1. What is defect management ?

* Defect management is the process of identifying, tracking, and resolving problems (or "defects") in a product, system, or project.

In simple words:

1. **Finding Defects**: First, the problems or issues are discovered, usually during testing or through user feedback.
2. **Tracking Defects**: Once a defect is found, it is logged into a system where it can be tracked and monitored.
3. **Fixing Defects**: The team works to fix the defect, which might involve fixing the code, design, or process causing the problem.
4. **Verifying Fixes**: After the fix is made, it is tested again to ensure the defect is resolved and nothing else was broken.
5. **Closing the Defect**: Once the defect is fixed and verified, it is marked as closed.

Defect management helps ensure that products or systems are of good quality, meet user expectations, and perform as intended.

2. What are the types of defects ?

There are several types of defects that can occur in products or systems, depending on what is being developed (software, hardware, etc.). Here are some common types:

### **1. Functional Defects**

These defects occur when the product or system doesn't work as expected or doesn’t meet its intended purpose.

* **Example**: A button in an app that doesn’t perform the intended action when clicked.

### **2. Performance Defects**

These defects affect how well the product or system performs. It could be too slow, unresponsive, or resource-hungry.

* **Example**: A website that takes a long time to load or a game that lags during gameplay.

### **3. Usability Defects**

These defects make the product difficult or confusing to use due to poor design or user interface issues.

* **Example**: An app with hard-to-understand navigation or unclear instructions.

### **4. Security Defects**

These involve vulnerabilities in the product that could be exploited by attackers.

* **Example**: A website that allows unauthorized users to access sensitive data due to weak passwords.

### **5. Compatibility Defects**

These defects happen when the product doesn’t work properly across different environments, devices, or systems.

* **Example**: A mobile app that works on Android but crashes on iOS.

### **6. Regression Defects**

These occur when a recent change or update introduces new issues or causes previously fixed problems to reappear.

* **Example**: A new software update that breaks a feature that worked fine before.

### **7. Syntax or Code Defects**

These are errors in the software code that cause problems like crashes or unexpected behavior.

* **Example**: A typo in the code that causes the program to crash.

### **8. Data Defects**

These involve incorrect or corrupted data within the system.

* **Example**: A software application that displays incorrect customer information due to a database error.

### **9. Boundary Defects**

These defects occur when the system doesn’t handle unusual inputs or edge cases correctly.

* **Example**: A form that crashes when a user enters a very large number or special character.

### **10. Configuration Defects**

These happen when the system is not set up properly, causing it to malfunction.

* **Example**: A server misconfiguration that prevents a website from loading properly.

Each type of defect requires different approaches for detection, tracking, and fixing. Identifying these defects early helps ensure better product quality and user satisfaction.

3. What are defect stages ?

* The stages of a defect, in simple words, describe the steps a problem goes through from when it is first discovered to when it is fixed. Here are the stages:

1. **New**: The problem is found and reported, but no one has started fixing it yet.
2. **Assigned**: The problem is given to someone (usually a developer or a team) to work on and fix.
3. **Open**: The person or team is working on fixing the problem, such as finding out why it happened and how to solve it.
4. **Fixed**: The problem has been fixed by the team, and they apply the solution.
5. **Retesting**: After the fix, the problem is tested again to make sure it’s truly fixed and hasn’t caused any new issues.
6. **Closed**: The problem is confirmed to be fixed, and it’s marked as resolved.
7. **Reopened** (if needed): If the problem wasn’t fixed properly, or if it comes back, it gets reopened for further investigation.

These stages help keep track of problems and ensure they are properly addressed.

4. What is BUG(defect) life cycle ?

The **bug life cycle** is the process a bug (or defect) goes through from when it's first discovered until it's fixed and closed. In simple terms, it’s the journey a bug takes to be identified, resolved, and verified. Here's an overview of the bug life cycle in simple words:

### **1. New (Identified)**

* The bug is found and reported by someone, like a tester, and logged into a system.
* **Example**: A tester finds that a button doesn't work in an app.

### **2. Assigned**

* The bug is assigned to someone (usually a developer) who will work on fixing it.
* **Example**: A developer is told to fix the broken button.

### **3. Open**

* The developer starts working on fixing the bug, trying to figure out what caused it and how to fix it.
* **Example**: The developer looks at the code and figures out what went wrong with the button.

### **4. Fixed**

* The developer applies a solution or makes changes to fix the bug.
* **Example**: The developer updates the code so the button works properly.

### **5. Retesting**

* After the bug is fixed, it is tested again to ensure that the fix worked and that the bug is gone.
* **Example**: The tester checks if the button now works as expected.

### **6. Closed**

* If the bug is fixed and verified, it’s marked as "closed." This means no further work is needed on it.
* **Example**: The tester confirms the button works, and the bug is closed in the tracking system.

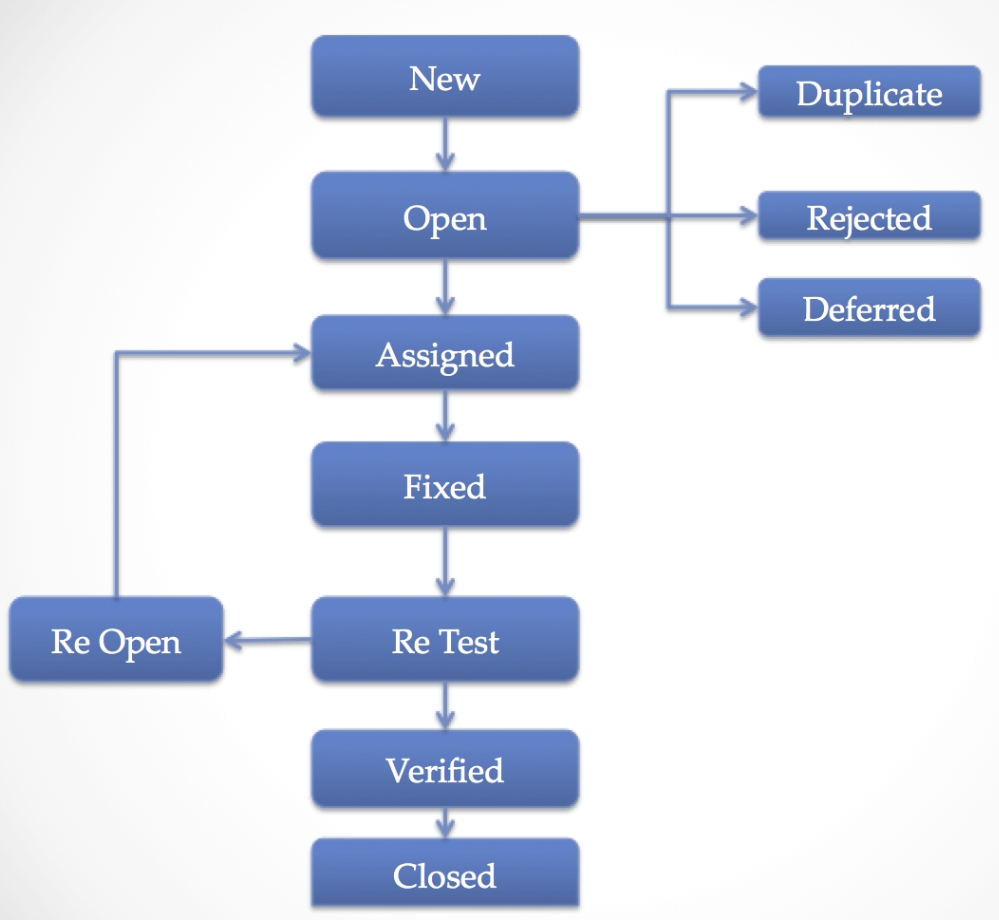
### **7. Reopened (if needed)**

* If the bug was not fully fixed or if it comes back after being marked closed, it is reopened and sent back for more work.
* **Example**: After testing, the button still doesn't work correctly, so the bug is reopened to be fixed again.

### **Bug Life Cycle Summary:**

1. **New**: The bug is reported.
2. **Assigned**: The bug is given to someone to fix.
3. **Open**: Work begins to fix the bug.
4. **Fixed**: The bug is fixed.
5. **Retesting**: The fix is tested to ensure it works.
6. **Closed**: The bug is confirmed fixed and closed.
7. **Reopened**: If the bug isn't fixed properly, it is reopened.

This life cycle ensures that bugs are handled systematically and that issues are resolved before the product is released.



**BUG LIFE CYCLE**

5. What is defect management scenario ?

* Defect Management Scenario
* The typical defect lifecycle : Submitted > Open > Assigned > Resolved > Migrated > Retest(Solved Defect) > Closed
* Developer unable to fix defect in first attempt : Submitted > Open > Assigned > Resolved > Migrated > Retest(Still Defect) > Reopen > Assigned > Resolved > Migrated > Retest(Solved Defect) > Closed
* Defect is determined by test lead to be invalid : Submitted > Rejected
* Defect is determined by developer to be invalid : Submitted > Open > Assigned > Rejected(Not a Defect)
* Developer unable to recreate defect : Submitted > Open > Assigned > Rejected(Not a Defect)
* Client decides to defer correction of a defect : Submitted > Open > Deferred(By Project Leader)

6. Why defect report necessary ?

* Defect reports are among the most important deliverables to come out of test. They are as important as the test plan and will have more impact on the quality of the product than most other deliverables from the test.
* Effective defect reports will:
* Reduce the number of defects returned from development
* Improve the speed of getting defect fixes
* Improve the credibility of test
* Enhance teamwork between test and development

7. What are the fields of defect report ?

* You should provide enough detail while reporting the bug keeping in mind the people who will use it – test lead, developer, project manager, other testers, new testers assigned etc.
* This means that the report you will write should be concise, straight and clear.
* Following are the details your report should contain:

1. Bug Title
2. Bug identifier (number, ID, etc.)
3. The application name or identifier and version / Test Case Id or Description
4. The function, module, feature, object, screen, etc. where the bug occurred
5. Environment (OS, Browser and its version)
6. Bug Type or Category/Severity/Priority
7. Bug status (Open, Pending, Fixed, Closed, Re-Open)
8. Test case name/number/identifier
9. Bug description
10. Steps to Reproduce
11. Actual Result
12. Tester Comments

8. What does the tester do when the defect is fixed ?

* Once the reported defect is fixed, the tester needs to re-test to confirm the fix. This is usually done by executing the possible scenarios where the bug can occur. Once retesting is completed, the fix can be confirmed and the bug can be closed. This marks the end of the bug life cycle.