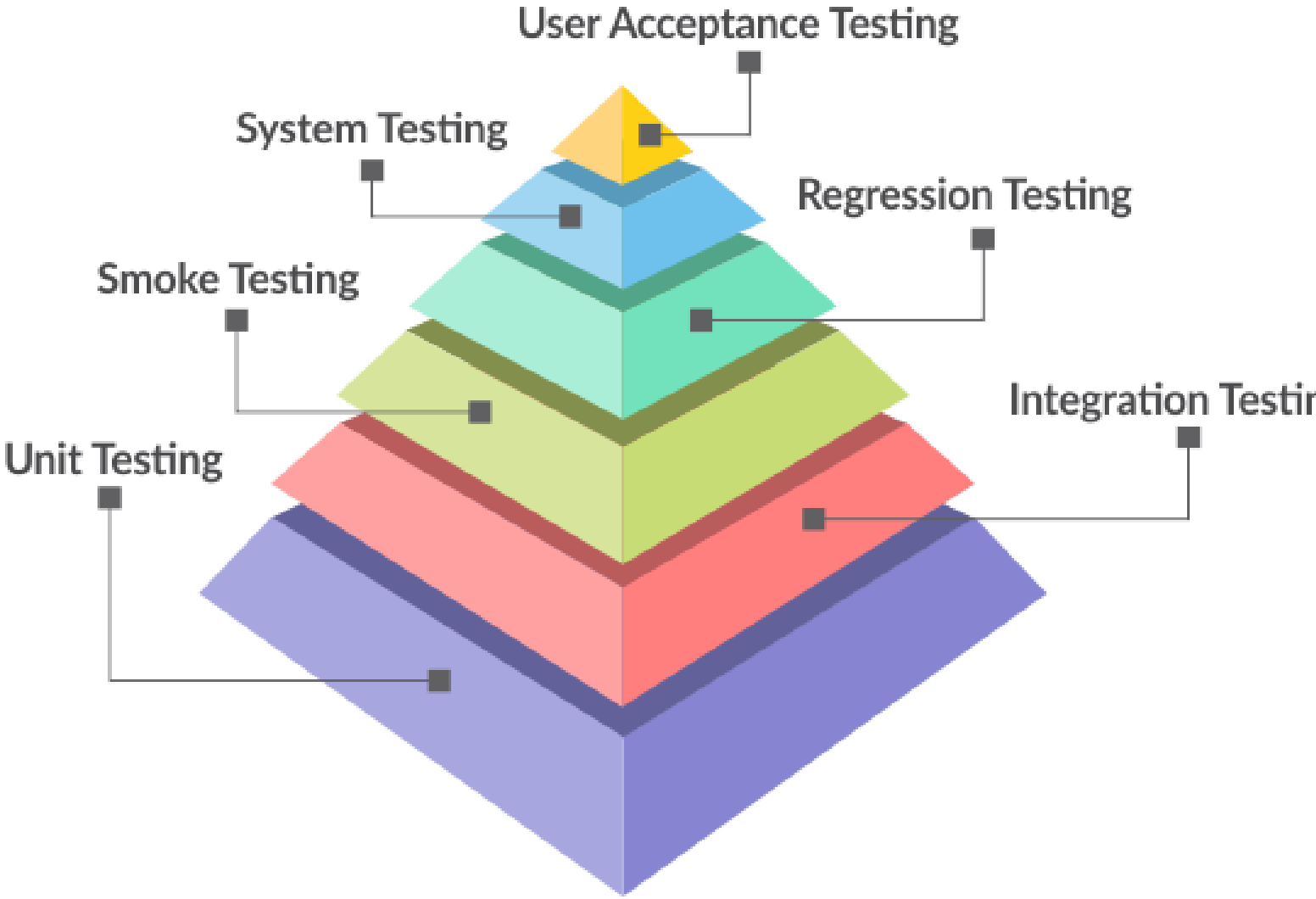
Types of Testing
+
Bug Life Cycle
+
SDLC



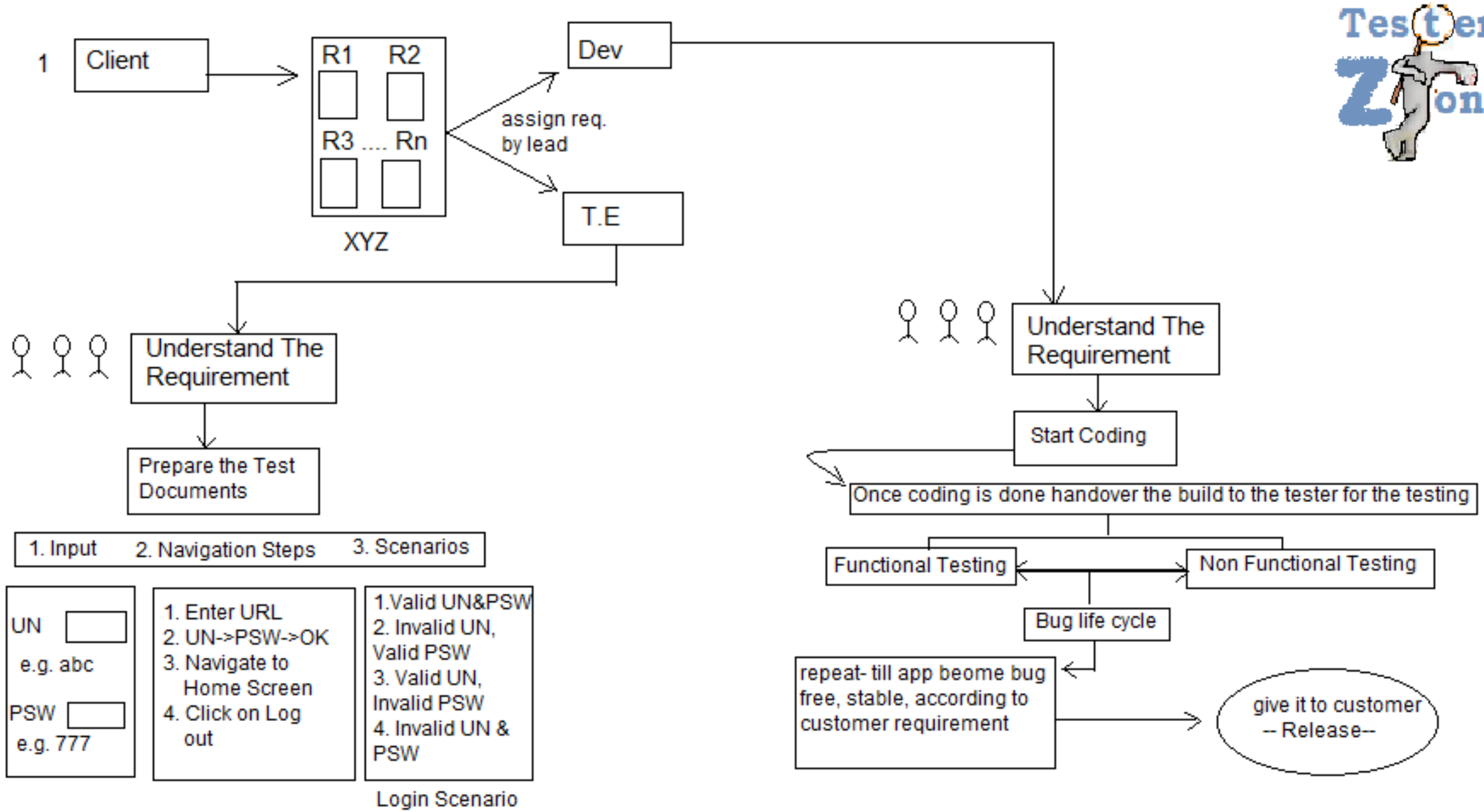
By Mithilesh Singh

Key topics in Manual Testing

- Testing Flow in IT
- Definition >>Testing
- Type of Testing
  - FunctionlaityTesting
    - Functional Testing
    - Integration Testing
    - System Testing
    - User Acceptance Testing
    - Exploratory testing
    - Sanity Testing
    - Smoke Testing
    - Regression Testing
  - Non Functionla Testing
    - Usability Testing
    - Performance Testing
      - Load Testing,
      - Stree Testing
      - Scalability Testing
      - Volume Tetsing
    - Compatibility Testing
    - Localization/Globalization Testing

Adhoc/Monkey Testing
severity/priority
Testcase/Test Scenario/Test Document
Test Case design Template
RTM- Requirement tracibility matrix
SDLC-Software development life cycle
Build process
Interview Questions

Bug Defect Life Cycle.
Bug Defect Template
Type of Applications
Test case Review Process
Alpha- Beta testing
STLC-Software testing life cycle
Resume preparation



1. Testing Flow in IT

"Testing Flow in IT" Explanation

- Client gives the requirement to the BA(Business Analysis) in form of CRS(Customer requirement Specification).
- BA converts the requirement in SRS(Software requirement specification).
  Note: Since CRS is high level notes, BA creates SRS so that technical people can understand it clearly.
- Lead breaks the requirement into multiple chunks and allocate to different testers and developers.
- Both(Dev and Tester) start analyzing the requirement first.
- Developers start writing code based on assigned requirements.
- Teters start writing test cases for assigned requirements so that they can do testing based on that.
- Once the development done, developer handover the code to Build team to compress it in build and handover that build to the testers for testing.
- Testers start testing the build based on test case and if functionality differs, testers file bug and share with developer.
- Developer go through the bug and try to fix it soon and after fixing gives a fresh build to tester with updated code for the testing.
- Testers re-test the build and if it works as per business need, we sign off for production.

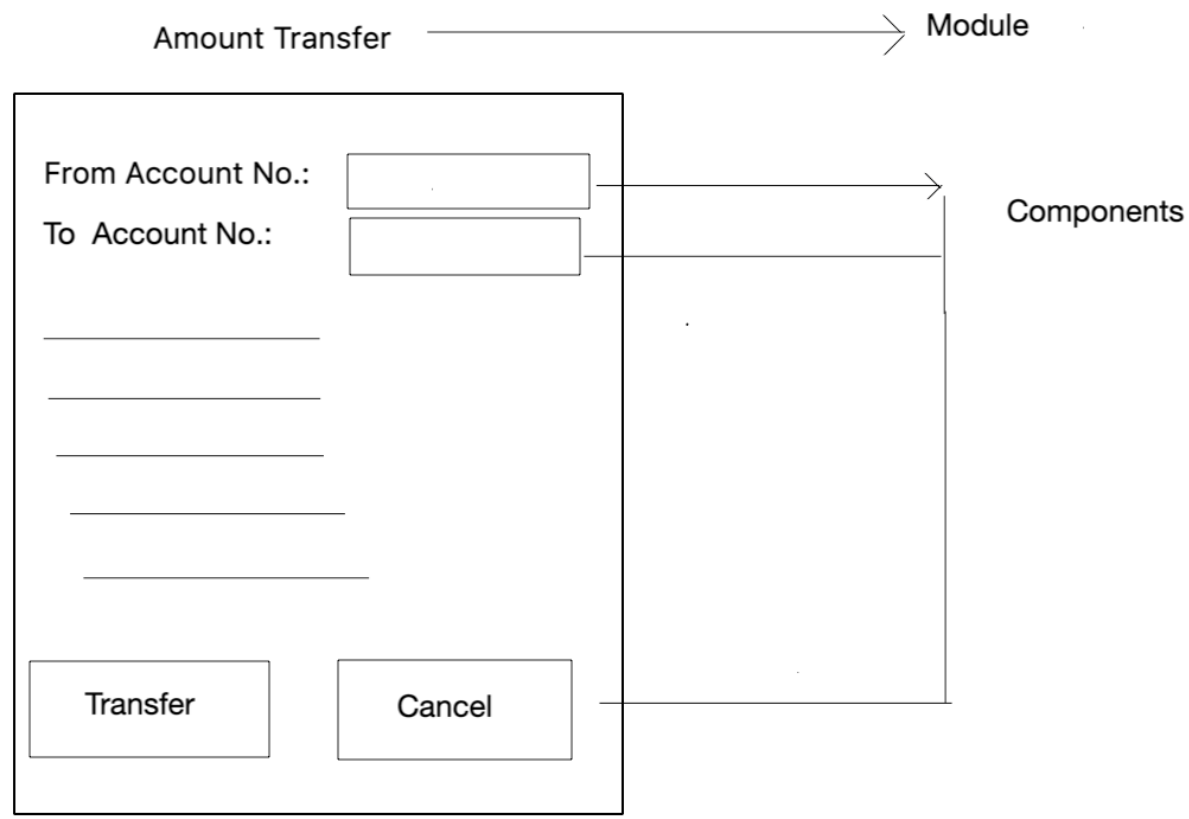
Functional Testing

- Functional Testing is a type of testing where we test the components of a module of an application independently.
- We always start with functional testing in the company.
- I am taking one real time example to explore the functional testing flow. E.g. every day we do transfer amount to other's right? Let's take this scenario only.

Real time Example

Customer Requirement

- 1 Amount Transfer
- 1.1 From Account No--> Text box
- 1.1.1 should Accept 4 digits
- 1.2 To Account No--> Text box
- 1.2.1. Should Accept 4 digits
- 1.3 Transfer--> Button
- 1.3.1 Transfer--> Enable
- 1.4 Cancel--> Button
- 1.4.1 Cancel --> Enable



Requirement Explanation

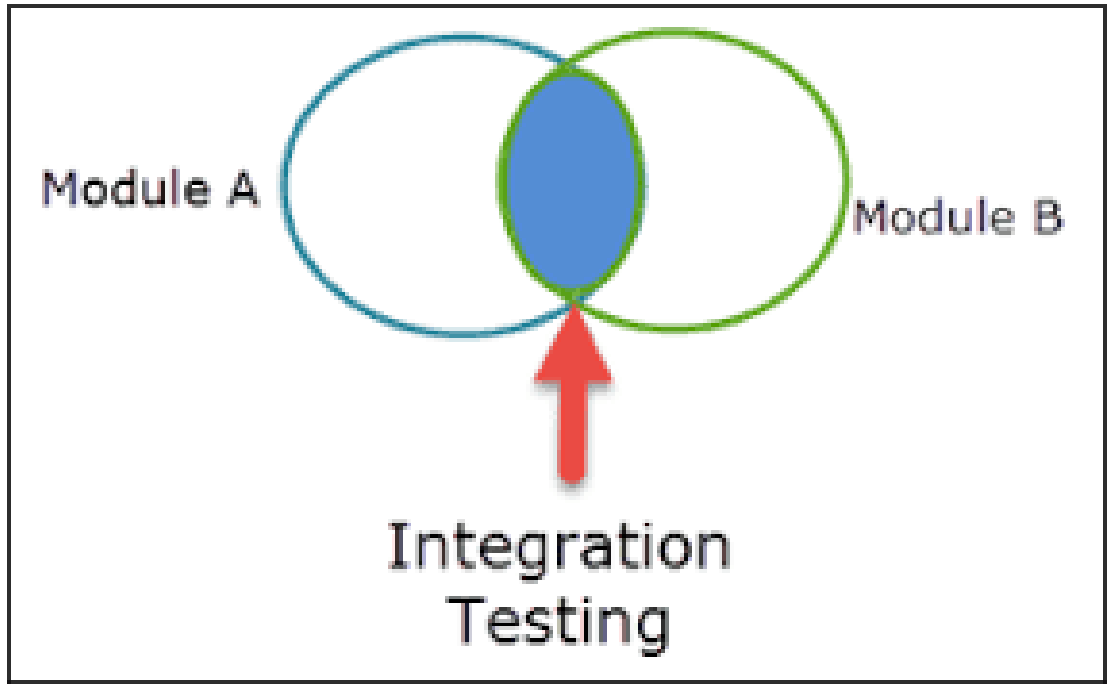
Client has given requirement of an amount transfer module which should contain "from and to account holder number field", "amount", "Transfer and Cancel" button.

Amount transfer will be module and inside this From/To account no, transfer/Cancel buttons would be submodules/components/sub-features.

Note: Focus on the Requirement numbering. First module number, then submodule and finally description of sub module.

Integration Testing

- As soon as we get application we start checking each and every modules independently one by one, once all the modules are stable we start checking the relationship between the dependent modules.
  Checking dependency between the modules is known as "**Integration testing**".

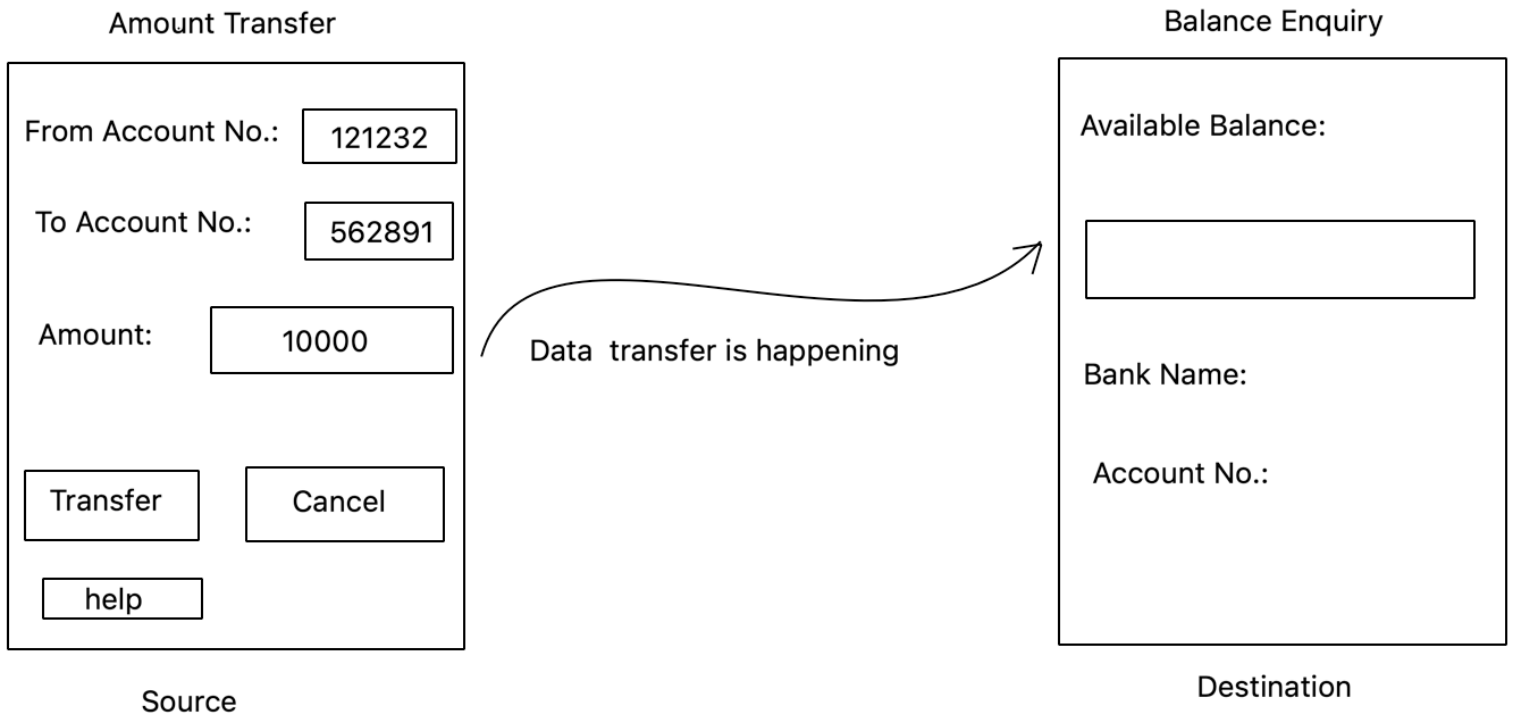


Rules of Integration Testing

- To perform integration testing or we go for Integration Testing once Functional Testing is done.
- To perform integration testing very good product knowledge is require. Since multiple testers will be working in the same project.
- Integration Testing is done only between the dependent modules i.e. there are some modules in the application where integration testing is not mandatory
  ex. Help module in Gmail Application.
- In Integration Testing Source, destination and data is require.

Rules of Integration Testing

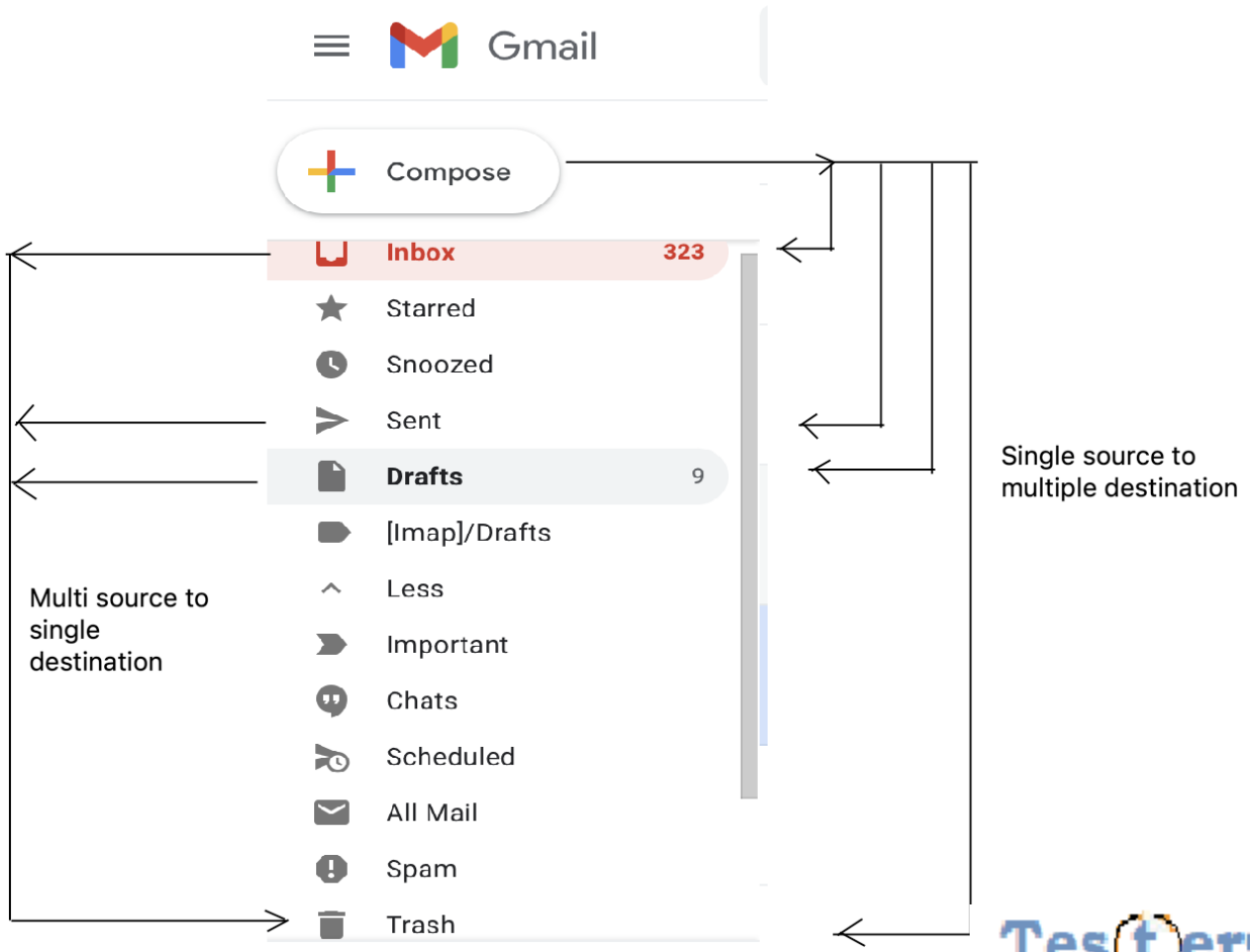
The data Transfer from source to destination and gets saved in the destination



In Integration approach data should transfer between the modules, in this snap amount will be transffered and reflected into Balance enquiry module so we can say integration is possible between these two modules.

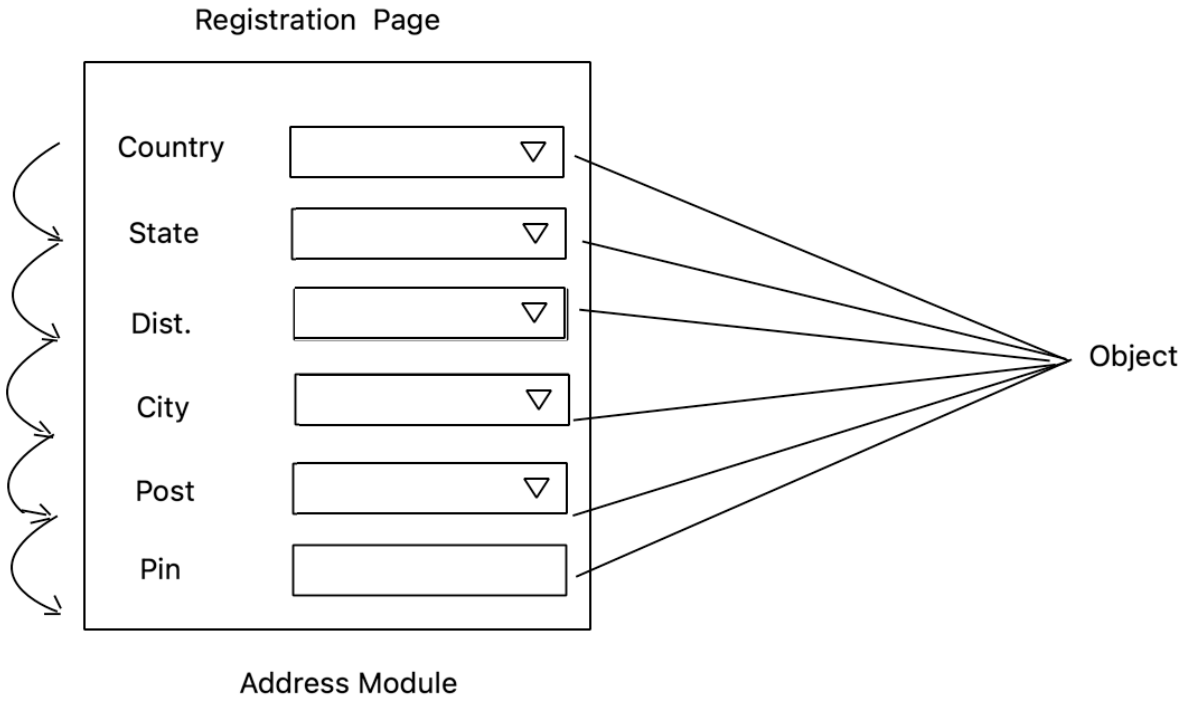
Rules of Integration Testing

Some time data transfer from single
source to multiple destinatio n whereas
vice versa is also possible



Rules of Integration Testing

Integration Testing is done within the same module also i.e. data may transfer within a module from one object to another.



This is an example of an address module/feature availables in any registration page. Whenever we select the country, all states for that country will be listed, based on selected state dist. Will appear and flow goes on so in this case we can say that one sub feature is dependent on other so hence we can perform integration testing in the same module as well

Types of Integration Testing

- 1. Incremental Integration Testing
  a. Top-Down Incremental Integration Testing
  b. Bottom-up Incremental Integration Testing
- 2. Non Incremental Integration Testing

Incremental Integration Testing

In Incremental Integration Testing approach there should be clear relationship between the modules. There are two approaches under this.

a. Top- down Approach: We add the modules incrementally from top to the bottom and check the data flow in the same order.

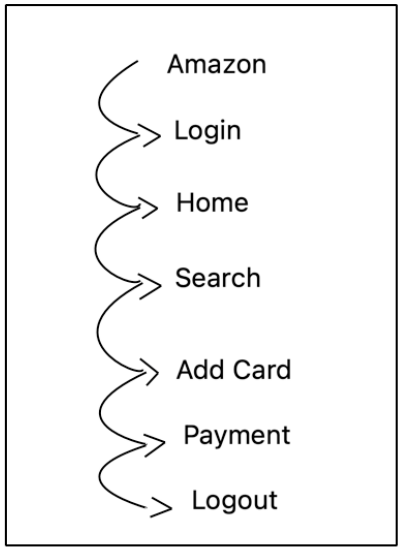
b. Bottom- up Approach: In this approach modules are added incrementally or one by one from bottom to the top and data flow is checked in the same

Non-Incremental Integration Testing

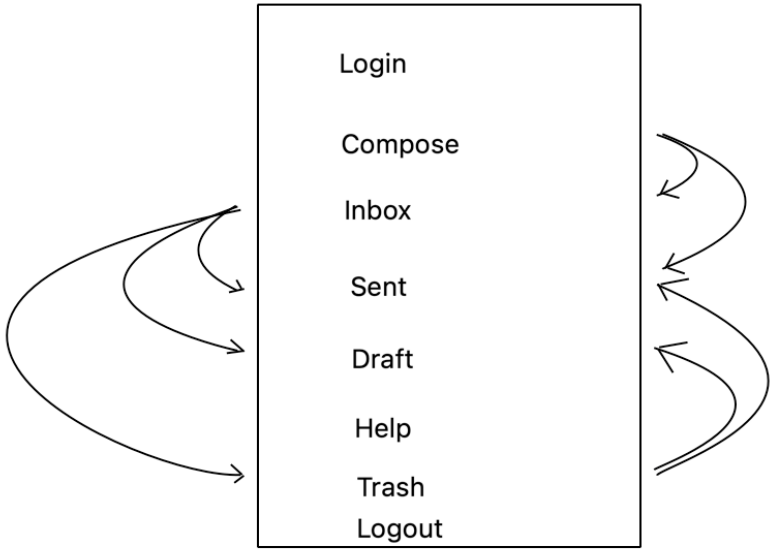
This Testing Approach we follow whenever there is no clear relationship between the modules.

In such case we create the data in one module, bang on all other existing module and check for the existance of the data , it is also known as big bang method.

Flow diagram Explanation:



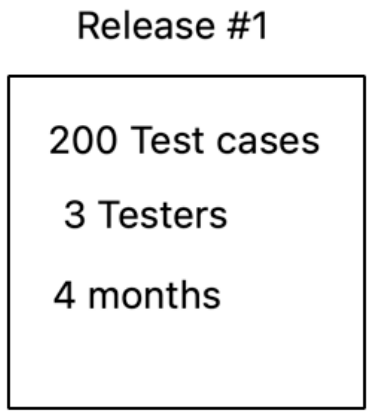
Flow Diagrame of Top-Down Incremental Integration Testing

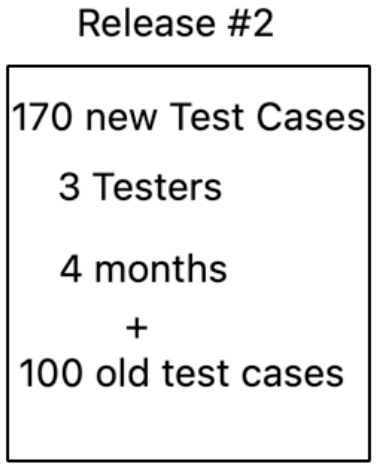


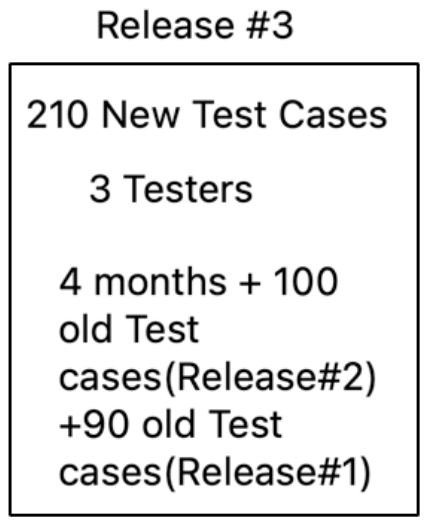
This is an example of Top down and Bottom-up approach collectively —> Big Bang Approach.

Regression Testing

- Whenever developer does some code changes, because of bug fixes or the enhancement request across the builds or across the releases these changes may have some impact area, checking these impact area or affected area is called as "Regression Testing".







Explaination: In the above diagram you can observe that there are 3 releases for a product. in first release we had 5 features in the application, wrote 200 test cases for those modules, assigned 3 testers and time allocated for this is around 4 months.
In Release 2 we had planed to add few more modules, now the point is those modules might or might not be dependent on older modules. in this case we are assuming those modules are dependent to each other so in release 2 we have total 8 modules. so we have 170 cases approx. for new 3 modules and as i said there is dependency between the modules, older modules also needs to be checked. for older modules we would have test cases which we can easily collect from the release 1 repository. suppose old cases are around 100. so toal 8 modules, 270 cases and 7 testers. like wise case count, module and testers count will be increasing, but in this case drawback is like how much resourse we should waste due to this repetative task??
So these were few challenges in regression testing. let's talk about this in next slide.

Key Points :

- Test cases will be increasing in the regression testing due to addition of new modules relase wise.
- Every Release will have their own test case repository.
- Test cases only increase when there will be dependency between privious modules and new modules.
- Regression testing is not mandatory in all the releases . It depends upton the dependency of the modules.

What is Regression Test Cases

- Test Cases which are picked from privious released product repository based on dependency between the modules is known as Regression Test Cases.
- Note : To pick the regression test cases we should have very good knowledge of product and also need to aware with dependency between the modules(area of impact).

When should start regression??

- Generally Regression Testing starts from release no 2. because in first release everything will be new.

  Note: It does not mean that we not at all do regression testing in first release. We can not go with release level regression testing but we do build level regression testing in every release(even in very first release also).

  we might have multiple builds in single release and every builds most of the time contains few bug fixes. To retest those fixes we re-

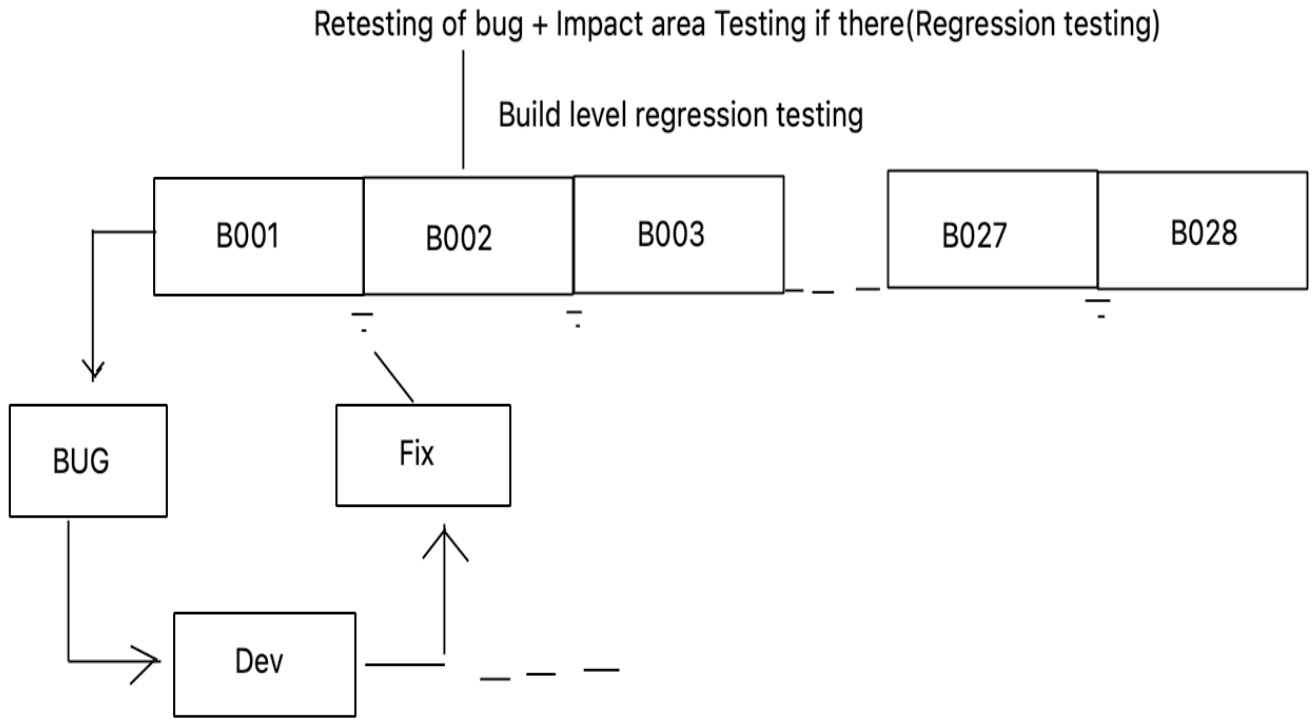
- execute some scenarios which is again repetative, that comes under build level regression testing.

  Note: if there is some modification in older test cases based on enhanced older feature, those modified test cases will not consider as part of regression even it was already written.

Regression Process

- In Old Release or Release 1 we don't do regression Testing. It always start with release 2.
- As soon as customer give the requirement for the new features. We(Customer, Test Engineer and Developer) will analyse the impact area.
- Customer , Tester and developer all three will create their own report which will be based on their product knowledge and few assumption.
  Customer will analyse based on business knowledge
  Developer will analyse based on product knowledge.
  Testers will analyse based on product knowledge
- All three reports will handover to the lead.
- Test lead will consolidate it and pick the regression test cases from the different releases and store it in the regression suite.
- When developer and testers will hand over the reports at the same time they will start working in new features
  Testers may be explore the requirement/write the test cases.
  Developer will be fixing the bugs(new feature bugs)
- Soon Testers and developers will complete their work in new features. They will be assigning with the regression casses
- Developer and testers will be working on impact areas .

Build Level Regression Testing



Explanation: As we can observe in the diagram suppose we have multiple builds through out the SDLC process. First of all developer will implement some part of application and hand over to the testers for testing. Suppose testers found some bugs, testers will report to the developer and developer will open that bug and fix it. Once developer will fix the bug again he has to provide a bug free application to the testers. That is why developer creates one new build with bug fixes and give it to the testers. Now testers will retest the bug fix and if there is any impact area then test that part also. This process of testing impact area is known as Build level regression testing.

Note: new build does not comes after bug fixes, it might contain some feature level changes also

Release Level Regression Testing

- In first release we do build level regression Testing.
- In Release 2 we will do build level regression testing. First for new features or modules then we will re-execute the regression test cases.
- Release level Regression Testing is always formal way of regression testing because Testers, Developer, Test lead and client all involves in it.

Challenges of Regression Testing

- Time consuming Process due to
  a. Test Cases increase every Release
  b. Less Resources
- No accurecy in testing due to repetitive task.

  Note: To identify the impact area is also a kind of challenge

Solution against the regression Challenges i.e. Automation

- Testing the application by using some tool is known as automation testing.
- There are many automation testing tool like QTP,SilkTest, Selenium etc
- Generally we will go for automation testing only when there are multiple release or multiple regressional cycles or when there is a repetative task that needs to be performed.
- In the first release of the project there is no concept is automation testing since we don't do regression.
- From the next release the manual and automation team start working together i.e. the manual team understand the new set of requirements also analyse the impact are, prepare regression test suite and hand it over to the automation team.
- Once Regression test suite is ready the automation team start writing the test script the manual team and development team is completely busy in working on the new features.
- By the time all the new features are stable the automation team could have completed writing test script for regression cases.
- The automation team starts regression testing while executing the test script on the new application for the regression features(old features)
- Based on the results if any script fails then the manual team varifies the functionality and if bug exist they prepare a bug report and send it to developer team.
- Once the bug is fixed the automation team re-execute the script which are fail and the manual team retest the bug along with impact area if necessary
- Those process continuous untill all the new features as well as regression features are stable this is how the perform the automation testing or regression testing.

Smoke Testing

- As soon as we receive build to test the application, Testers always start with functional testing but it takes more time to encounter blocker bug due to which time duration of project can be increased and also possibilities to sit idle by tester during the project will more.
- To avoide above scenario we also go with kind of testing is known as Smoke Testing.
- With the help of smoke testing we can find Blocker bugs in the initial stage so the dev will have more time to fix it.

- " **Checking the basic and critical features of an application or a build before going for one round of deep testing is known as Smoke Testing"**

  Or

  Testing the features with high level input(only one set of positive input) is known as smoke testing.

  Note: There is difference between Testable and stable build.
  Testable means smoke is done blocker bugs will not be there but some other bugs might be occur.
  Stable means we have done with testing and now it is ready for release.

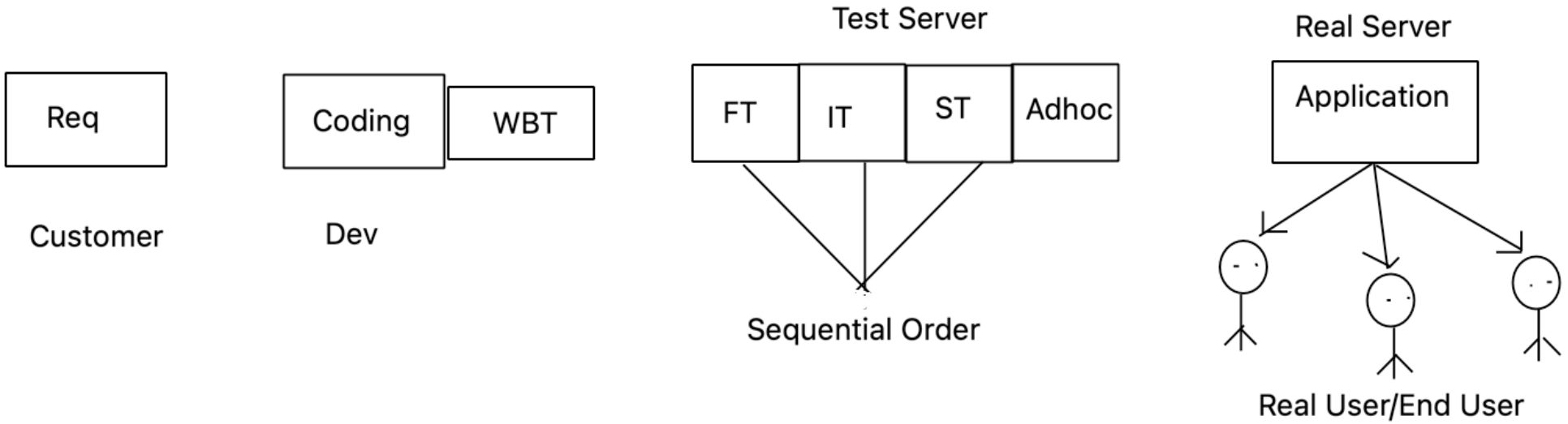
What is BVT?

- BVT stands for Build verification testing.
- Smoke testing is known as Build verification testing.
- In Smoke testing we verify , if each and every build is testable or not hence it's also known as build verification testing.

  E.g. If I have gmail application and need to perform smoke testing then I will try to send one mail to me and to other account and check mail is deleivered or not, and then check the draft and trash module functionality. I will not check help and settings functionality because that is not much critical modules. Sending mail is critical feature because that is something for which app is build so that should be work.

Adhoc Testing/Monkey Testing

- Testing the application without any rules and procedure or testing the application randomly iis known as Adhoc Tesing .



Why and When Adhoc Testing?

- Since User/Customer don't know how to use the application so they might use randomly and found some bug. To avoid this we prefer adhoc testing.
- We always prefer adhoc testing after performing the functional Integration and system testing because these are the sequential order testing and followed by many of the users. Many few customer face problem during random use of application that's the reason we always use adhoc testing at the end only if time permits.
- Adhoc testing is called as monkey testing because as monkey don't use proper land to walk, they always jump from one place to another improper way, adhoc testing also don't have any rules we simply perform it as we want so due to this random process of testing only it is known as "Monkey testing".

Example of Adhoc Testing

- Suppose we are using Any banking application and generally people do transfer amount using following steps:
  1. Enter your account details
  2. Enter sender accound details
  3. Enter amount detail and click on Transfer and check the balance.

  but I want to do adhoc testing and we know adhoc is random flow, I am following step no 1 and 2 and at 3rd step I am trying to click transfer button 10 times and somehow it allows me to do so without any error pop up and at the end money got transfferd 10 times to user as well.

  Note: since this flow was random but it is a kind of bug in app right? So in this way we can find some unexpected bugs through adhoc testing.

Performance Testing

- Checking the behaviour of an application by applying load is known as performance testing
- In performance Testing we check the response time of app, load on the application and stability of an application.
- Load: It talks about no. Of users using the application at a time.
- Response Time: Time taken by the server to responde to the client request.
- Stability: It talks about the time duration till when the application can sustain in a desire load.
- Performance Testing can not be done manually, we use some tools like Jmeter, Loadrunner, webstress etc.

Key Points:

- Once the application is stable in the company and move it to the product server, n no of users may access the application simultaneously and during the process they may encounter some performance issues to avoid those issues we do one round of Performance Testing.
- Performance Testing generally done once after application is functionally stable.

Performance Testing Process

- Manual Test engineer will identify the scenarios which is needs to be checked in performance testing on the basis of few factors.
- Most frequently used scenarios
- Critical feature(main feature of an application)
- Huge Data Load:
- Performance test engineer will write the script for identified scenarios.Now devide the load using the "usage Pattern"
  Note: Usage Pattern means no of users going to use module.
- Run the test or execute the script
- We find the result in form of response time, we can take consider average response time and verify with expected result..
  Note: in performance testing we also use vitual user because they don't exist physically.
- If Result does not meet the goal then analyse the issue(Bottle neck), solve it and re-execute the proces.

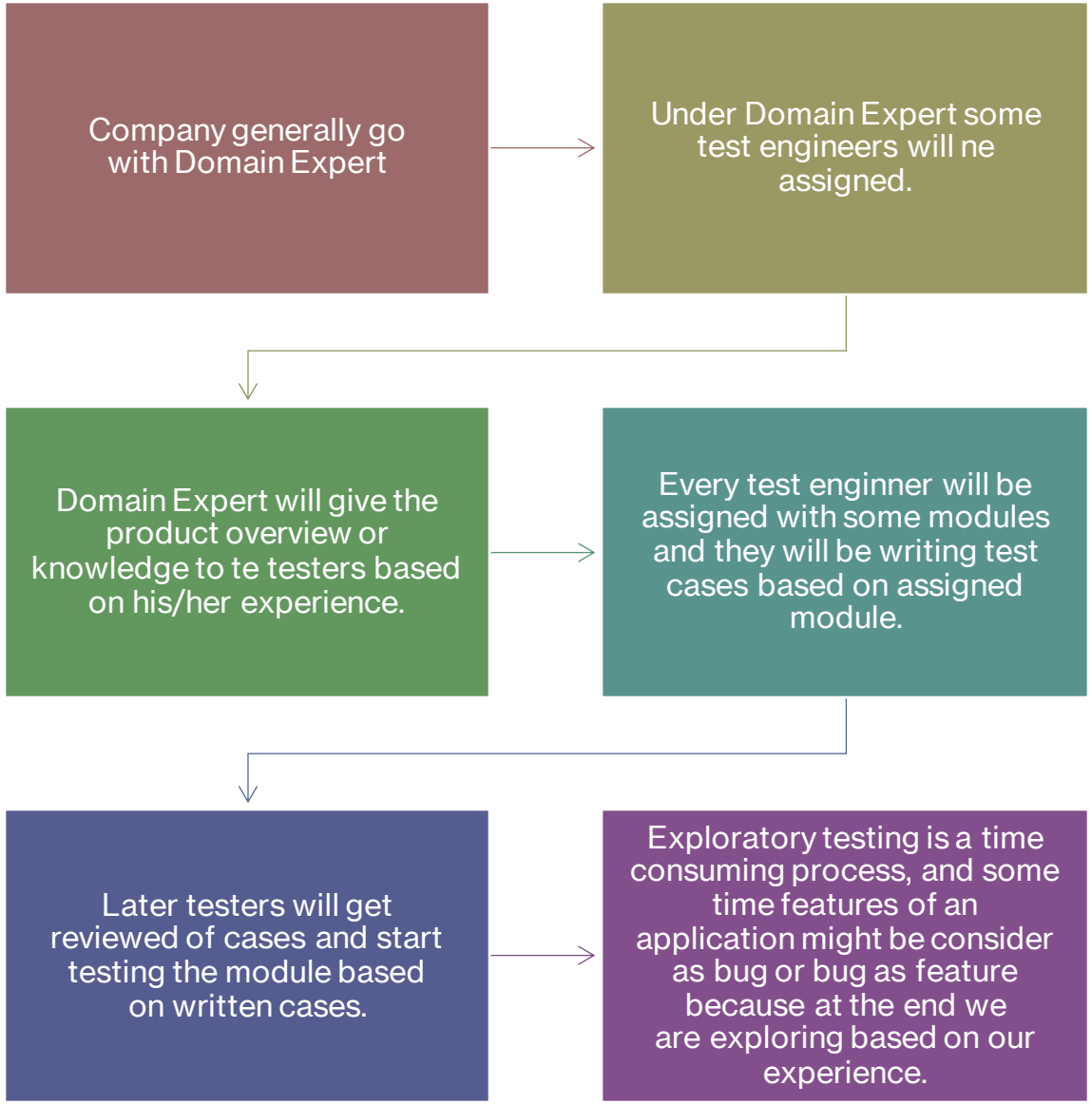
Types of Performance Testing

- **Load Testing:** Checking the behaviour of an application by applying load less then or equal to desire load is known as load testing
- **Stability Testing**: Checking the behaviour of an application by applying load for a long interval of time is called stability testing.
- **Stress Testing**: Checking the behaviour of an application by applying load more than desire load
- **Scalability Testing:** Checking the behaviour of an application by increasing or decreasing the load in a particular scale for a certain period of time.

Exploratory Testing

- Though it is very straight forward topic people use to merge it with Adhoc testing concept,
- Exploratory Testing always done when there is "no requirement or requirement is not clear"
- No requirement does not mean product specific requirement will not be available, through out testing phase we can do this exploratory testing any time. It is just to explore the feature and add the cases. So when we say explore means that feature or subfeature related requirement might not be available or if available also but not up to mark. So as a tester we can explore and add.
  I hope I am clear with my statement..
- Company also prefer experience guy or domain expert to do this kind of testing

Exploratory Testing Process



Usability Testing

- Checking the user friendliness of an application known as usability Testing.
- User Friendly means:
  1. Application should be easy to access, all the important features should be exposed to the user first as well as highlighted so that they can easily find the feature they want.
  2. It should be easy to use i.e. the navigation steps to perform any action must be as less as possible.
  3. Easy to understand i.e. all the terminologies or the features name must be use in simple languages so that any person can easily understand.
  4. Look and feel of an application must be good i.e. the font sizes and the colour used should be in particular standard.

Usability Checklist

- Every page should home page link
- Every page logout link
- All tags must exist
- Pagination exist
- Bread Crumbs exist

.....

.....

400+ usability checklist.....

Process of Usability Testing

- In Usability testing Process, we identify the need of usability testing.
- If there is no need of UT then no need to proced but if yes then derived all the usability testing checklist.
- After making checklist review and approved by customer
- Finally handover this checklist to developer to implemented this checklist
- After that developer handed over the ready application TE.
- Test engineers execute the checklist on the ready application
- At the end prepare usability test report with other other documental prrofs and handover it to customer with stable application.

Dashboard | Liquid CRM | Monthly Quality Checks - Goo | Tahoe Pet Station | Mobile Usability | Small Animal Treats — Tahoe

tahoepetstation.com/collections/small-animal-treats

Apps | Welcome to CircNet | New Media Retailer | Circnet | Highrise: Latest A... | Liquid CRM | PassPack | Basecamp 3 | Shopify Partner Pr... | Other Bookmarks

PET STATION

What are you looking for?

Usability issue(overlapping)

$4.99 FLAT RATE SHIPPING
Average Delivery in 3-5 Days

LOGIN

DOG | CAT | REPTILE | AQUATICS | SMALL ANIMAL | LARGE ANIMAL | BIRD | LOCATIONS | MORE INFO

PRODUCT CATEGORIES

- DOG
- CAT
- REPTILE
- AQUATICS
- SMALL-ANIMAL
  - *ALL IN SMALL-ANIMAL*
  - ENRICHMENT
  - HAY
  - LITTER & BEDDING
  - FOOD FOR SMALL ANIMALS
  - TREATS FOR SMALL ANIMALS
  - HIDE OUTS

Home › Small Animal Treats

SMALL ANIMAL TREATS

Pets love receiving treats, but they are not all created equal. Dr. Micah Kohles of Oxbow Animal Health discusses the best choices when it comes to treats, and offers tips on how to use treats to add valuable enrichment in your pet's routine.

Waiting for sessions.bugsnag.com...

Compatibility Testing

- Checking the behaviour of an application on different platform(Combination of software and hardware) is known as Compatibility Testing.

  Note: Software indicates the different Operating systems and browsers, hardware talks about different harddisk, RAM/ROM size

  Note: When the application is handed over to the customer they may use it from different different platform and during this process they may encountered some compatibility issues, to avoid such kind of issues we need to do one round of compatibility testing.

Compatibility Testing Process

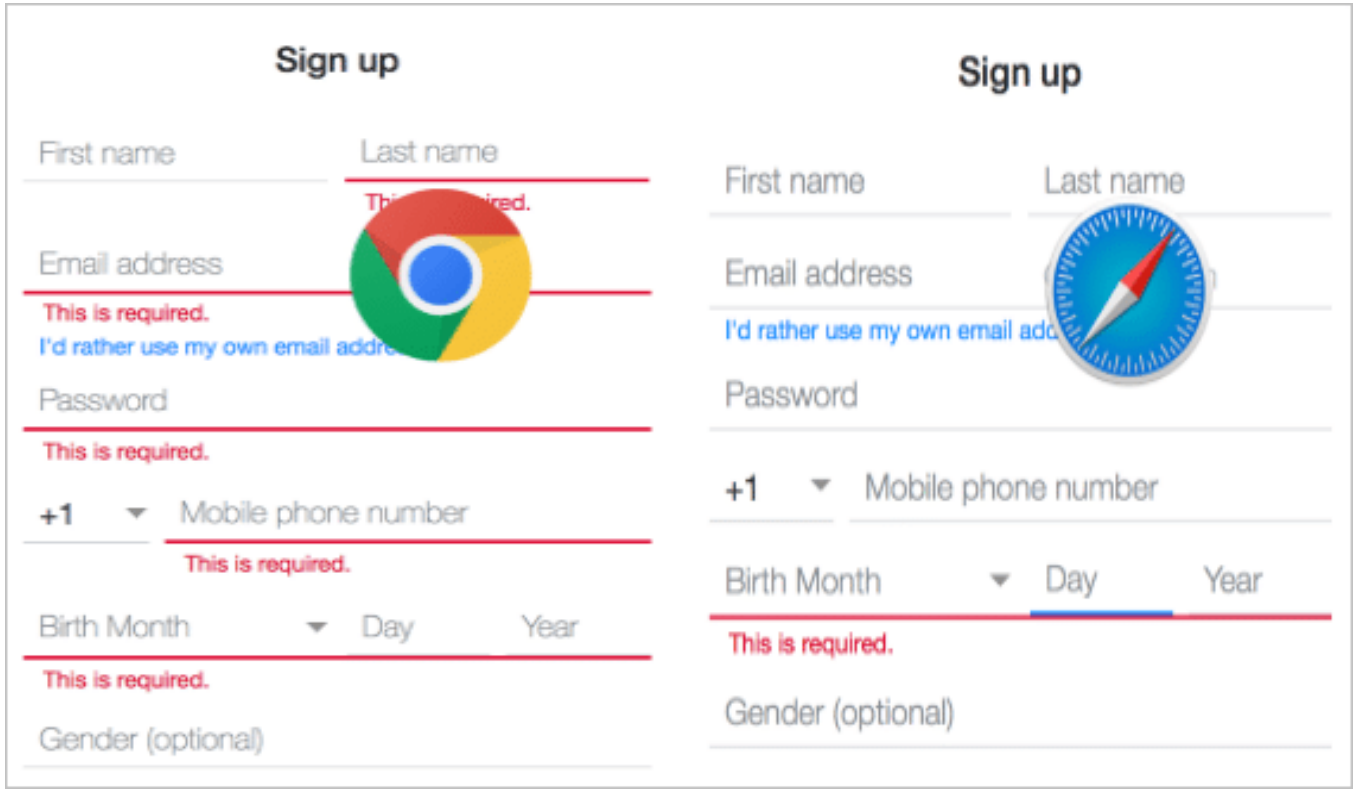
- First of all both functional and Non functional requirement is given by user
- Base platform is installed in all the testers where they will do all type of testing.
- All the testers will do one round of functional testing on their assign module and to make application functionality stable.
- Admin create a 'Virtual Machine'. VM set up is nothing but a server where all necessary platform installed.
- To access multiple OS simultaneously, a software is also needed which is known as VMWare.
- Aftrer completing the functionality testing, All Test engineers do one round of testing on necessary platform which is access remotely by them using their own credential data(User id and Password).
- Once testers connect through different- different platforms, we need to open it and only check end to end flow of an application.
- Application should work on the necessary platforms.

This is nothing but compatibility Testing process or Software Compatibilty Process.

Note: Different browser base compatibility testing can be done in the local system itself so no need of any software like VMWare. But same browser with different version can be tested wih VMware software.

What is Compatibility bug?

- Whenever something is happening in one platform but the same functionality is not working or content is not appearing in another platform(OS, Browser) such kind of issues are known as Compatibility issues.
  Note: Normal bugs occur in each platform but compatibility bug occurs in some platform.



User Acceptance Testing

- Once the application is developed , Tested and stable it will be handed over to the customer.
- Customer before accepting the application performs one round of testing for their satisfaction this testing performed by customer is knowm as "User Acceptance Testing".
- It is performed in the customer env.
  Note: Customer can not blindly trust on company because small issue in the application may lead to business failure.
- We always go for UAT, once the application handed over to the customer.

System Testing

- Checking the end to end flow of an application by nevigating through all the necessary modules of a software is known as system testing.
- In system testing basically we cover all users scenario and check if the application is working as a whole system.
- In large application multiple testers can be assigned and each one can be assigned with some modules. So at the end every testers will be doing system testing on their assigned module. It might involve few other modules except their assigned module, to check the end to end flow. At the end all end to end application should be tested no matter who is tested which module of the application.

  Note: System testing does not mean entire application flow always, if you have module having multiple sub-modules and all are dependent to each other we can do system testing in module level also but please remember module levl system testing is different than application level.

Difference between UAT and System Testing?

UAT

- UAT performs in the customer environment.
- UAT depends on the Customer
- In UAT we check only the main Business scenario.

System Testing

- It is perform in the company environment.
- It is perfomed by the test engineer only
- In this testing we check all end to end flow of the application.

Globalization / Localization

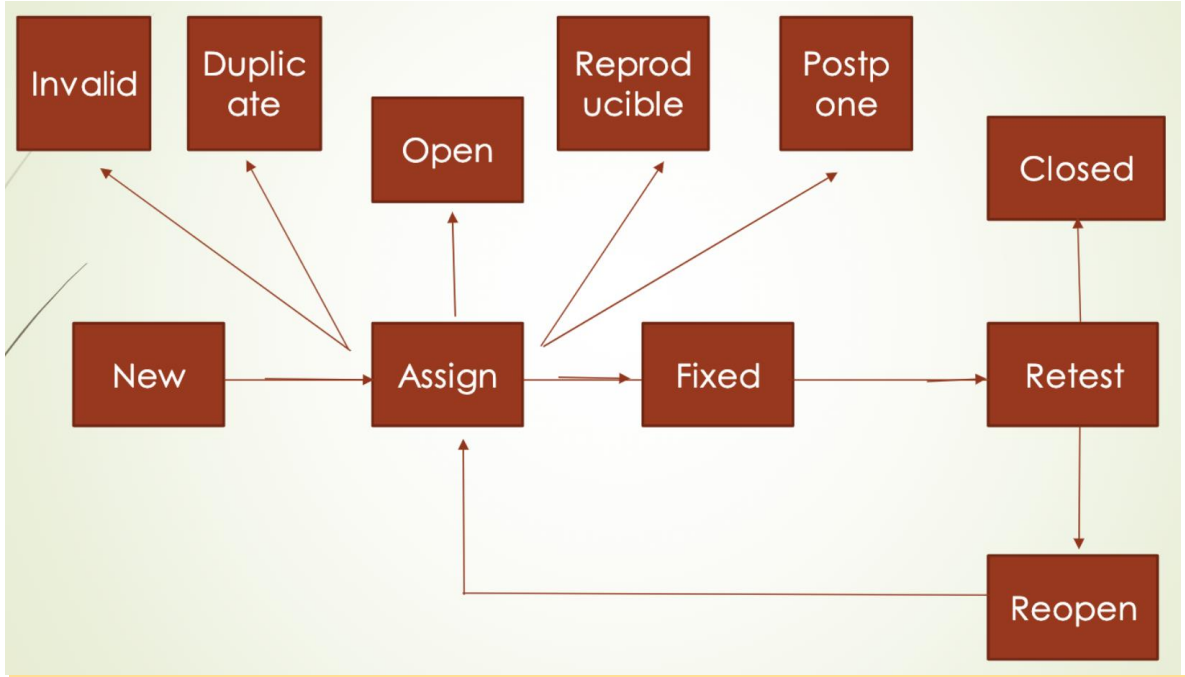
Globalization

- **Globalization Testing** is a software testing method used to ensure that the software application can function in any culture or locale (language, territory or code page) by testing the software functionalities using each type of international input possible. The purpose of Globalization testing is to ensure that software can be used internationally or worldwide. It is also called Internationalization Testing.

Localization

- Localization testing is a key part of the software and website localization process. It takes place when a piece of software or a website is being localized for use in a new language and/or region. A localization tester performs tests on it to ensure that the key fundamentals function properly.

Bug Life Cycle



New : When bug is newly logged in bug sheet.
Assign : when bug is reported to Developer.
Open : once Developer open the filed bug.
Fixed : After fixing the code error by developer.
Retest : At the time of retesting the fixed bug.
Reopen : Assigned same bug to developer again.
Closed : After bug fixed.
Invalid :When developer reject the filed bug.
Duplicate: when same bug is filed by two different test engineer.
Postpone: when developer don't have time to fix the bug.
Not Reproducible: developer not able to find out.

It talks about complete life cycle of a bug right from the stage the bug was found, fixed, retested and closed.

1. As soon as we find a bug the status of bug in report will be given as new which indicates that the bug is just found
2. This bug report is handed over or reported to the consult person(developer) and status is changed to assigned.
3.As soon as developer gets the bug they mark it as open.
4. Go through the entire bug report and to check bug is valid or not
5. If bug is valid then developer starts reproducing the bug followed by navigation steps in the bug report and identify the exact location of the bug.
6. Once the bug is successfully reproduced they start analyzing the code and perform the necessary code changes.
7. Once all these actions are completed the developer changes the status of fix and send it back to the Testers for retesting process.
8. Based on Retesting results the test engineer chnages the status accordingly
Close: if bug is fixed properly.
Reopen: If bug still exist even after the code chages are done
These process continuous until all the bug in application is fixed or closed.

Bug Status:

- **Invalid**: Whenever the developer don't accept the bug or reject the bug, the status is given as invalid.
- **Duplicate**: If the same bug is reported multiple times by the different testers the status is given as duplicate.
- **Not reproducible:** In this case the developer accept the bug but not able to physically locate the bug after going through the navigation steps.
- **Postpone**: Whenever the developer tells that the bug will be fixed in the future releases in such cases the status will be changed to postpone or differed.
- **Can't fix**: Suppose dev is accepting the bug and also able to physically locate the bug but cannot do necessary code changes due to some reasons. In such cases the developer give the status as can't fix.
- **RFE:** Request for enhancement is a kind of enhancement or the suggestion given by the testers towards the development of the application in the form of bug report.

Key points:

- Invalid bug might occur because of testers or developer both.
- Developer always say invalid based on his/her own understanding about the product, but he/she also can miss important things. Always better to prefer requirement doc before marking anything invalid.
- Duplicate bug status is black mark for the testers because it waste the time of developer.
- Not reproducible bug is cause of "Incomplete Navigation Steps", "Platform mismatch", "Test Data Mismatch", "Server Mismatch or inconsistent bug.
- Can't fix status might be because of no technology support or might affect core functionality of the application.
- All Can't fixed bugs are minor bugs but all minor bugs cannot be can't fixed bugs.
- Postpone bug always consider as priority basis, in the early stage developer can not mark any bug as postpone because they will have time to fix it. until he/she does not have any major/critical bug in their plate they can not simply postpone these bugs in early stage.

SDLC

- SDLC Stands for Software Development Life Cycle. It is a step by step procedure to develop an application. It consist of various phases such as
  1. Requirement Collection,
  2. Feasibility Study,
  3. Design,
  4. Coding,
  5. Testing,
  6. Installation and
  7. Maintenance.

Requirement Collection

- In this phase Business Analyst goes to the customer place to collect the requirement or Business need of the customer.
- This requirement collected by the business analyst, as always in the form of document.
- This is also known as Customer Requirement Specification(CRS).

Feasibility Study

- Once the requirement is collected, high level team members will sit together and decide the project durability based on some factors like, whether this can be developed and tested or not?

  Note: As a developer or Testers we have also some boundry line. We cannot blindly start developing anything or test anything. So to start with our work, feasibility study is very much needed.

Design

01

This is like a blueprint of the Application

It is of two types: 1. High level Design, 2. Low Level Design.

02

High Level Design: It is also known as module level design and it is mainly done by architecture.

03

Low Level Design: It is also known as Component level design. It is mainly done by Manager.

Coding

- Once the blue print of an application is ready. It is handed over to the developer. Developer starts writing the code based on the requirement as well as design of the application by choosing the particular programming language.

  Note: Developer cannot choose any programming language his/her self, This might be the part of Requirement document. Most of the time it depends upon the company what they are preferring.

Testing

Once the application is ready, it is handed over to the testers and TE start testing the application until it is stable.

While testing the application TE may encounter some bugs, These bugs needs to be reported back to the developer.

Developer will fix it and give it back to the testers for retesting.

This process of finding the bugs and get it fixed by developer goes on until the application is bug free.

Note: We can never say any application as stable. But we try to build a bug free product to release it in production. Since in IT we push our build in production once it is stable. We use to say this stable word for bug free product.

Installation and Maintenance

In this phase the final stable product is carried from the company environment to customer environment

It is done by the separate team known as installation team.

While using the application, customer may encountered some issues, to overcome such issues developer and testers do support remotely or being available at client place. This comes under the maintenance.

Mithilesh Singh