Code Review Checklist

# Requirements & Stakeholder Approval

This section ensures that the code meets all the specified requirements and has been approved by stakeholders.

Have the requirements been met?

Have stakeholder(s) approved the change?

Is the code’s behavior consistent with the project’s specifications?

# Code Formatting & Readability

This section focuses on the readability and formatting of the code, ensuring that it adheres to standard conventions.

Is the code formatted correctly?

Unnecessary whitespace removed?

Is the code well-organized and easy to read?

Are naming conventions consistent and descriptive?

Is the code properly indented and formatted?

# Error Handling & Logging

This section verifies that the code has proper error handling and logging mechanisms for easier debugging and troubleshooting.

Are different errors handled correctly?

Are edge cases and potential error scenarios handled appropriately?

Are errors and warnings logged?

Are exceptions used appropriately and caught at the correct level?

Is logging implemented for debugging and troubleshooting purposes?

Are error messages clear, descriptive, and actionable?

# Code Design & Architecture

This section evaluates the overall design and architecture of the code, ensuring it is modular, maintainable, and follows best practices.

Does the code follow established design patterns and architectural guidelines?

Are separations of concerns followed?

Is the code modular and maintainable?

Are functions and classes of reasonable size and complexity?

Does the code adhere to the principles of separation of concerns and single responsibility?

# Performance & Efficiency

This section assesses the performance and efficiency aspects of the code, looking for bottlenecks and opportunities for optimization.

Is the code performance acceptable?

Are there any potential performance bottlenecks or inefficiencies?

Is memory usage optimized?

Are algorithms and data structures appropriate and efficient?

Are there any opportunities for caching or parallelization?

# Security

This section verifies that the code adheres to secure coding practices and doesn't introduce any security vulnerabilities.

Does the code follow secure coding practices?

Are there any potential security vulnerabilities?

Is user input validated and sanitized properly?

Are authentication and authorization mechanisms implemented correctly?

# Testing

This section ensures that appropriate tests are in place and that they cover the essential functionality and edge cases.

Does the code include appropriate unit tests or integration tests?

Is the test coverage sufficient for the critical functionality and edge cases?

Are the tests passing and up-to-date?

Is the test code well-structured, readable, and maintainable?

# Documentation

This section checks for adequate documentation that explains the code's functionality and complex segments.

Is there sufficient documentation?

Are inline comments used effectively to explain complex or non-obvious code segments?

Do functions, methods, and classes have descriptive comments or docstrings?

Is the ReadMe.md file up to date?

Is there high-level documentation for complex modules or components?

# Code Reuse & Dependencies

This section reviews how well the code reuses existing libraries and manages dependencies.

Is the code properly reusing existing libraries, frameworks, or components?

Are dependencies managed correctly and up-to-date?

Are any unnecessary dependencies or duplicate code segments removed?

Are dependencies secure, actively maintained, and of sufficient quality?

# Compliance & Standards

This section ensures that the code aligns with company or project-specific coding standards and guidelines.

Does the code comply with company or project-specific coding standards and guidelines?

Are any linters or static analysis tools used to enforce coding standards?

# Other

This section covers other considerations that didn't fit into the above categories but are still important for code quality.

Magic values avoided?

No unnecessary comments?

Minimal nesting used?

Is the code not repeated (DRY Principle)?

Relevant Parameters are configurable?

Feature switched if necessary?

Has the release been annotated (GA etc)?

Are inputs sanitized?

# Resources

1. [Ultimate 10-Step Code Review Checklist - Swimm](https://swimm.io/learn/code-reviews/ultimate-10-step-code-review-checklist#acomprehensive10stepcodereviewchecklist)
2. [What You Need in a Code Review Checklist | Dev Interrupted Powered by LinearB](https://linearb.io/blog/code-review-checklist)
3. [Boost Your Code Review Process with This Comprehensive Checklist (evoketechnologies.com)](https://www.evoketechnologies.com/blog/code-review-checklist-perform-effective-code-reviews/)
4. [Code review checklist: 7 steps to level up your review process (pluralsight.com)](https://www.pluralsight.com/blog/software-development/code-review-checklist)
5. [Code Review Checklist](https://www.codereviewchecklist.com/)
6. [Examples of Code Review Checklists and Guides | by Andrei Gridnev | Medium](https://andreigridnev.medium.com/examples-of-code-review-checklists-and-guides-2dfed082a86d)