**Code Review Checklist**

1. **Functionality**

- Does the application work as expected?

- Does the new feature add value? Or is it scope creep?

- Will it impact existing Functionality?

- Is it going to create inconsistency in other places?

1. **Readability, Code syntax and Formatting**

- Is the code clear and concise?

- Does it comply to Meriplex’s coding guidelines?

- Are all language and project conventions followed?

- Are identifiers given meaningful and style guide-compliant names?

- Ensure proper naming conventions are followed (Pascal, CamelCase, etc.)

- Ensure code is properly indented and follows the same rule (tab/two space)

1. **Design Principle**

- Is the code properly planned and designed?

- Is code in sync with existing code patterns/technology?

- Did we think about reusability?

- Is the code well organized in terms of placement of components?

1. **Patterns, idioms and best practices**

- No hard coding, use constants/configuration values.

- Does the code keep with the idioms and code patterns of the language?

- Does the code make use of the language features and standard libraries?

1. **Documentation and maintainability**

- Is the code self-documenting or well-documented?

- Did you add Comments mentioning reason of change, todo, workarounds in the code?

- Is the code free of obfuscation and unnecessary complexity?

- Is the control flow and component relationship clear to understand?

1. **Debuggability, Testability and Configurability**

- Are we logging exceptions, flow of control, user behavior for better debugging and consumer behavior understanding?

- Is code testable?

- Is code configurable enough, to avoid changes in business or view layers or even code changes?

1. **Performance, reliability and scalability**

- Is the code optimized for in terms of time and space complexity?

- Does it scale as per the need?

- Does it cover failure scenarios?

- Does it have instrumentation like reporting for metrics and alerting for failures?

1. **Security**

- Is the code free of implementation bugs that could be exploited?

- Have all the new dependencies been audited for vulnerabilities?

- Does it have Authentication, authorization, input data validation against security threats?

1. **Etiquettes**

- Is the PR atomic?

- Does the PR follow the single concern principle?

- Are the commit messages well-written?

1. **Notice What’s Missing**

- Did you tried using app/functionality as end user?

- Does it cover loading, error handling, edge cases and unexpected input handling?

- Will it work in all support environment OS, browsers, platforms etc?

- Does it need feature flag control?

- Does it have proper instrumentation?