

Find the detailed version of this roadmap along with other similar roadmaps

roadmap.sh

Code Reviews

Team Wide Practices

- Document and standardize the code review process.
- Ensure that the purpose of code reviews is clear to everyone.
- Ensure that "Definition of Done" is documented and clear to everyone
- Encourage team members to participate in code reviews.
- Define a process for conflict resolution in code reviews.
- Have a definitive style guide for style preferences.
- Use automation to speed up the code reviews (linting, sniffing etc)
- Set clear expectations for code review turnaround times.
- Provide adequate time for code reviews and ensure that it is a priority.
- Use code reviews as an opportunity for knowledge sharing and learning.
- Encourage reviewing code in unknown-areas for cross-functional knowledge.
- Constantly monitor and improve code review process.
- Recognition and rewards for those with track record of quality feedback.
- Encourage communication/collaboration; avoid treating code reviews as a one-way process.
- Hold regular code review sessions to discuss broader trends or issues that arise during the review process.
- Encourage authors to seek feedback during development before submitting a formal code review.Retry

Development (Author)

- Follow the coding standards and any other team guidelines.
- Stay consistent with the overall project design and architecture.
- Write a failing test if the change is for a bug fix.
- Break down complex tasks into smaller easily manageable PRs.
- Consider the impact of the change on other parts of the system.
- Take notes on any questions or concerns about the change to discuss them during the review.
- Write the automated tests.
- Write the documentation for the feature or changes if required.
- Update any documentation that may have made obsolete through the changes.

Post Development (Author)

- Review your code before submitting for review.
- Ensure that the changes are complete and ready for review, including all necessary tests and documentation.
- Verify that the code change has been properly tested in a development environment.
- Double-check that the code adheres to the project's coding standards and best practices.
- Identify any potential performance, security, or scalability concerns and note them for discussion during the review.
- Make sure to add proper title, description, any screenshots, relevant links, configuration changes etc in the PR.
- Approach the review process with an open mind, and be willing to learn from and collaborate with other team members.

Before Review (Reviewer)

- Understand the requirements and the context in which change was made.
- Based on the requirements, prepare a list of items that should have been covered in the changes.
- Ensure that you understand the codebase and its architecture.
- Review any documentation or design specifications related to the change.
- Make list of any potential risks or issues that could arise with the change.
- Approach the process with an open mind, be willing to provide constructive feedback and collaborate to improve code quality
- Consider the overall quality of the code, including readability, maintainability, and scalability.
- Determine the appropriate level of review needed based on the scope and impact of the code change.
- Be willing to collaborate with the author to resolve any issues or concerns that arise during the review process.Retry

During Review (Reviewer)

- Be respectful and professional in your feedback, avoiding personal attacks or derogatory comments.
- Provide clear and actionable feedback, including specific suggestions for improvement and explanations of any concerns.
- Identify any potential performance, security, or scalability concerns, and discuss them with the author.
- Prioritize your feedback, focusing on the most important issues first.
- Review any tests included with the code change to verify that they adequately cover the functionality and edge cases.
- Ensure that the code change adheres to the project's coding standards and best practices.
- Ensure that the relevant documentation has been updated.
- Team wide styleguide is the absolute authority styling. Verify changes against those instead of personal preferences
- Leave comments to suggest improvements, but prefix it with "Nit" if it's not critical to meeting the standards
- Seek continuous improvement, not perfection.
- Keep the short-term and long-term considerations in mind.
- Consider using pair programming as an alternative or supplement to code reviews.
- Provide positive feedback in addition to constructive criticism, to reinforce good practices and boost team morale.

After Review (Author)

- Address all the feedback received, including any concerns or questions raised.
- Implement the suggested changes and provide explanations where needed.
- Run the tests and ensure that they all pass after making changes
- Update any documentation or code comments affected by the changes.
- Seek feedback from other team members if you are unsure about the changes.
- Submit the updated code for a second review if needed.

After Review (Reviewer)

- Resolve conflicting opinions in a timely manner; don't let a PR sit around due to disagreement.
- Verify that all the feedback has been addressed by the author.
- Review the updated code and ensure that the suggested changes have been implemented as expected.
- Run the tests again and ensure that they all pass.
- Address any questions or concerns that the author may have.
- Be open to feedback from the author and be willing to make adjustments to your feedback if necessary.Retry

After Approval (Both)

Software Design

Engineering Manager

Continue Learning with following relevant tracks