Defect Management Workflow

This document outlines the comprehensive defect management workflow for the GitHub Spec Kit Training Program.

Overview

Our defect management system ensures systematic tracking, resolution, and prevention of issues throughout the training program lifecycle.

Defect Categories

1. Training Content Issues

- Curriculum errors or outdated information
- Exercise instructions that are unclear or incorrect
- Missing or broken links in training materials
- Template formatting or content issues

2. Technical Issues

- Script execution failures
- Environment setup problems
- Testing framework issues
- Integration failures

3. Documentation Issues

- Incomplete or inaccurate documentation
- Missing prerequisites or setup instructions
- Broken internal or external links
- Formatting inconsistencies

Defect Lifecycle

1. Detection

- **Automated Detection**: Scripts and tests identify issues
- **Manual Detection**: Users report issues via GitHub issues
- **Review Detection**: Code review process identifies potential problems

2. Logging

- All defects are logged in DEFECT_LOG.md (DEFECT_LOG.md)
- Each defect receives a unique identifier using format: **DEF-YYYY-####** (e.g., DEF-2025-0001)
- Severity and priority are assigned using standardized GitHub labels
- Initial assessment and categorization

3. Assignment

- Defects are assigned based on category and expertise

- Training content issues → Content team
- Technical issues → Development team
- Documentation issues → Documentation team

4. Resolution

- Root cause analysis performed
- Fix implemented and tested
- Documentation updated as needed
- Resolution verified by reporter when possible

5. Closure

- Defect marked as resolved in tracking system
- Lessons learned documented
- Process improvements identified

Severity Levels

Critical P1

- Training program cannot proceed
- Major functionality broken
- Security vulnerabilities
- **Response Time**: Immediate within 2 hours (24/7 coverage)
- **First Responder**: On-call engineer or repository maintainer
- **Escalation**: Immediate notification to project lead

High P2

- Significant impact on training effectiveness
- Important features not working
- Workaround available but difficult
- **Response Time**: Same day within 8 hours (business hours: 9 AM - 6 PM EST)
- **First Responder**: Assigned team lead
- **Escalation**: Project lead notification within 4 hours if unresolved

Medium P3

- Moderate impact on user experience
- Minor functionality issues
- Easy workaround available
- **Response Time**: Within 2 business days
- **First Responder**: Team member with relevant expertise
- **Escalation**: Weekly review if unresolved

Low P4

- Cosmetic issues
- Enhancement requests
- Documentation improvements
- **Response Time**: Within 1 week
- **First Responder**: Any available team member

- **Escalation**: Monthly review for prioritization

GitHub Labels Mapping

Our severity levels map to the following GitHub labels for consistent tracking:

- **P1 Critical** → `priority: critical`, `severity: high`
- **P2 High** → `priority: high`, `severity: medium`
- **P3 Medium** → `priority: medium`, `severity: low`
- **P4 Low** → `priority: low`, `enhancement`

Tools and Scripts

Automated Defect Management

- **fix_defects.py (scripts/fix_defects.py)** - Automated defect resolution
- **update_defect_log.py (scripts/update_defect_log.py)** - Defect log maintenance
- **fix_workflows.py (scripts/fix_workflows.py)** - Workflow issue resolution

Testing and Validation

- **Test Suite (env/tests/)** - Comprehensive testing framework
- **Environment Validation (scripts/validate_environment.sh)** - Setup verification

Reporting Guidelines

For Users

1. Check existing issues before creating new ones
2. Use appropriate issue templates:
   - Bug Report (.github/ISSUE_TEMPLATE/bug_report.md)
   - Training Question (.github/ISSUE_TEMPLATE/training_question.md)
3. Provide detailed reproduction steps
4. Include environment information
5. For security vulnerabilities, see SECURITY.md (.github/SECURITY.md)

For Contributors

1. Follow the pull request template
2. Include tests for bug fixes
3. Update documentation as needed
4. Reference related issues in commits

Metrics and Monitoring

Key Performance Indicators

Mean Time to Resolution (MTTR) by severity level:
- **Owner**: Engineering Manager
- **Review Cadence**: Weekly for P1/P2, Monthly for P3/P4

Defect Escape Rate - issues found in production:
- **Owner**: Quality Assurance Lead
- **Review Cadence**: Monthly assessment

Customer Satisfaction - user feedback on resolutions:
- **Owner**: Product Manager
- **Review Cadence**: Quarterly survey and analysis

Defect Density - defects per training module:
- **Owner**: Content Team Lead
- **Review Cadence**: After each major content update

Reporting Schedule

- **Weekly Reviews**: P1/P2 defects, MTTR analysis
- **Monthly Reviews**: All severity levels, escape rate assessment
- **Quarterly Reviews**: Customer satisfaction, process improvements

Process Improvement

- Regular retrospectives led by Project Manager
- Root cause analysis for critical defects
- Process refinement based on lessons learned
- Tool and automation improvements

Contact and Support

For defect-related questions or escalations:
- Create an issue using our templates
- Tag appropriate team members
- For critical issues, contact repository maintainers directly
- For security vulnerabilities, follow SECURITY.md (.github/SECURITY.md) procedures

Related Documentation

- DEFECT_LOG.md (DEFECT_LOG.md) - Current defect tracking
- Training Outcomes (training/metrics/training-outcomes.md) - Quality metrics
- Integration Plans (docs/IntegrationPlan.md) - System integration details
- Security Policy (.github/SECURITY.md) - Vulnerability reporting procedures

---