gradle repository inside it which acts as a wrapper for our Java application. The reason we have done is so that we can use pipelines to continuously build and test our additional changes we push or merge to out master branch. This automates our testing and makes our software development cycle more efficient. A docker container is spawned inside our repository which runs the tests and the screenshot below shows information about the status of these tests. These reports are very helpful in continuously finding bugs in our software development cycle as well.