***FUNCTIONAL Test Cases***

We have already shown the template for writing test cases.

The main intention of a TE is to write test cases effectively and also efficiently.

Here, the actual result is written after execution of the test cases and most of the time it would be the same as the expected result. It will be different if only the test step is failed. So, actual result field can be skipped, and we can elaborate about the bug in comments field.

Also, the input field can be removed and these details can be added in description field.

Its not that the above mentioned template is a standard one, the template can vary in each company and also with each application depends on TE and Test Lead. But, for testing 1 application, all TEs should follow a standard template which is formulated.

Test cases should be written in such a way that even if a new TE should/can understand and execute the same.

When entering URL hostname, hostname is mentioned as development team throws the s/w to testing team after say 4months – so we are not sure of the hostname.

Username and password are not mentioned as they keep changing. The user might need to change it every 15days depending upon the application.

When writing functional test cases - we 1st check for which field we write test cases and then elaborate accordingly. If say, amount transfer is the field we are writing FT, then elaborate this is not login feature.

Input is required in FT, if the application is data driven, then i/p column is required else it is time consuming.

###### Lessons for writing Functional Test Cases

* Before we start writing test case, come up with options and select the best option and then only start writing test case
* In Expected result, use „should be‟ or „must be‟
* Elaborate only those steps on which we have focus. Do not elaborate on all the steps.
* Highlight object names
* Do not hard code the test case. Write a generic test case
* Organize steps properly so that it reduces lot of execution time.

***INTEGRATION Test Cases***

Something which is covered in FT case should not be written in IT case and something written in IT case should not be written in ST case.

Always start any test case with navigation steps to help new user understand it.

Strategy to write Integration test cases,

* Understand the product
* Identify the possible scenarios
* Prioritize
* Write the test cases according to priority

When TE is writing test cases, the factors he needs to consider are,

* If the test cases are in detail
* Coverage, all test case values or scenarios are depicted
* Think in execution point of view
* Template used to write test cases must be unique

Best test case is when less number of steps are involved, coverage is more, when these test cases are thrown to anyone, they need to understand.

***SYSTEM Test Cases***

Consider the scenario given in **Pg 35 and Pg 36.** Let us write system test cases for the end-to-end business flow. The basic scenario is shown in the above p

**Header of Test Case :**

We always fill the body of the test case first before filling up the header of the test case.

##### Test case name :

*CBO\_AT\_more than balance*

Project name

Module name

Scenario to be tested

##### Requirement number :

*32.3 Amount Transfer*

##### Module name :

*Amount Transfer*

##### Pre-condition :

*Test Engineer should know the balance of user A and user B*

Pre-condition is a set of actions or settings that you should have done before executing step number 1.

**For ex,** If in an application, we are writing test cases for add users, edit users and delete users – the precondition would be – see if user A is added before editing it and deleting it.

##### Test data :

The data we should have before executing step number 1. **Ex –** username, password, account number of users.

The test lead may be give the test data like username or password to test the application. Or the TE may himself generate the username and password.

##### Severity :

It can be *major, minor* or *critical.*

To analyse the severity of the test case, it depends on the header.

Choose the severity according to module. If in the module, there are many features. In that, even if 1 feature is critical, we claim that test case to be critical. It depends on the feature for which we are writing the test case.

In Gmail,

Compose mail -> Critical Sent items -> Minor

Feedback -> Major

In a banking application, Amount transfer -> Critical Feedback -> Minor

We write severity because we can prioritize our execution based on the severity of the feature.

##### Test case type :

It can be functional test cases or integration test cases or system test case or positive or negative or positive and negative test cases.

##### Brief description :

*Following test case ensures that Amount Transfer feature is not allowing more than balance.*

Test engineer has written test case for a particular feature. If he comes and reads the test cases for a moment, he will not know for what feature has written it. So, this gives a brief description of for what feature test cases are written.

**Step number** is important. If say, step number 10 is failing – we can document defect report and hence prioritize working and also decide if it‟s a critical bug.

***Header format of a Test Case***

|  |
| --- |
| **Test Name :** CBO\_AT\_more than balance |
| **Requirement Number :** 70.3 (in SRS document, amount transfer is numbered) |
| **Project Name :** Citibank Online |
| **Module Name :** Amount Transfer |
| **Severity :** critical (depends on feature we are testing) |
| **Pre-condition : *1)***Test engineer should know the balance amount of user A & B. ***2)***  execution of Balance check test case |
| **Test Case Type :** Functional testing, Integration testing, System testing |
| **Test Data :** Username and password of user A & B, account number |
| **Brief Description :** briefly explains the scenario |

##### Footer of a Test Case :

1. **Author :** Who wrote this test case

##### Reviewed by :

1. **Approved by :**

##### approval date :

##### Test Case Review :

###### Test Case Review process / Peer review process :

Customer gives requirements – development team start developing the product looking at the requirements

* testing team start writing test cases looking at the requirements. Test engineer (you) are writing test cases for a particular module based on the requirements. Once all the possible test cases have been written for that particular module, you send a mail to the Test lead saying that you have finished writing test cases for that module. Now, what the test lead does is – he tells someone in the same testing team to review your test cases. The reviewer reviews all your test cases looking at your module‟s requirements and in case of any mistakes sends it to you and also to your test lead. You correct all the mistakes and send a copy of the corrected test cases both to the test lead and to the reviewer. It need not be that all mistakes pointed out by the reviewer be correct, if you feel they are wrong, then you need to give proper justification as to why your test cases are correct. Once the reviewer says all the test cases are fine, he sends a mail to the test lead saying all the test cases are fine. The test lead then approves your test cases and sends an approval mail to you saying that all the test cases are fine and to start executing the test cases.

While reviewing, the reviewer checks the following,

1. ***Template –*** he checks whether the template is as per decided for the project

###### Header :

* 1. Checks whether all the attributes are captured or not
  2. Checks whether all the attributes in the header are filled or not
  3. Checks whether all the attributes in the header are relevant or not

###### Body :

* 1. Check whether all possible scenarios are covered or not
  2. Check whether the flow of test case is good or not
  3. Check whether the test case design techniques are applied or not
  4. The test cases should be organized in such a way that it should less time to execute
  5. Check whether the test case is simple to understand and execute
  6. Check whether proper navigation steps is written or not

Once test cases are reviewed, the review comments should not be sent via email or in notepad. The ***test case review template*** is shown below,

Review Ethics

1. Always review the content not the author
2. Even after review if there any mistakes both author and reviewer are responsible.
3. While reviewing spend time only in identifying the mistakes not in identifying the solution for it.

***TEST CASE REVIEW TEMPLATE***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Name** | **Step No.** | **Reviewer** | | **Author** | **Comments** |
|  | | **Comments** | **Severity** |  | |
| CBO\_AT\_more than balance | Pre- condition | Pre- condition is missing | Major |  | not fixed. Give justification saying pre- condition is not needed for this |

Q. Why we review the test cases?  
Or  
If I give you few test cases what could be your approach to review? Or  
How do you review the test case?

1. First I will see the requirement for which the TC is written then I will go to the body of the TC and try to find Missing scenarios  
   Repeated scenarios  
   Wrong scenarios
2. Check whether the scenarios are organized properly or not so that it should take less time for execution.
3. Check whether it is simple to understand so the new engineer will be able to execute it without asking the

question

1. I will look into the header of TC and check weather all the attributes are covered or not.

I will check weather all the attributes are having relevant content or not

1. I will check whether TC template is according to standard, defined in the project.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 8 | Click on "Back button" step is  missing | minor |  | fixed |
| CBO\_Insurance\_Age field validation (5 - 55) | 10 | Insufficient test coverage. Apply test case design techniques and derive the following values - 56,  4, 10years,  10.5 | critical |  | fixed |

Reviewer will use the above template and send the comments. If the author fixes the test case, he would report as fixed. If he feels that the test case he has written is correct, he will not fix them – but needs to give proper justification for it.

**Interview Tips**

***In interview, when the interviewer asks “how do you review a test case and what do you review in a test case?”***

***Always answer should start with – Body of test case, then header and finally template. VERY VERY IMPORTANT !!***

***INTERVIEW QUESTIONS***

1. ***Totally, in your current project, how many screens (features) are there ?***

***Ans) an average complex application will have about 60 – 70 screens. A simple application has around 20 – 30 screens. So tell about 60-70 screens.***

1. ***Totally, how many test engineers are there in your current project ?***

***Ans) For 70 screens, 10 – 15screens / engineer. 70/15 = 5engineers. So, you can tell anywhere between 8-10 test engineers.***

1. ***Totally in your current project, how many test cases are there in your current project ?***

***Ans) For 1 screen – you can write 10 – 15test cases (includes FT, IT, ST, positive and negative scenarios) 70 \* 15 = 1050. You can tell anywhere 1100 – 1200 test cases.***

***Test case means 1 entire document with header, footer and many scenarios.***

***Interview questions (continued .. )***

###### Totally, how many test cases you have written in your current project ?

***Ans) This includes for all 3releases. You have joined in 3rd release. In 2releases, they would have written around 700test cases(worst case – 650test cases). You wrote in 3rd release test cases, so 550/5 = 110. You can tell anywhere between 80 -110 test cases (maximum of 240 also)***

###### How many test cases can you write per day ? Ans) you can tell anywhere between 3 -5test cases. 1test case – 1st day, 2nd day

***2test cases – 3rd day, 4th day 4test cases – 5th day***

###### 8 – 9test cases – 18th day Always answer like this,

***“initially, I used to write 3-5test cases. But, later stages, I started writing 7 – 8test cases because,***

###### Knowledge about the product became better

* ***I started re-using the test cases (copy and paste)***

###### Experience on the product

***Each test case that I write would generally have 20 -40 steps.***

###### How many test cases can you review per day ?

***Ans ) around 8test cases you write, so you can review 8\*3 = 24 test cases. Tell 24- 30 test cases per day***

###### How many test cases can you execute per day ?

***Ans) Around 35 – 60 test cases you can execute per day.***

***For 6th, 7th and 8th question – always remember this ratio : „x‟ test cases you can write, „3x‟ test cases you can review and „5x‟ test cases you can execute per day.***

**Procedure to write the test cases :-**

System study

Identify all possible test scenarios

Write test cases by applying test case design techniques, using standard template

Review test cases given to you for reviewing

Fix the review comments of your test cases given by the reviewer

Test Case approval

Store it in test case repository

*System study –* understand the application by looking at the requirements or SRS given by the customer.

*Identify all scenarios :*

1. When the product is launched – what are the possible ways the end user may use the s/w. Identify all possible ways
2. What and all possible business flows are there
3. Document all possible scenarios in a document / paper – it is called test design/high level design.

Test design is a record having all the possible scenarios.

1. Brainstorming session
2. Measure the efficiency of brainstorming session

*Write test cases :*

1. Convert all the identified scenarios to test cases
   * Group scenarios related to 1feature
   * Prioritize
   * Write test cases
2. When converting, apply test case design techniques
3. Use standard test case template – standard means the one decided for the project.

##### Test Case Repository : (TCR) QA

|  |  |
| --- | --- |
| **CBO\_Testing** | |
| …… …… …..  ….. …… ……  …… ……. …..  ….. ……. …..  ….. ……. ……  **LOANS INSURANCE AMOUNT TRANSFER** |  |
|  | |

TCR is a shared folder. For security, they keep the entire folder in version control tool(VCT).

Customer gives requirements, developers are developing features and test engineers are writing test cases looking at the requirements. The test cases which are approved by the test lead are stored in a test case repository. When test cases are needed for execution, then test engineers will *check in* and retrieve their respective test cases. After execution, the test engineers then *check out* of the TCR. When any test cases are not needed, it is *dropped* from the TCR.

Always, the testing team keep taking back-up of the entire TCR folder to prevent any crashes from affecting the project.

QC(quality center) is used to store all test cases or the test cases might be stored in Test Link which is a test management tool.

##### Procedure to execute test cases :

Customer gives requirements – developers are developing the features looking at the requirements. The test lead gives a list of features for each TE to write test cases and execute them. The TEs first understand the product by looking at the requirements and then start writing test cases.

Now, let us consider that TE1 has been given loans feature, TE2 has been given insurance feature and TE3 has been given Amount Transfer feature. Now, all these TEs start writing test cases for their respective features. After the test cases have been written, reviewed and approved – they are stored in the test case repository.

Now, by this time – developers have given the 1st build – in 1st build, features which have been developed are – Personal loans(half), life insurance(half) and Amount balance. Whenever developer gives builds – create another folder (Tiger\_Test Execution Result) – create in that folder B01(1st build) – copy,paste all test cases from QA – Build1 comes – remove all test cases for the features which are not yet developed – keep only those test cases for which the features have been developed i.e, the relevant test cases – then do smoke

testing and all other testing and then fill in the Status columns of the Test Cases – bugs are caught and sent to the development team.

After this, developers give the 2nd build B02 – here, the features developed are – complete personal loan, complete amount transfer and also complete life insurance – create a folder named B02 – copy, paste the entire test cases in that folder – and the same method as in B01 follows.

Same procedure follows all Builds. Thus we have all the results and can check the results for all the builds. Build comes in – execute all relevant test cases – by the end of the cycle, summary report must be filled. This summary report is known as ***Test Execution Report / Test Summary Report***.

***TEST EXECUTION REPORT (Build1)***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name** | **Total Test**  **Cases** | **Total**  **Executed** | **Total**  **Pass** | **Total**  **Fail** | **%Pass** | **%Fail** |
| Loans | 420 | 120 | 96 | 24 | 80% | 20% |
| Insurance | 500 | 200 | 180 | 20 | 90% | 10% |
| Amount  Transfer | 400 | 100 | 95 | 5 | 95% | 5% |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Total** | **1320** | **420** | **371** | **49** | **88%** | **12%** |

This report is called ,

***Tiger – B01 – Test Execution Report / Test Summary Report***

Test Lead prepares this report. The TE sends the individual features which he has tested and tells how many he has executed and all that statistics.

The Test Lead then sends the report to,

* Test Manager
* Development Team
* Management
* Customer (depends on whether the project is a **„fixed bid‟** or „**time & material bid**‟.

If it is a „time and material‟ bid, then the test execution report must be sent to the customer as well.

The development team needs,

* A list of test cases that are failed
* Each developer needs a list of test cases which are failed for his features.

In each sheet is a list of test case names and everything as shown below,

***LOANS feature***

|  |  |  |  |
| --- | --- | --- | --- |
| **Step**  **No.** | **Test Case Name** | **Status** | **Comments** |
| 1 | ….. | Pass | …. |
| 2 | ….. | Pass | …. |
| 3 | …. | Fail | …. |
| 4 | …… | Fail | …. |
| 5 | …… | Fail | …. |
| 6 | ….. | Not  Executed | …. |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | …. | Not  Executed | …. |
| ….. | …. | …. | …. |
| …. | …. | …. | …. |
| … | … | … | …. |
| 420 | … | …. | …. |

The developer knows only the test case name. then he directly goes to the test case through a link or directly and then sees which test case is failed.

The Test Execution Report is stored in the B01 (respective builds) outside all folders. For compatibility testing, the Test Execution Report looks something like this,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Windows XP** | | **Windows 7** | | **Windows Vista** | |
| **Step**  **No.** | **Test Case Name** | **Status** | **Comments** | **Status** | **Comments** | **Status** | **Comments** |
| 1 | ….. | Pass | … | Pass | … | Pass | …. |
| 2 | …. | Pass | …. | Pass | …. | Pass | … |
| 3 | … | Pass | … | Pass | … | Pass | … |
| … | … | … | … | … | … | … | … |
| … | …. | … | … | … | … | … | … |
| .. | …. | … | … | … | …. | … | … |
| 7 | … | Fail | … | … | … | Pass | … |
| … | …. | Fail | … | … | … | Pass | … |
| 237 | … | Fail | … | … | … | Fail | … |

Again, we can do compatibility testing for browsers on various platforms,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***WINDOWS XP*** | | | | | |
| **Internet Explorer** | | **Mozilla FireFox** | | **Opera** | |
| **Status** | **Comments** | **Status** | **Comments** | **Status** | **Comments** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |