**BUGS AND ISSUES TRACKING PROCESS REVIEW**

**VERIFACTS SERVICES PRIVATE LIMITED**

**Sprint Planning:** Bug tracking should be seamlessly integrated into sprint planning. Teams allocate a portion of each sprint for bug fixes and improvements.

**Backlog Prioritization:** Maintaining a well-prioritized backlog is essential. This ensures that the important issues are addressed first, aligning with the product owner's requirements.

# **Bug Severity Levels**

Defining and understanding the severity levels of bugs is vital. These levels often include:

**Critical:** Bugs that completely disrupt the software's functionality and may cause data loss or severe security vulnerabilities.

**Major:** Bugs that have a significant impact on the user experience or core functionality but don't necessarily result in data loss.

**Minor:** Non-critical issues that don't seriously impact functionality but may still affect the user experience.

**Cosmetic:** Minor aesthetic issues that do not affect functionality but may be distracting or unprofessional.

Understanding the severity of a bug helps the team prioritize and allocate resources effectively.

# **Bug Lifecycle**

Managing the entire lifecycle of a bug is crucial for effective bug tracking and issue management. This lifecycle typically includes the following stages:

* Open: The bug is reported and awaits investigation.
* Assigned: The bug is assigned to a team member for further analysis.
* In Progress: The team member works on resolving the issue.
* Testing: The bug fix is tested to ensure it has been effectively resolved.
* Verified: The issue has been resolved and confirmed as fixed.
* Closed: The bug is closed, and its status is updated to reflect that it has been resolved.

# **Reporting and Metrics**

Utilizing metrics is a powerful way to improve bug tracking and issue management. Key metrics to consider include:

**Time to Resolution:** Measure the average time it takes to resolve different types of bugs. This helps in identifying bottlenecks in the process.

**Bug Density:** Calculate the number of bugs per line of code. High bug density might indicate problematic areas in your codebase.

**Reopened Bugs:** Keep track of how many bugs are reopened after being marked as resolved. This can highlight recurring issues or inadequate testing.

**Customer Feedback:** Use customer feedback and user-reported issues as a source of valuable information for bug identification.

# **Bugs and Issues tracking process:**

**Issue Identification:**

Users or Testers: Issues are often identified by users during testing or by dedicated testers during the quality assurance phase.

Automated Testing: Utilize automated testing tools to identify potential bugs or issues in the code.

**Issue Logging:**

Bug Report: Create a detailed bug report that includes information such as:

* Steps to reproduce the issue.
* Expected behavior.
* Actual behavior.
* Environment details (OS, browser, device, etc.).
* Severity and priority of the bug.
* Attachments like screenshots or log files.

**Issue Assignment:**

Assign Responsibility: Assign the bug to the appropriate individual or team responsible for fixing it. This is often a developer, but it could involve other roles, such as designers or system administrators.

**Issue Prioritization:**

Priority Levels: Prioritize issues based on severity and impact on the software.

Critical/Major/Minor: Use a classification system to categorize issues into critical, major, or minor categories.

**Development and Testing:**

Development: Developers work on fixing the identified issues.

Code Review: Peer review of the code changes to ensure quality and adherence to coding standards.

Unit Testing: Developers perform unit tests to ensure that the changes have not introduced new issues.

**Issue Resolution:**

Resolution Status: Update the status of the issue as it progresses through the resolution process.

Communication: Communicate progress and resolution timelines to stakeholders.

**Verification:**

Testing: Test the fixed code to ensure that the reported issue has been resolved.

Regression Testing: Verify that the bug fix has not introduced new issues.

**Closure:**

Confirmation: Confirm with the reporter or tester that the issue has been resolved to their satisfaction.

Documentation: Update documentation as necessary to reflect the resolution of the issue.

**Reporting and Analysis:**

Reporting Tools: Use reporting tools to track the overall health of the software and identify patterns or recurring issues.

Analysis: Analyze the data to improve development processes and prevent similar issues in the future.

**Continuous Improvement:**

Retrospective: Conduct retrospectives to identify areas for improvement in the bug tracking and resolution process.

Feedback Loop: Establish a feedback loop with users and testers to continuously improve the software quality.