Onboarding Guide for New Software Engineers

Your Team Name

November 3, 2024

Contents

1 Introduction to the Team

1.1 Team Purpose and Mission

Welcome to the team! Our primary mission is to create innovative, user-centered software solutions that solve critical problems within the [industry/domain] space. We are dedicated to developing high-quality applications that are both scalable and maintainable. Our team's values emphasize collaboration, continuous improvement, and customer focus. We aim to not only meet but exceed user expectations through thoughtful, data-driven design and rigorous testing.

As part of our team, your contributions will play a key role in helping us achieve our goals. We encourage everyone to bring fresh ideas and explore new technologies that can enhance our products. Each team member is empowered to influence the technical direction of our projects and contribute to decision-making processes. We believe in an open-door policy and value transparent communication. If you have ideas or feedback, please feel free to share them. We value each team member's voice and encourage everyone to be proactive in sharing insights, identifying opportunities, and suggesting improvements.

1.2 Team Structure

The team is composed of various roles, each designed to support different aspects of the product development lifecycle. Here's a breakdown of the primary team members and their responsibilities:

- **Team Lead** – [Name]: The team lead acts as the primary coordinator for project activities and oversees all team functions. This person is responsible for setting priorities, coordinating work across departments, and ensuring alignment with broader company objectives. They also facilitate team meetings, project planning sessions, and ensure that the team's work aligns with the organizational roadmap.

- **Product Manager** – [Name]: Our product manager (PM) is responsible for defining product requirements, translating user needs into technical specifications, and setting project milestones. They interact frequently with stakeholders and users to gather feedback, which helps in refining the product roadmap. The PM also works closely with you to provide clarity on the purpose and expected outcome of each feature.

- **Engineering Manager** – [Name]: The engineering manager (EM) is focused on the professional development of team members, conducting performance reviews, and setting development goals. They provide regular feedback, assist with problem-solving, and are available for one-on-one sessions to discuss your career path and address any challenges you face. EMs also handle resource allocation, ensuring each project has the right people with the right skills.

- **Tech Leads** – [Names]: Each tech lead specializes in a different technical domain such as front-end, back-end, or DevOps. Tech leads ensure code quality, provide technical direction, and offer guidance on best practices. They also play a crucial role in mentoring team members, conducting code reviews, and helping solve complex technical problems.

2 Tech Stack Overview

2.1 Core Technologies

Our tech stack is chosen to balance performance, scalability, and ease of development, and is based on the evolving needs of the application. Here is a breakdown of the primary tools and frameworks that we utilize in our projects:

- **Frontend**: We use [React/Angular/etc.], which is ideal for creating dynamic and responsive interfaces. React's component-based architecture allows us to build modular UI elements that are easy to test, maintain, and reuse. Additionally, we use state management libraries such as Redux or MobX to handle complex application states effectively. Styling is handled using CSS-in-JS libraries like Styled Components, which helps us maintain consistency across the UI.

- **Backend**: Our backend services are powered by [Node.js/Python/Java/etc.], offering a fast, scalable, and maintainable environment. By using RESTful APIs and, in some cases, GraphQL, we enable efficient client-server communication. This setup allows our frontend applications to retrieve precisely the data they need, reducing over-fetching and improving performance.

- **Databases**: We use a combination of [PostgreSQL, MongoDB, etc.] to manage data, depending on whether we need relational or document-based storage. PostgreSQL is our choice for structured data, offering ACID compliance and complex queries. MongoDB, on the other hand, is used for flexible, schema-less data storage, which is particularly useful for rapidly evolving applications.

- **Infrastructure**: The foundation of our infrastructure is [AWS, GCP, Azure], which offers a suite of services for hosting, scaling, and managing our applications. We leverage Docker for containerization, ensuring consistency across development, testing, and production environments. Kubernetes is used for orchestrating and managing our containers, allowing us to scale seamlessly as demand fluctuates.

2.2 Development Tools

To maintain consistency and enhance productivity, we use a set of standardized tools across the team:

- **Version Control**: We use Git for version control, with all repositories hosted on GitHub/GitLab. Version control is central to our workflow, as it allows us to collaborate efficiently, track changes, and revert code if necessary. It's essential to follow our commit message guidelines, which are based on conventional commits, to ensure clarity and traceability in our code history.

- **CI/CD**: Our CI/CD pipeline is powered by Jenkins and GitHub Actions. Every code push triggers a set of automated tests, including unit tests, lint checks, and security scans. Once tests pass, code is automatically deployed to staging, where it undergoes further testing before reaching production. This process allows us to catch issues early, streamline releases, and ensure our deployments are reliable.

- **IDE**: We encourage using VSCode or IntelliJ, as these IDEs offer rich plugins and debugging tools that integrate well with our tech stack. You can customize your IDE with specific plugins, such as Prettier for code formatting, ESLint for syntax checking, and GitLens for Git history visualization, to streamline your development workflow.

3 Code Ownership and File Structure

3.1 Directory Organization

Our codebase is structured to enhance readability, maintainability, and modularity. Here is an overview of our primary directories and their contents:

- `src/`: This directory contains all source code files, organized into modules that reflect different features or functionalities. By grouping related components together, we make it easier for developers to find, edit, and test code within specific parts of the application.
- `tests/`: All unit tests, integration tests, and other testing files are located here, structured to mirror the organization of `src/`. Each module or feature has corresponding tests to ensure comprehensive test coverage. Test files follow a naming convention that associates them with their respective components or modules.
- `docs/`: This folder stores technical documentation, including API documentation, design diagrams, and other resources that describe how the codebase operates. Maintaining up-to-date documentation in this directory is crucial for onboarding and future reference.
- `config/`: Configuration files such as environment variables, deployment scripts, and any other settings required for different environments are stored here. Configurations are separated for different stages (development, staging, production) to facilitate smooth transitions across environments.

3.2 Ownership by Module

To ensure accountability and maintain the integrity of each part of the codebase, we assign module ownership:

- **Frontend Module** – [Frontend Lead's Name]: Responsible for all client-side code, including UI/UX implementation, accessibility, and responsiveness.
- **Backend Module** – [Backend Lead's Name]: Oversees server-side code, database management, and API development.
- **Infrastructure** – [DevOps Lead's Name]: Manages CI/CD, container orchestration, and system monitoring, ensuring that our services are reliable and scalable.

4 How to Ask for Help

4.1 Mentorship and Support

To facilitate a smooth onboarding, you are paired with an onboarding buddy—a mentor who is familiar with both the codebase and team dynamics. Your mentor is here to help you acclimate to our workflow, provide guidance on technical challenges, and introduce you to team processes. Don't hesitate to reach out to them with any questions or for code review assistance, especially in your first few weeks. Regular check-ins with your mentor will also give you opportunities to discuss your progress and clarify any doubts.

4.2 Points of Contact for Specific Areas

Depending on your queries, here's who to reach out to:

- **Technical Issues**: [Tech Lead's Name] can assist you with codebase navigation, debugging, and any technical challenges you face. Whether you need advice on implementing features or guidance on technical best practices, they are your primary resource.
- **Product Queries**: For product-related questions, such as feature requirements or user feedback, [Product Manager's Name] will provide detailed insights. They can help clarify product goals and prioritize tasks effectively.
- **Development Practices**: If you have questions regarding coding standards, testing strategies, or process improvements, [Engineering Manager's Name] is available to guide you.

5 Development Workflow

5.1 Git Workflow

Our team follows a Git workflow that promotes collaboration, accountability, and clean code management. Here is an outline of our branch structure:

- **Feature Branches**: Created for new features. Naming convention: `feature/your-feature-name` Each feature branch is dedicated to a single feature to minimize merge conflicts and ensure clarity.
- **Bugfix Branches**: Created to address specific issues, with the naming convention `bugfix/issue-description`. Each bugfix branch includes a reference to the issue it resolves.
- **Release Branches**: Used to prepare for deployments. Naming convention: `release/version-nu` Release branches are created as code stabilizes for deployment.

5.2 CI/CD Process

Our CI/CD process is automated to streamline the path from development to deployment. Each commit goes through rigorous testing, including unit tests, integration tests, and code style checks. The pipeline deploys successful code to the staging environment, where it undergoes further testing before moving to production. This ensures that only stable, tested code is deployed, reducing the chance of bugs in production.

6 Coding Standards

6.1 Language Guidelines

Each language we use follows specific guidelines to ensure code quality and maintainability:

- **JavaScript**: We follow the Airbnb style guide and enforce syntax rules with ESLint, ensuring readability and consistency across the codebase.
- **Python**: All Python code adheres to PEP 8, with Black used for consistent formatting.
- **Java**: Code follows Oracle's standards, emphasizing naming conventions, and modular design.

7 Testing Guidelines

7.1 Unit Tests

Unit tests are written for all new functions and use [Jest/Pytest/etc.]. They should cover key scenarios and edge cases to catch errors early.

8 Documentation and Knowledge Sharing

8.1 Confluence/Google Docs

Our documentation repository includes feature docs, design decisions, and user guides. Please review these documents to gain insights into ongoing projects.

Appendix

Glossary and common terms used within the team.