# Effective Strategies for Reviewing Pull Requests

Having covered how to create efficient Pull Requests, this chapter shifts focus to the art and science of reviewing them. A well-executed review process is vital for maintaining code quality and fostering a positive team dynamic. The following are key strategies and points to consider for effective PR reviews:

# Commenting Efficiently

## Politeness and Professionalism in Commenting

Effective communication in PR reviews is not just about what is said but also how it's conveyed. Maintaining a tone of professionalism and courtesy in comments is essential, irrespective of how well team members know each other. Balancing professionalism with a friendly tone contributes to a positive and productive work environment.

While comments should be professional, they needn't be overly formal. The goal is to create an atmosphere of mutual respect. Avoid using profanity or disrespectful language. A constructive, polite tone goes a long way in fostering collaborative and respectful interactions.

## Focus on the Code, Not the Developer

When providing feedback, ensure that **your comments are directed at the code and not the developer who wrote it**. This distinction is crucial, especially when pointing out areas that need improvement or when discussing contentious topics. By focusing on the code, you

keep the conversation objective and centered on problem-solving. Let's illustrate this with an example:

**Less Effective**: “Why did you use threads here when there’s obviously no benefit to be gained from concurrency?”

**More Constructive**: “I noticed the implementation uses a concurrency model. It appears that this adds complexity without yielding clear performance benefits. Could we explore a simpler, single-threaded approach here, as it might streamline the process without impacting performance?”

By using the more constructive approach, you not only pinpoint the issue but also provide a clear rationale and a potential solution. This method encourages a dialogue focused on improving the code and finding the best solution, rather than placing blame or making personal judgments.

## Constructive Feedback

When you disagree with a piece of code, it's important to not only state your case **but also provide an alternative solution if possible.** Explaining why you disagree and suggesting a better approach can be immensely helpful. Additionally, supplement your comments with links to internal or external resources when relevant, as this can provide further context or support for your feedback.

## Comment Formatting

Leverage the platform's formatting capabilities to make your comments clearer and more impactful. For instance, on GitHub, you can format code suggestions using Markdown syntax, enclosing code snippets within triple backticks. This approach enhances the readability and comprehension of your feedback.

## Prompt Response and Strategic Review Management

Some collaboration tools have features to automate the merging of PRs based on specific criteria, such as receiving a certain number of approvals. However, this can lead to situations where a PR is automatically merged while it's still under review, potentially bypassing crucial feedback. To manage this effectively:

- **Preventing Premature Merges**: Implement additional rules to control the automatic merging of PRs. For example, you could set a rule that a PR can only be completed once all comments are resolved. This gives reviewers a chance to flag that they are still reviewing the PR. Simply adding a comment like “I’m reviewing this now” can serve as a marker that the PR is under active review. Once your review is complete, resolve that comment to signal that it’s ready for the next steps.
- **Communicating Availability and Timeframes**: If you’re unable to conduct a full review immediately upon receiving a PR, it’s helpful to acknowledge its receipt and communicate your availability. A quick response stating when you plan to review it keeps the PR author informed and manages expectations. For example, a simple message like, “Thanks for the PR, I’ll be able to review it by tomorrow afternoon,” can be very effective.
- **Recommending Alternate Reviewers**: If your schedule doesn’t permit a timely review, suggest alternative reviewers who might have the bandwidth to respond more quickly. This approach helps in maintaining the momentum of the PR process and ensures that reviews are not bottlenecked.

## Minimizing Nitpicking with Automation

In code reviews, dwelling on minor details can slow down the process and distract from significant issues. Integrating automation into your workflow can be transformative to overcome this issue. It ensures