**Code Quality Metrics Document**

**Project: Online Shop MVP**

**Date: July 2024**

**1. Code Quality Metrics**

**1.1 Code Coverage**

* **Goal:** Ensure at least 85% code coverage.
* **Description:** Code coverage measures the percentage of code that is tested by automated tests. High code coverage helps in identifying untested parts of the codebase.
* **Implementation:**
  + Use testing frameworks such as PyTest for Python or JUnit for Java.
  + Integrate coverage tools such as Coverage.py for Python or JaCoCo for Java.
  + Generate coverage reports and include them in the CI/CD pipeline.
* **Current Status:** 85% coverage achieved.

**1.2 Code Reviews**

* **Goal:** 100% of code changes are reviewed by at least one other developer.
* **Description:** Code reviews are essential for maintaining code quality, ensuring adherence to coding standards, and facilitating knowledge sharing among team members.
* **Implementation:**
  + Use pull requests (PRs) for all code changes.
  + Implement mandatory code review policies in version control systems like GitHub or GitLab.
  + Reviewers should check for code readability, adherence to coding standards, and potential bugs.
* **Current Status:** All code changes undergo a code review process.

**1.3 Static Code Analysis**

* **Goal:** Resolve 95% of issues identified by static code analysis tools before merging.
* **Description:** Static code analysis tools help identify potential issues in the code, such as bugs, security vulnerabilities, and code smells, without executing the code.
* **Implementation:**
  + Integrate static code analysis tools such as SonarQube or ESLint into the CI/CD pipeline.
  + Configure tools to check for coding standards, security vulnerabilities, and code smells.
  + Review and resolve identified issues as part of the code review process.
* **Current Status:** 95% of identified issues are resolved before merging.

**2. Test Coverage**

**2.1 Unit Tests**

* **Goal:** Achieve high coverage of individual units of code (functions, methods, classes).
* **Description:** Unit tests validate the functionality of specific units of code in isolation.
* **Implementation:**
  + Write unit tests for all new and modified code.
  + Use mocking frameworks to isolate units of code.
  + Ensure that tests cover various scenarios, including edge cases.
* **Current Status:** Comprehensive unit tests in place with 85% coverage.

**2.2 Integration Tests**

* **Goal:** Ensure smooth interaction between different units/modules.
* **Description:** Integration tests validate the interaction between integrated units/modules to detect interface issues.
* **Implementation:**
  + Write integration tests for critical paths and modules interaction.
  + Use test databases and environments to mimic real-world interactions.
* **Current Status:** Integration tests cover all critical interactions.

**2.3 End-to-End (E2E) Tests**

* **Goal:** Validate the complete functionality of the application from start to finish.
* **Description:** E2E tests simulate real user scenarios to ensure the entire application works as expected.
* **Implementation:**
  + Write E2E tests using frameworks such as Selenium, Cypress, or Puppeteer.
  + Test key user workflows such as registration, login, product browsing, and checkout.
* **Current Status:** Key user workflows are covered by E2E tests.

**3. Clean Code Approach**

**3.1 Coding Standards**

* **Goal:** Maintain a consistent coding style across the codebase.
* **Description:** Adhering to coding standards improves code readability and maintainability.
* **Implementation:**
  + Follow industry-recognized coding standards (PEP 8 for Python, Google Java Style Guide, etc.).
  + Use linters to enforce coding standards automatically.
* **Current Status:** Coding standards are documented and enforced via linters.

**3.2 Code Readability**

* **Goal:** Write code that is easy to read and understand.
* **Description:** Readable code reduces the cognitive load for developers, making it easier to maintain and extend.
* **Implementation:**
  + Use meaningful variable and function names.
  + Write concise and clear comments where necessary.
  + Follow consistent indentation and spacing practices.
* **Current Status:** Code readability is reviewed during code reviews.

**3.3 Refactoring**

* **Goal:** Continuously improve the codebase.
* **Description:** Refactoring involves restructuring existing code without changing its external behavior to improve non-functional attributes.
* **Implementation:**
  + Regularly review and refactor code to eliminate code smells and improve structure.
  + Use automated refactoring tools where applicable.
* **Current Status:** Refactoring is part of the development process and performed regularly.

**4. CI/CD Integration**

**4.1 Continuous Integration (CI)**

* **Goal:** Ensure code quality and functionality are validated before integration.
* **Description:** CI involves automatically running tests and analysis tools on each code change to detect issues early.
* **Implementation:**
  + Set up a CI pipeline using tools like Jenkins, GitHub Actions, or GitLab CI.
  + Include steps for running unit, integration, and E2E tests.
  + Integrate static code analysis and code coverage reports.
* **Current Status:** CI pipeline is fully integrated and runs on each code change.

**4.2 Continuous Deployment (CD)**

* **Goal:** Automate the deployment process to ensure quick and reliable releases.
* **Description:** CD automates the deployment of code changes to production, ensuring that new features and fixes are delivered continuously.
* **Implementation:**
  + Set up a CD pipeline to automate the deployment process.
  + Include steps for deploying to staging environments for final validation.
  + Implement automated rollback mechanisms in case of deployment failures.
* **Current Status:** CD pipeline is in place with weekly releases to production.

By adhering to these code quality metrics and a clean code approach, the project ensures high standards of software development, resulting in a robust, maintainable, and scalable application. These practices also contribute to more accurate estimations and better project planning.