**Q1. When a bug is fixed, how do you ensure that the fix doesn't introduce new issues or break existing functionality (regression testing)?**

**1. Regression testing:** Re-run existing tests

**2. Verify fix:** Test the specific bug fix

**3. Test adjacent features:** Ensure related functionality works

**4. Run automated tests:** Catch unexpected issues

**5. Manual testing:** Double-check critical functionality

**Q2. How do you handle a situation where you've reported a bug, but the release deadline is fast approaching, and the bug hasn't been fixed yet?**

**1.** Escalate critical bugs to the project manager or product owner.

**2**. Provide a risk assessment explaining the bug's potential impact.

**3.** Suggest workarounds to minimize the impact, if possible.

**4.** Prioritize non-critical bugs for the next release."

This approach helps manage bugs effectively, minimizing delays and risks, and ensuring a smooth release.

**Q3.** **What is your approach if you find a bug that is difficult to reproduce due to environmental or configuration differences?**

To solve a tricky bug, I follow these steps:

**1. Recreate the scene:** Try to set up the same environment where the bug happened.

**2. Gather clues:** Collect logs, screenshots, and user inputs to spot patterns.

**3. Use special tools**: Enable debugging tools or detailed logging to track the issue.

**4. Team up with developers:** Work together to analyze possible causes.

**5. Test in different worlds:** Try the same thing on different operating systems, browsers, or network conditions.

**Q4. Can you give an example of a "high priority" bug vs. a "high severity" bug? How do they differ in the real-world context of software development?**

 **High Priority Bug:** A typo in the company’s name on the homepage—minor impact but urgent fix.

 **High Severity Bug:** A crash occurring when a user submits a payment—critical failure but may not block all users.

 Priority is based on **business urgency**, while severity is based on **technical impact**.

**Q5. How do you track the status of long-standing or unresolved bugs that get pushed from one sprint to the next?**

**1**. **Bug tracking tool:** Use a tool like Jira, Trello, or Asana.

**2. Labeling:** Label bugs as "long-standing" or "unresolved".

**3. Priority:** Set a priority level to ensure focus.

**4. Regular review:** Review and discuss bugs in sprint planning.

**5. Escalation:** Escalate critical bugs to the project manager or team lead.

This approach helps ensure that long-standing bugs aren't forgotten!

**Q6.** **How do you verify and test bug fixes to ensure they don't affect other areas of the product (e.g., regression testing, impact analysis)?**

**1. Regression testing:** Re-run existing tests to ensure fix didn't break other areas.

**2. Impact analysis:** Assess potential effects on related features or components.

**3. Automated testing:** Use automated tests to quickly identify regressions.

**4. Manual testing:** Perform thorough manual testing to validate fix.

**5. Code review:** Review code changes to ensure they meet quality standards.

**Q7. What would you do if a bug impacts multiple users or environments, and there are different ways to reproduce it?**

**If a bug affects many users and environments, and happens in different ways:**

**1. Find common links:** Look for shared factors like OS, browser, or device.

**2. Document variations:** Write down all differences and test each one separately.

**3. Team up with developers:** Analyze logs together to spot patterns.

**4. Use monitoring tools:** Track real-time data to understand what's happening.

By working together and using the right tools, you can identify the root cause and fix the bug!

**Q8. In a real-world agile environment, how do you deal with bugs that surface during the final stages of the sprint, especially when testing is incomplete?**

When bugs appear late in the sprint:

**1. Decide quickly:** Is the bug critical or can it waits?

**2**. Fix critical bugs immediately and re-test.

**3.** Document non-critical bugs for the next sprint.

**4. Review and adjust:** Improve testing for the next sprint."

This means you prioritize bugs, fix critical ones quickly, document non-critical ones, and review to improve testing for the next sprint!

**Q9. How do you ensure effective communication between testers and developers throughout the bug life cycle?**

To ensure effective communication:

**1. Clear bug reporting:** Provide detailed, reproducible steps.

**2. Regular updates:** Share bug status, progress, and changes.

**3. Collaboration tools:** Utilize platforms like Jira, Slack, or Trello.

**4. Face-to-face meetings:** Hold regular sessions for discussion and clarification.

**5. Feedback loops**: Encourage open communication and feedback.

This ensures testers and developers are aligned throughout the bug life cycle!

**Q10. Have you encountered a situation where a bug was marked as "Duplicate" but you believe it to be a unique issue? How did you handle the situation?**

If a bug is marked "Duplicate" but you think it's different:

**1.** Check the original report for similarities.

**2.** Add more proof (screenshots, logs, videos).

**3.** Talk to the developer or team lead for clarity.

**4.** If needed, reopen the bug with new details.

This helps ensure unique bugs get fixed!

**Q11.** **How do you manage bugs that arise during user acceptance testing (UAT) and are reported by clients or end-users?**

When clients or end-users report bugs during testing:

**1.** Recreate the issue to understand it.

**2.** Gather details like steps and screenshots.

**3.** Fix critical bugs first that impact the most.

**4**. Keep clients updated on progress.

**5.** Quickly fix critical issues before release.

This ensures bugs get fixed promptly and clients stay informed!

**Q12. When you encounter a "critical" bug that could impact the product release, what steps do you take to ensure the bug is addressed immediately?**

When a critical bug is found:

**1.** Tell the team and bosses right away.

**2**. Figure out what's causing the problem.

**3.** Test a fix to make sure it works.

**4.** Fix it as soon as possible.

This way, big problems get solved quickly!