Quality Mindset Framework

Phase 1 - Establishing the Foundation (Weeks 1-4):

1. *Kickoff Meetings:*

- Conduct kickoff meetings with developers and quality engineers/analysts to introduce the quality mindset initiative.

- Align on goals, benefits, and success metrics to ensure a shared understanding and commitment.

2. *Workflow Review:*

- Review current developer and testing workflows to identify pain points, process gaps, and collaboration issues.

- Gather data on defect rates, test coverage, and other relevant metrics to establish a quality baseline.

3. Code Review and Peer Feedback:

- Introduce code review checklists and make it mandatory for all code to go through peer review before release.

- Schedule regular code reviews and provide tools to capture feedback effectively.

4. *Unit Testing Standards:*

- Define and roll out unit testing standards to ensure comprehensive test coverage.

- Coach developers on writing effective unit tests early in the implementation process.

5. *Code Coverage Goals:*

- Establish code coverage goals based on component criticality.

- Set targets for achieving high code coverage in critical areas.

6. *CI/CD Automation:*

- Implement Continuous Integration/Continuous Deployment (CI/CD) automation for builds, syntax checking, and unit tests.

- Deploy test environments early in the development process through automation.

7. *Risk-Based Testing:*

- Guide the quality engineers/analysts towards a risk-based approach to testing.

- Identify high-risk areas through usage data, complexity analysis, and impact analysis.

- Focus manual testing efforts on these high-risk areas.

8. *Collaboration Meetings:*

- Initiate weekly collaboration meetings between developers and quality engineers/analysts to review defects, improve tests, analyse trends, and share knowledge.

Phase 2 - Expanding Test Coverage and Analytics (Weeks 5-8):

1. *Unit Test Coverage:*

- Expand unit test coverage based on risk analysis and component criticality.

- Aim for over 90% coverage in high-risk components.

2. *Self-Service Testing:*

- Provide self-service access to testing environments and tools for developers.

- Encourage developers to shift left on testing by enabling them to perform testing tasks independently.

3. *Automation Opportunities:*

- Identify opportunities for automation in integration, performance, security, and UI testing based on repetitive workflows.

- Start implementing automation in these areas to improve efficiency and effectiveness.

4. *Quality Dashboard Reporting:*

- Build out a quality-focused dashboard reporting system that provides insights into defects, coverage, releases, and trends.

- Review the dashboard regularly to monitor progress and identify areas for improvement.

5. *Root Cause Analysis:*

- Conduct deeper root cause analysis on defects to identify underlying issues.

- Conduct post mortems on critical issues to prevent their recurrence in the future.

6. *Production Monitoring and User Feedback:*

- Start collecting production monitoring data and user feedback to inform testing scenarios.

- Incorporate real-world usage data into testing strategies to ensure comprehensive coverage.

Phase 3 - Continual Improvement and Knowledge Sharing (Ongoing):

1. *Collaboration and Knowledge Sharing:*

- Utilise regular collaboration meetings to facilitate knowledge sharing between developers and QA professionals.

- Establish forums or platforms for broad knowledge sharing on quality across the organisation.

2. *Automated Metrics Delivery:*

- Automate the delivery of code quality metrics from analysis tools to enhance visibility and enable data-driven decision-making.

3. *Progress Tracking and Course Correction:*

- Track progress towards coverage, defect, and productivity goals.

- Course correct as needed based on the insights gained from metrics and feedback.

4. *Post-Release Telemetry:*

- Use post-release telemetry to validate test coverage and scenarios.

- Update testing strategies based on real-world usage data to ensure ongoing relevance and effectiveness.

5. *Cultivating a Culture of Quality:*

- Foster a culture of quality through leadership, coaching, and the provision of appropriate tools and resources.

- Promote modern techniques such as Behavior-Driven Development (BDD), shift left testing, and self-service access to testing.

6. *Focus on Simplicity and Documentation:*

- Maintain a focus on simplicity, documentation, and other code quality standards.

- Encourage developers to adhere to best practices and ensure that code is well-documented and easy to understand.