Assessment for Software Quality Assurance (Intern) Post

Name : Mahruf Zaman Utso

Email : mahruf.z44@gmail.com

Mobile Number: 01950744794

Q1. Explain the steps for Bug Cycle?

Answer:

We know that, in software testing, bug life cycle which is also known as defect life cycle is the set of some specific set of states or steps that defect or bug goes through its entire life cycle.

Below are the steps or states of Defect Life Cycle:

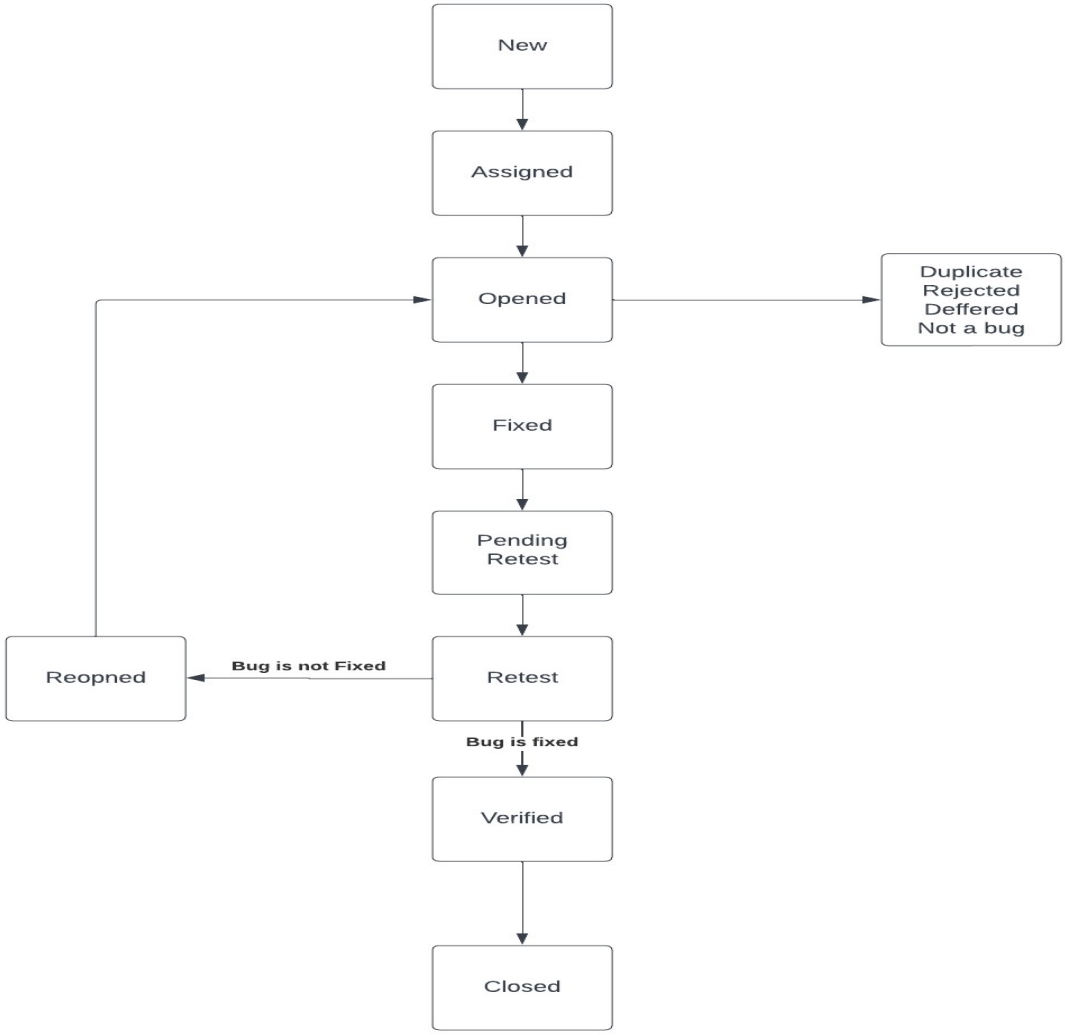

Figure 1: Flow Diagram of Bug Life Cycle

Explanations of the steps:

1. **New:** Defect's status is assigned as new when the bug is logged and posted for the first time.
2. **Assigned:** After the bug is posted in the new state, the leader of the tester approves the bug and assigns the bug to the development team.

3. **Open:** After being assigned, the developer starts analyzing the bug and works on the bug fix.
4. **Fixed:** In this state, a developer makes the required code modification, checks it, and marks the bug as "Fixed."
5. **Pending Retest:** After the defect is fixed, the developer gives a particular code for retesting the code to the tester. Since, the testers are yet to begin the retesting, the status assigned is "Pending Retest."
6. **Retest:** Software tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Retest."
7. **Verified:** After Retesting, If there is no bug detected in the software, then the bug is fixed and the status assigned is "Verified."
8. **Reopened:** The tester updates the status to "Reopened" if the bug still exists after the developer has fixed it. The bug goes through its life cycle once more.
9. **Closed:** If the bug doesn't exist in verified state then tester assigns the status "Closed."
10. **Duplicate:** The status is assigned as "Duplicate" If the defect is repeated twice or the defect corresponds to the same concept of the bug.
11. **Rejected:** If the developer feels the defect is disputed then he can change the defect's status to "Rejected."
12. **Deferred:** The status is assigned to "Deferred" If the current defect is not of a prime priority and if it is expected to get fixed in the next release.
13. **Not a bug:** The status assigned to a bug is "Not a bug" If it does not affect the functionality of the application.

Q2. What is meant by boundary value analysis?

Answer:

Boundary value analysis is a kind of black box testing where input data units are divided into equivalent partitions that can be used to derive test cases and boundary values are tested between equivalence partitions. It reduces time required for testing because of small number of test cases.

Invalid	Valid	Invalid	Invalid
Less than zero to 0	1 to 10	11 to 99	100 to greater than 100
Partition 1	Partition 2	Partition 3	Partition 4

Figure 2: Boundary Value Analysis

In the above Figure, we can see the test data are divided into four partitions. Here, we will check the boundary values from each partition like 0,1,11,100 for the boundary value analysis.

Q3. What is Agile testing and what is the importance of Agile testing?

Answer:

Agile testing is software testing that involves the testing of the software from the customer point of view.

Importance of Agile Testing: This testing does not wait for development team to complete the coding first and then doing the testing. The coding and testing both go simultaneously and it requires continuous customer interaction.

Q4. Explain Low Severity & High Priority Bug?

Answer:

Severity of a bug is the impact of the bug on the system and priority describes the importance and order in which the bug should be fixed.

Low Severity and High Priority Bug means the impact of the bug on the system is low but importance and order in which the bug should be fixed is high.

Example: The spelling mistakes that happens on the cover page or heading or title of an application.

Q5. List the basic components of the defect report format.

Answer:

1. **Defect ID/Name:** The defect's individual and unique identification number.
2. **Defect Description:** A thorough explanation of the Defect, including details on the module where it was discovered.
3. **Version:** Version of the application in which defect was found.
4. **Start Date:** Start Date when this testing for identifying defect is done.
5. **Steps:** Detailed instructions and screenshots are provided so the developer can replicate the errors.
6. **Test Data:** The Data that were put for the testing while identifying the bug.
7. **Reference:** To better grasp the flaw, references are made to the document like the specifications, design, architecture, or even screenshots showing the problem.
8. **Detected By:** ID or name of the tester who reported the issue.
9. **Status:** Current Status of the defect.
10. **Fixed by:** ID or name of the developer who fixed the issue.
11. **Date Closed:** Date when the report is finished or the defect is closed.
12. **Severity**: Severity describes the impact of the defect on the application. It could be could be High/Medium/Low based on the impact.
13. **Priority:** Priority describes the importance and order in which the bug should be fixed. It could be could be High/Medium/Low based on the urgency.

Q6. Explain Low Severity & High Priority Bug?

Answer:

Severity of a bug is the impact of the bug on the system and priority describes the importance and order in which the bug should be fixed.

Low Severity and High Priority Bug means the impact of the bug on the system is low but importance and order in which the bug should be fixed is high.

Example: The spelling mistakes that happens on the cover page or heading or title of an application.

Q7. How do you take screenshots in Selenium Webdriver?

Answer:

In Selenium Webdriver, taking a screenshot involves three steps.

Step 1: Convert web driver object to TakeScreenshot.

```
TakesScreenshot screenShot =((TakesScreenshot)webdriver);
```

Step 2: Call getScreenshotAs method to create image file.

```
File SrceenShotFile=screenShot.getScreenshotAs(OutputType.FILE);
```

Step 3: Copy file to Desired Location

```
File DestinationFile=new File(fileWithPath);

FileUtils.copyFile(SrceenShotFile, DestinationFile);
```

Q8. Explain Regression and Confirmation testing.

Answer:

Regression Testing: Regression Testing is done to verify that modified code does not break existing functionality of the application.

Confirmation Testing: Confirmation Testing is a type of software testing method in which testers check whether the previously posted bugs are corrected or not in the systems or components.

Q9. Explain how do you arrive at a project estimation?

Answer:

There are four steps to follow to arrive at a project estimation.

Step 1 - Divide the whole project into the smallest tasks: Here the whole project task is divided into the subtasks.

Step 2 - Allocate each task to team members: Each task is given to the proper project team member in this step.

Step 3 - Effort estimation for tasks: In this step, various techniques like Functional Point Method and Three Point Estimation are applied to estimate the effort for tasks.

Step 4 - Validate the estimation: After aggregating the estimate of all tasks, it is needed to be forwarded to the management board, who will review and approve it.

Q10. What is the purpose of exit criteria?

Answer:

The purpose of exit criteria is to define when a test level is completed.