### **Detailed Code Review Checklist**

#### **1. Code formatting**

While going through the code, check the code formatting to improve readability and ensure that there are no blockers.

* Use alignments and proper white space. Also ensure that code block starting point and ending point are easily identifiable.
* Ensure that proper naming conventions (Pascal, CamelCase etc.) have been followed.
* Remove the commented code as this is always a blocker, while going through the code.

#### **2. Architecture**

* The code should follow the defined architecture.
* Split into respective files (Action, Pages, Conf and Tests).
* Use appropriate Design patterns.

#### **3. Coding best practices**

* No hard coding, use constants/configuration values.
* Do write comments for what you are doing and why you are doing.
* Avoid multiple nested loops and multiple if/else blocks.
* Use framework features, wherever possible instead of writing custom code.

#### 

#### 

#### 

#### **4. Non Functional requirements**

* **Maintainability** – The application should require the least amount of effort to support in near future. It should be easy to identify and fix a defect.
* **Readability:** Code should be self-explanatory. *Get a feel of story reading, while going through the code*. Use appropriate name for variables, functions and classes.
* **Debuggability:** Provide support to log the flow of control, parameter data and exception details to find the root cause easily.
* **Configurability:** Keep the configurable values in place (conf file) so that no code changes are required, if the data is changed frequently.
* **Reusability:** Consider reusable services, functions, components and classes.
* **Extensibility:** Easy to add enhancements with minimal changes to the existing code. One component should be easily replaceable by a better component.

#### **5. Object-Oriented Analysis and Design (OOAD) Principles**

* **Single Responsibility Principle (SRS):** Do not place more than one responsibility into a single class or function, refactor into separate classes and functions.
* **Open Closed Principle:** While adding new functionality, existing code should not be modified. New functionality should be written in new classes and functions.