**Code Review Checklist**

**Initial Review**

* Does the project build with no errors?
* Does the code execute as expected?
* Is the code clear and easy to understand?
* Does the implementation match the design?
* Does the implementation cover all the requirements?
* Are there JUnit test cases for the project?
* Do all the test cases pass?
* Are there test cases that are ignored?
* Is there documentation for the project?
* Do you have to manually obtain Jars, xml files and other artifacts that the system requires to run?

**Comments and Coding Conventions**

* Are there any coding conventions? If yes, does the code comply with it?
* Are variable and method names descriptive and do not require comments?
* Do all methods and classes contain Javadoc comments?
* Are method parameters used for input or output clearly identified as such?
* Are complex algorithms and code optimizations adequately commented?
* Are there sections of code that are commented out?
* Are the method, class and variable names chosen clearly representative of their purpose?
* Are there comments for third party libraries usage?
* Are there unnecessary comments such as change history information?
* Are the comments accurate?

**Formatting**

* Is the formatting consistent in the project?
* Are there blank lines to show separate concepts?
* Are there comments between two variables?
* Are concepts that are closely related kept vertically close to each other?
* Are variables the first things you see in the method?
* Are instance variables declared at the top of the class?
* If one method is calling other method, is the caller above the callee?
* Does the program naturally flow?
* Do you have to scroll right to read the code?

**Classes**

* Are there coding conventions? If yes, does the class comply with it?
* Is the name of a class meaningful?
* Are there a lot of public variables?
* Are there unused variables?
* Does the class follow step down rule (private methods that are called from public methods should be right after the public method)?
* Does the class comply with Single Responsibility Principle?
* Does the class comply with the Open Closed Principle?
* Are there methods that are never called?
* Does the class contain dead code?
* Are there magic numbers in the class?
* Are logging levels used correctly?

**Methods**

* Are there methods that are longer than 25 lines?
* Does the method have odd strings, not clear data types and APIs?
* Does the method contain duplicate code?
* Does the method do one thing?
* Are the methods excessively complex ?
* Is the method name descriptive?
* Does the method take more than four arguments?
* Does the method take a boolean value as a parameter? If yes, it might be a hint that this method does more than one thing.
* Does the method contain code that can be replaced by using an existing API?

**Error Handling**

* Does the code throw an exception that is specific or generic?
* Does the code use enum or interface to maintain the error codes?
* Does the code return error codes? If yes, are the error messages informative and passed along with exceptions?
* Is there a wrapper class for exception handling if the application is using a third party library?
* Do the methods or objects return null?
* Are assertions used everywhere data is expected to have a valid value or range?
* Are errors including thrown exceptions properly handled?
* Are resources released in all error paths?
* Is the method caller notified when an error is detected?
* Does the code handle possible null returns from a third party libraries?
* Does the project have test cases which cover all possible errors that can occur?

**Boundaries**

* Are there test cases for each boundary?
* Does the code at boundaries have clear separation so that future changes will be easily?
* Are there test cases for third party library usage?
* Is there any code that passes Maps or any other interface at a boundary?

**Resource Leaks and Thread Safeness**

* Is allocated memory deallocated?
* Are all objects (Database connections, Sockets, Files, etc.) freed even when errors occur?
* Are all global variables thread-safe?
* Are objects accessed by multiple threads and are they coded correctly?
* Are locks released in the same order they are obtained?
* Are there any possible deadlocks?

**Performance**

* Are objects duplicated when only references are needed?
* Can the code be optimized?
* Has the code been profiled?
* Does the code make efficient use of memory, CPU cycles, bandwidth or other system resources?