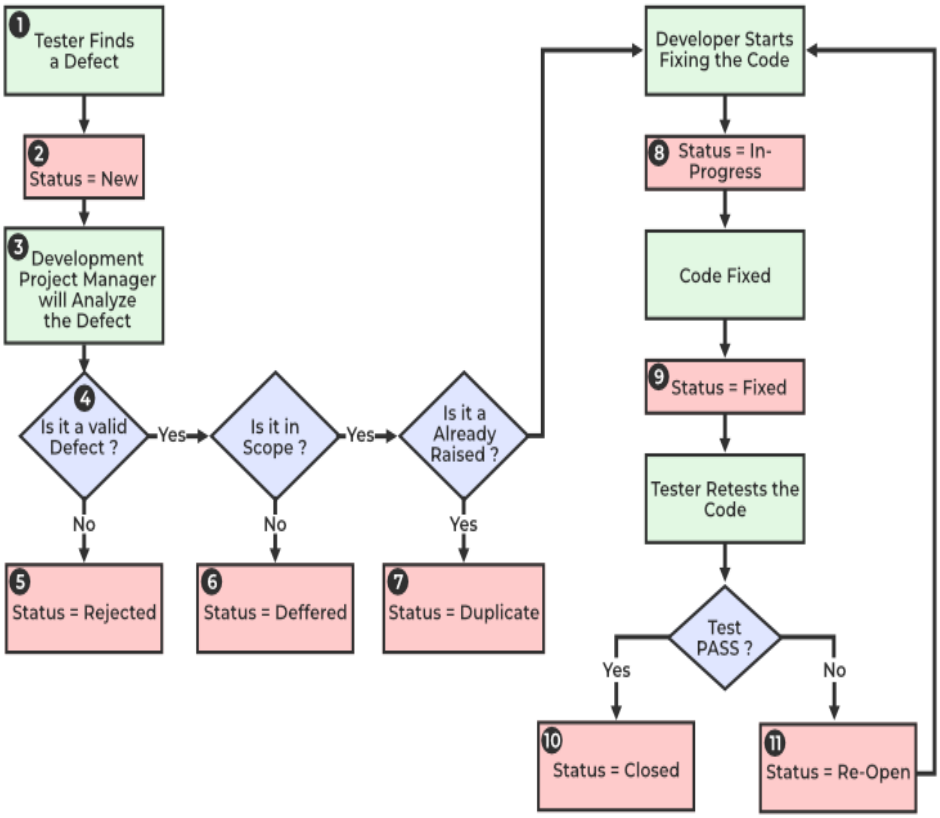
TASK-16 -DEFECT LIFE CYCLE

In the document we mentioned details about defect life cycle, different status of defects and defect report.

Generally, we have various status of bug in a bug life cycle like-new, open, assigned, fixed, retest, reopen these are different status of defect life cycle.

In this the status managed by a tester are new, closed and reopen.

The status managed by the developer are open, fix, hold, duplicate, differed, reject.



From the above flowchart we can clearly understand the status of a defect in defect life cycle.

Defect status or bug status is the status through which the bug is going through:

1.**NEW**: When the bug is found by the tester, it comes under the new status. The tester provides a defect document to the developer after which the developer starts working on the defect.

2.**ASSIGNED:** defects that are in the status new will be approved and assigned to the development team to work on it.

3**.OPEN**: In an open state the defect is addressed by the developer and the developer team works on fixing the defect.

4.**FIXED**: After making required changes in the code or fixing the bug developed by the tester the status is noted as fixed by the developer.

5.**PENDING REQUEST**: After fixing the defect mentioned by the tester the developer again sends the application to the tester to retest the application. Then the application is pending in the tester side for retesting, so it is mentioned as "Pending Request" state.

6.**RETEST**: In this stage the developer checks whether the bug is fixed or not by the developer by retesting the application this state is mentioned as "Retest".

7.**REOPEN**: After retesting the tester finds the defect still exists in the application then the defect needs to reopened and entire steps need to be done again to debug the application.

8.**VERIFIED**: After retesting the application sent by the developer the if we find the bug is verified then the application is verified, and the status of the application is "verified".

9.**CLOSED**: After verifying that the bug is cleared from the application then the defect is closed, and state is defined as "Closed".

Few other states of application are:

1.**REJECTED**: If the developer finds the defect is not genuine, they mark the defect status as rejected.

2.**DEFFERED**: If the developer feels the bug can be fixed in further updates or releases the developer mentions the status of defect as deferred.

3.**DUBLICATE:** It is possible that the defect is repeated twice or there is another defect similar to this then the developer mentions the defect as a Dublicate.

4.**NOT A DEFECT**: If the defect doesn't have any effect on the functionalities of the application, then its status is mentioned as "Not A Defect".

5.**NON-REPORODUCABLE**: If the defect is not reproducible due to software mismatch or other reason than the defect is mentioned as “non-reproducible”.

6.**CAN’T BE FIXED**: If the developer is unable to fix the defect due to lack of technical support, high cost to fix a bug or lack of skill the bug is stated as “can’t” be fixed.

7.**NEED MORE INFORMATION**: If the development team is unable to produce the defect by following the same steps mentioned in the document, then the developer mentions the document as “Need more information”.

SEVERITY:

Severity in defects mentions the level of harm the defect can do on the system or application. It is a way of categorizing the defects based on the severity.

Some common classifications of defects are:

- **Critical:** Defects that cause complete system failure, data loss, security breaches, or other critical issues that prevent the software from functioning at all. These defects usually require immediate attention and resolution.
- **Major**: Defects that significantly impair the functionality of the software or cause major inconvenience to users. While the software may still function, it does so with notable limitations or issues that affect its usability.
- **Moderate**: Defects that have a noticeable but not critical impact on the software's functionality. They may cause inconvenience or inefficiency but do not render the software unusable.
- **Minor:** Defects that have a minimal impact on the software's functionality or usability. They may be cosmetic issues or minor annoyances that don't affect the core functionality of the software.
- **Trivial:** Defects that have little to no impact on the software's functionality or usability. These are usually minor cosmetic issues or small inconsistencies that don't affect the overall user experience.

PRIORITY:

Priority in software development refers to the relative importance or urgency assigned to a task, feature, or defect. It determines the order in which items should be addressed or worked on by the development team. Priority is often decided based on factors such as the impact on users, business objectives, deadlines, and dependencies.

Here's a typical priority scale used in software development:

- **Critical:** Tasks or defects that must be addressed immediately as they significantly impact the functionality, usability, or security of the software. Critical items often require immediate attention to prevent severe consequences.
- **High:** Tasks or defects that are important and should be addressed as soon as possible, but they do not necessarily require immediate attention like critical issues. High priority items may have a significant impact on users or business goals if not resolved promptly.
- **Medium:** Tasks or defects that are important but can be addressed after critical and high priority items are resolved. Medium priority items may affect usability or efficiency but are not urgent.
- **Low:** Tasks or defects that have minimal impact on users or business goals and can be addressed later, after critical, high, and medium priority items are resolved. Low priority items are often minor enhancements or non-critical bugs.

COMBINATIONS OF PRIORITY AND SEVERITY:

- **Critical Severity, High Priority:** Defects that cause severe issues such as system crashes, data corruption, or security breaches and require immediate attention to prevent major disruptions to the software's functionality or to address critical security vulnerabilities.
- **Major Severity, High Priority:** Defects that significantly impair the usability or functionality of the software but do not cause complete system failure. These issues may have a notable impact on users or business operations and need to be addressed urgently.
- **Critical Severity, Medium Priority:** While the severity of these defects is critical and could lead to severe consequences, they might not be as urgent as other critical

issues. They still require prompt attention, but they can be scheduled and managed alongside other high-priority tasks.

- **Moderate Severity, High Priority:** Defects that have a moderate impact on the software's functionality but are deemed highly important due to their impact on user experience or business operations. These issues should be addressed promptly to prevent significant disruptions.
- **Minor Severity, Low Priority:** Defects that have minimal impact on the software's functionality or usability and are not considered urgent. These issues can be addressed during a later development cycle or when resources become available after more critical tasks are completed.
- **Trivial Severity, Low Priority:** Defects that have little to no impact on the software's functionality or user experience and can be safely deferred later. These issues are often minor cosmetic issues or small inconsistencies that do not affect the overall usability of the software.

BASIC TERMINOLOGY USED IN DEFECT LIFE CYCLE:

Defect leakage: The defect that was found in developing or testing phases but has made its way to the operational production environment is called defect leakage.

Defect leakage= (defects found in production/total no.of defects) *100

Defect ageing: The time interval between the date of detection of bug to the date of bug closure is called defect ageing.

Defect rejection ratio: The ratio of number of defects accepted by the development team to the number of defects raised by the testers is called the defect rejection ratio.

Defect removal efficiency: Defect Removal Efficiency relates to the ability to remove defects introduced to a system by a project during the project life cycle. At its simplest DRE can be expressed as a percentage.

DRE = (total defects found during the project/total defects introduced by the project) *100. Bug triaging: Bug triaging is a process where we discuss the priority of the bugs in a bug triage meeting.

DEFECT REPORT TEMPLATE:

BUG OR DEFECT REPORT TEMPLATE:

Consider the defect where the application doesn't show the confirmation notification after booking the tickets in "INSTAGRAM" application.

FIELD	**APPLICATION**
Title	User is not getting notification for a message.
ID	0011
DATE REPORTED	15-2-24
REPORTER	Divya Mettu
APPLICATION	Instagram
VERSION OF THE BUILD	1.002
PLATFORM/OS	Mobile(android/iOS), windows etc...
SEVERITY	Moderate
PRIORITY	p1
RESOLUTION	1920x1080 and 1280 x 720 pixels
ENVIRONMENT	QA
STEPS TO REPRODUCE	1.Login to Instagram. 2.Open the messaging option. 3.Send the message from another login.
EXPECTED RESULT	1.Message is received to your account. 2.Notification is shown on top of the messaging application.
ACTUAL RESULT	1.Message is received to your account. 2. Notification is not shown on top of the messaging application.

ATTACHMENTS	Please refer the below screen shot
ASSIGNED TO	Harsha
STATUS	New