Code Review Template for Productivity Study

Note: This is a sharable template only. You will be supplied with a choice based form for Code Reviews internally so that you may choose to rate or ignore the section if it is not relevant.

Code Bank

- The Code Bank helped me identify the correct code-snippet
- I found the underly code-flow and control flow of the feature stories with the searched code-snippet

Implementation

- Does this code change do what it is supposed to do?
- Can this solution be simplified?
- Does this change add unwanted compile-time or run-time dependencies?
- Was a framework, API, library, service used that should not be used?
- Was a framework, API, library, service not used that could improve the solution?
- Is the code at the right abstraction level?
- Is the code modular enough?
- Would you have solved the problem in a different way that is substantially better in terms of the code's maintainability, readability, performance, security?
- Does similar functionality already exist in the codebase? If so, why isn't this functionality reused?
- Are there any best practices, design patterns, or language-specific patterns that could substantially improve this code?
- Does this code follow Object-Oriented Analysis and Design Principles, like the Single Responsibility Principle, Open-close principle, Liskov Substitution Principle, Interface Segregation, Dependency Injection?

Logic Errors and Bugs

- Can you think of any use case in which the code does not behave as intended?
- Can you think of any inputs or external events that could break the code?

Error Handling and Logging

- Is error handling done the correct way?
- Should any logging or debugging information be added or removed?
- Are error messages user-friendly?
- Are there enough log events and are they written in a way that allows for easy debugging?

Usability and Accessibility

- Is the proposed solution well designed from a usability perspective?
- Is the API well documented?
- Is the proposed solution (UI) accessible?
- Is the API/UI intuitive to use?

Dependencies

- If this change requires updates outside of the code, like updating the documentation, configuration, readme files, was this done?
- Might this change have any ramifications for other parts of the system, backward compatibility?

Security

- Does this code open the software for security vulnerabilities?
- Are authorization and authentication handled in the right way?
- Is sensitive data like user data, credit card information securely handled and stored? Is the right encryption used?
- Does this code change reveal some secret information (like keys, usernames, etc.)?
- If code deals with user input, does it address security vulnerabilities such as cross-site scripting, SQL injection, does it do input sanitization and validation?
- Is data retrieved from external APIs or libraries checked accordingly?

Performance

- Do you think this code change will impact system performance in a negative way?
- Do you see any potential to improve the performance of the code?

Readability

- Was the code easy to understand?
- Which parts were confusing to you and why?
- Can the readability of the code be improved by smaller methods?
- Can the readability of the code be improved by different function/method or variable names?
- Is the code located in the right file/folder/package?
- Do you think certain methods should be restructured to have a more intuitive control flow?
- Is the data flow understandable?
- Are there redundant comments?
- Could some comments convey the message better?
- Would more comments make the code more understandable?

- Could some comments be removed by making the code itself more readable?
- Is there any commented out code?