Zeal Internship 2024 - Clean Code, PRs, and Unit Tests (Oh My!) (2024-06-17 09_01 GMT-5)

Speakers discussed software engineering best practices, emphasizing clean code, following established rules and guidelines, and structured approaches to coding. They also shared insights on writing clean code, avoiding code smells, and improving the quality of pull requests. Additionally, they covered the importance of testing and quality assurance in software development, including unit testing, integration testing, performance testing, UI testing, test automation, and writing tests as code. Overall, the speakers stressed the need for a comprehensive testing strategy to ensure high-quality software.

Transcript

https://otter.ai/u/dYHGkX1p2ZDqnw1UhIMTymLjLrw?view=transcript

Action Items

- [ ] Interns should explore Zeal's Centers of Excellence Slack channels for technical resources and questions.
- [ ] Dylan and Scott are available as resources for interns to reach out to for guidance.
- [ ] Interns should apply principles of clean code like structure, organization, and documentation in their own work.

Outline

Software engineering and development with industry professionals.

- Jimmy and Gavin are curious about how Scott and Dylan write their code.
- Yun is relatively new to coding and has been told their code is massive.
- Speaker 5 shared his experience as a solutions architect and musician, mentioning his side hustle as a woodworker.
- Speaker 5 and Speaker 4 discussed their roles and interests, with Speaker 5 expressing gratitude for the internship opportunity.

Clean code in software engineering with various perspectives and resources shared.

- Speakers discuss the subjective and loosely defined concept of "clean code" in software engineering.

- Speaker 5 discusses the differences between software and hardware engineering, with a focus on the evolving nature of the field and the role of AI.
- Speaker 5 highlights the resources available within Zeal, including a Slack channel with centers of excellence for various technical topics.

Software development principles and code organization.

- Speaker 5 discusses the importance of readable, maintainable, and scalable code.
- Speaker 5 criticizes excessive commenting in code, suggesting it's better for source control to manage comments.
- Speaker 5 criticizes a code snippet for being too long and messy, violating the single responsibility principle.
- Speaker 5 shows a cleaner example of code that is easier to understand and maintain, with better indentation and error handling.

C# programming, code quality, and best practices.

- Speaker 4 highlights issues with C# code organization and naming conventions.
- Speakers discuss the benefits of using C++ over Java for programming, with one speaker mentioning its ability to understand memory management.

Software development principles and best practices, including single responsibility, interface segregation, and structured file organization.

- Speakers discuss software development principles, including single responsibility and interface segregation.
- Speakers discuss importance of file organization and structure in software development.

Coding practices, code smells, and deadlines.

- Speaker 5 shares experience and best practices for recognizing clean code.
- Speaker 4: Code can be unreadable, but compiler helps (12 words)
- Speaker 5: Good programmers can still produce bad code under pressure (13 words)

Code smells, responsibilities, and writing clean code.

- Speaker 4 mentions code smells, a term used to describe something that doesn't look right in the code, but doesn't know the origin of the phrase.
- Speaker 5 discusses how code smells can be propagated through code reuse and generational differences in software development.
- Speaker 5 suggests identifying and fixing issues on the spot, but also considering the impact and size of the issue before doing so.
- Speaker 5 and Speaker 4 discuss the importance of trust and boundaries when deciding whether to fix issues outside of the sprint.
- Speaker 5 emphasizes the importance of writing clean code as a developer responsibility, not just copying and pasting code.

- Speaker 5 advises caution when using generative AI tools, as they can lead to unintentional code copying and pasting.

Technical debt, code quality, and trust in software development.

- Speaker 5 advises developers to be mindful of temporary code solutions that can find their way into production, causing inefficiencies or harm.
- Speaker 5 encourages developers to avoid taking shortcuts and writing quick fixes that may not be the best code, leading to long-term problems.
- Speaker 4 explains technical debt to non-technical manager: "It's like borrowing money to pay for a feature now, but it will cost more later."
- Speaker 5 emphasizes importance of building trust with clients: "If we can develop trust, conversations about technical debt will be more palatable."

Effective code review process, including characteristics of good PRs.

- Speaker 5 emphasizes the importance of a system of checks and balances in code reviews, citing the need for both industry and organizational standards.
- Speaker 5 highlights the benefits of a consistent code review process, including an audit trail of changes and opportunities for professional growth.
- Speaker 5 discusses their pet peeve of divisive PRs that can bully and their efforts to build trust with a challenging client.
- Speaker 5 highlights the importance of small, focused PRs that follow predefined standards and conventions.

Best practices for code reviews and communication between team members.

- Speaker 5 emphasizes the importance of maintaining an end-to-end audit trail for code changes, linking them to user stories or bug fixes.
- Speaker 5 advocates for constructive and objective peer reviews, consistent across all developers and code commits, and respectful communication.
- Speaker 5 advocates for a more professional approach to PR reviews, emphasizing the importance of clear communication and avoiding unnecessary delays.
- Speaker 5 advises having a pre-code change conversation to avoid confusion and save time.
- Dylan shares an example of copy-paste code changes and their potential consequences.

New JPEG service, code review and suggestions.

- Developers discuss new JPEG service based on existing PDF service, with logging differences noted.
- Speakers discuss inconsistent naming conventions in code, leading to confusion.

Testing and merging code changes in a PR.

- Speaker 5 advocates for using constants instead of magic numbers in code.

- Speaker 5 explains test automation and unit testing to the group.

Unit testing, integration testing, and TDD in software development.

- Speaker 5 discusses different types of tests, including unit, integration, performance, stress, and load testing.
- Speaker 5 emphasizes the importance of testing for security and connectivity issues in integration tests.
- Speaker 5 discusses integration testing, using dapper and Azure Blob storage.
- Speaker 5 highlights the importance of comprehensive testing, including slower-running tests.
- Speaker 5 explains the difference between facts and theories, emphasizing the importance of understanding client's approach and goals.
- Speaker 5 advocates for writing test automation code as it's being developed, rather than doing it after the fact.

Test automation, code responsibility, and building testable code.

- Developers should write tests for their own code to ensure functionality and catch defects early (Speaker 6)
- Test automation can save time and effort in the long run, but requires initial setup and maintenance (Speaker 5)
- Speaker 4 and 5 discussed the challenges of test automation, including lack of documentation and integration with build pipeline.
- Speaker 5 shared their experience using chat GPG to generate boilerplate code for test automation, saving time and increasing efficiency.

PR review process, code coverage, and test quality.

- Speaker 5 discusses code review times and team availability.
- Speaker 5 advises against being single-threaded and suggests breaking tasks into smaller PRs for faster review and feedback.
- Speaker 5 and Speaker 1 discuss the benefits of smaller, more frequent releases in a codebase.
- Speaker 1 suggests creating tests for deliverable features, while Speaker 5 prefers quality over quantity.
- Speakers discuss the importance of test quality and code coverage, rather than just quantity.

Writing unit tests for software development, including edge cases and code coverage.

- Speaker 4: Edge cases require structure and testing to avoid issues in production.
- Speaker 5: Focused code with singular purpose makes testing easier and more effective.
- Speaker 4 highlights the cost of maintaining software, not just developing it.
- Speaker 5 emphasizes the importance of writing tests and having high code coverage.