Software Testing Introduction:

Software Testing: It is a systematic process of testing the functionality and behavior of an application against CRS
(Customer Requirement Specification)

Software Testing: It is an Art of catching the issues in the software against CRS (Customer Requirement Specification)

Software Testing: It is a process of executing the program with the intention of identifying the issue in the software.

SDLC: Software Development Life Cycle.

It is a systematic approach to develop any software application OR products. It start from requirement collection & ends with quality deliverables.

Models in the SDLC:

1. Waterfall model
2. Spiral model
3. Prototype model
4. V & V model
5. Hybrid model
6. Agile model
7. Derived model

Waterfall Model:

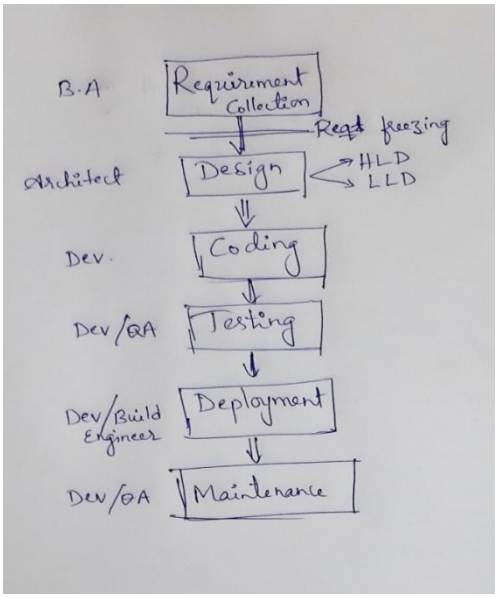
- It is a traditional model.
- It is also k.a., Sequential model OR Linear model

When to go for waterfall model:

-It is suitable for simple s/w
-It is suitable for static s/w (No change in the requirements)
-It is suitable when requirements are very clearly understood by the s/w company.

Different Phases in Waterfall model:

1. Requirement Collection
2. Design
3. Coding
4. Testing
5. Deployment/Installation
6. Maintenance



Requirement collection:

The BA (Business Analyst) from different company will go to customer location to gather the reqts after proper feasibility study.

Feasibility study includes:

(a) Resource availability
(b) Infrastructure OR Lab
(c) Profit

The BA will bid & get the project. Initially the reqts are in the form of CRS/BRS. After collecting the reqts they make a agreement with the customer to freeze the reqts.

Note:

Reqts freezing: means no more new reqts will be considered for development apart from the collected ones.

Reqts freezing is the typical feature of waterfall model.

After collecting the reqts the BA will convert CRS into SRS (Software Requirement Specification)

Design Phase:

Once the SRS is created, it will be shred with Architect.
The Architect/ Sr. Dev will go through the SRS & creates a Design Doc.
It is of 2 types:

1) HLD: High Level Design Doc

It contains a plan for Modules, Sub-modules, interfaces, front end and back end technologies.

2) LLD: Low Level Design Doc

It consist of controls to all the components inside the HLD

Coding Phase:

The Dev will start converting the design into actual s/w by writing the programing languages. Once all the design is converted into s/w we will move to testing phase.

Testing Phase:

As waterfall model is a traditional model, the testing was done by dev. Some times QA can also perform testing. During testing if you find any defects it should be informed to Dev. Once testing is completed & the s/w is defect free, then it will be released to customer.

Deployment Phase:

It is a process the deploying/Installing the s/w in the customer location is k.a., deployment. The deliverables will be kept in FTP machine.

Note: Deliverables: It means any artifacts which are shared with the customer.

The common deliverable includes:
1. Software (appServer, WebServer, DB server)
2. Release Note doc
3. Installation Guide Doc
4. Consolidated Test Execution Report
5. Consolidated Defect Report

Once the deliverables are placed in FTP machine, we send out a release mail to customer. wihch includes following details:
1. FTP machine name OR IP address
2. Credentials (UN/PWD)
3. Location
4. Contents

Note: The customer will have 2 types environments:
1. Stagging OR pre-Production environment
2. Production OR Prod environemnt.

Customer will go through the release mail & install the s/w in staging environment by following Installation Guide doc. After installing the s/w customer will perform one round of testing which is k.a., User Acceptance Testing (UAT). If UAT goes fine the same s/w will be moved to PROD environment.

Maintenance:

As per the agreement the company will provide support to the customer for a given period of time.

Adv of waterfall model:
1. As the reqts are freezed it is very easy to develop the s/w.
2. Less time is required
3. Less resource is required.
4. Less cost is required.

Dis-Adv of waterfall model:

1. As the reqts are freezed customer will not be satisfied.
2. Not all the phases are tested. Testing starts only after coding phase. Hence downward flow of defects will be higher.
3. The cost of fixing the defect will be much higher If the defect is found in the root system
4. Only one/few rounds of testing takes place, hence quality may not much better compare to other models.
5. It is not suitable for dynamic applications
6. It is not suitable for long term applications

Spiral model:

It is also k.a., Iterative & incremental model.

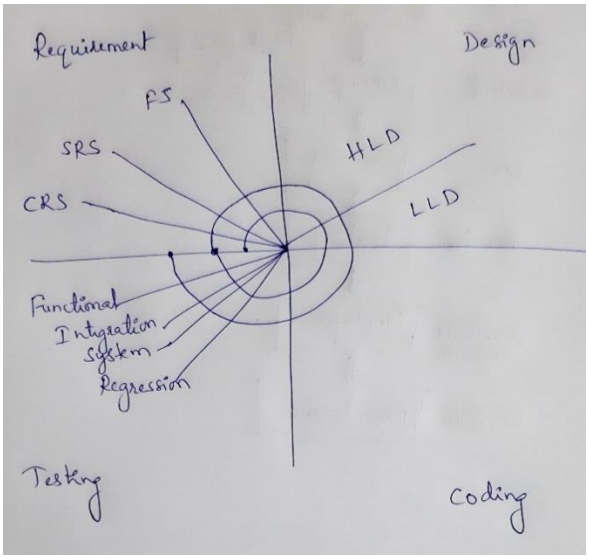
When to go for Spiral model:

1. It is suitable for long term applications
2. It is suitable for inter-dependency applications

Different Phases in Spiral model:

1. Requirement Phase
2. Design Phase
3. Coding phase
4. Testing phase

CRS: Customer Requirement Specification
BRS: Business Requirement Specification
SRS: Software Requirement Specification
FS: Functional Specification



1. Requirement Phase:

BA (Business Analyst) from different companies goes to customer location to collect the requirements after feasibility study. They bid and get the requirements. Initially requirements are in the form of CRS/BRS. BA will convert the CRS into SRS. If required FS also.

Here we collect set of requirements OR module wise requirements. (Unlike waterfall model we don't collect complete requirements here)

---

2. Design Phase:

Once SRS is ready it will shared with Architect. The architect will start converting SRS into design document.
2 types of design doc's:

1. HLD (High Level Design Doc): It consist of a plan for modules, sub-modules, interfaces, front end and backend technologies.

2. LLD (Low Level Design Doc): It consists of controls to all the components inside the HLD.

---

3. Coding phase:

The developer will start converting the design into actual software by writing the programming languages. Once all the design is converted into software, it is given to tester for testing purpose.

---

4. Testing phase:

Once the build (software) is ready it will be given to tester/Dev for testing purpose. Tester will perform following testing's:

(a) Functional Testing
(b) Integration Testing
(c) System Testing
(d) Regression Testing

Once the testing is over & the software is defect free, it will given to customer through deployment process.

Note: The duration of each Iteration in spiral model ranges from 6 months to 2 years.

They collects the new set of requirements hence second Iteration OR subsequent iterations will starts.

Any Production OR PROD issues will be addressed through patch/hot fixes OR Updates

Note:

(a) What is patch/hot fix?

Ans: It is a piece of code which can fix the existing issues in the software.

(b) Do we accept the requirement changes in spiral model?

Ans: Yes, we can accept the requirement changes in every new iterations.

(c) When testing iteration 2, do we need to test the interation 1 features as well?

Ans: Yes, It is done through regression (testing the old features of an application) testing.

(d) What If the problem is too big that the patch/hot fix is not enough to address the issue?

Ans: It is a rare OR never happening scenario. If in case it takes place then we need to stop the new iteration & redo the current iteration only & then release to the market.

OR

They stop the current version and immediately come up with new updated version to the market.

Ex: VISTA OS---> Window8 and later window 10

Advantages:

1. Suitable for long term applications
2. Requirement changes are accepted in every iteration
3. Inter-dependencies can be easily overcome.
4. Compare to waterfall model more testing cycle takes place in spiral model. Hence quality of the software will be good.

Dis-Advantages:

1. Not all the phases are tested. Testing starts only after coding phase. Hence downward flow of defects will be higher.
2. The cost of fixing the defect will be higher If the issue is found in the root system.
3. Not all the inter-dependencies can be overcome.

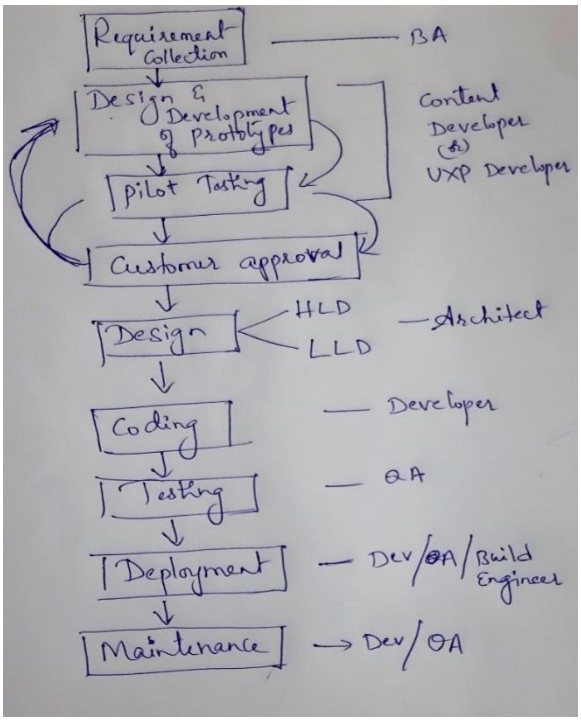
Prototype Model (PM):

Proto= dummy OR Pseudo

In this model initially we develop dummy (Protoype) application & asks for customer approval. If customer needs any changes it will be addressed in the prototype level itself. Once customer approves the prototype we start actual s/w development.

When to go for prototype model:

1. When customer is new to the software
2. When company is new to the domain
3. When requirements are not clear OR not updated.
4. Interaction between customer & company is a challenge.



Different Phases in PM:

1. Requirement collection
2. Design & development of Prototypes
3. Prototype Testing OR Pilot testing
4. Customer approval
5. Design
   (a) HLD
   (b) LLD
6. Coding
7. Testing
8. Deployment
9. Maintenance

---

1. Requirement collection:

BA from different companies goes to customer location to gather the requirements after proper feasibility study.

Once the requirements are collected (CRS/BRS) it will be converted into SRS by BA.

---

2. Design & development of Prototypes:

The SRS will be given to Content developer OR UXP developer. The content dev will start design & development of prototypes.

There are 2 types of prototypes:
(1) Static prototypes
(2) Dynamic prototypes

---

3. Prototype Testing OR Pilot testing:

Once content dev design the prototype, themself they will test the prototypes. The process of testing the prototype is k.a., prototypes OR pilot tesitng.

---

4. Customer approval:

All the tested prototypes will be given to customer for approval. During approval of prototypes the customer can ask for the changes/modifications. The content dev will modify the prototypes based on customer feedback , test it once & send to customer approval again. Once customer approves the prototypes we will go to actual design of the s/w.

---

5. Design Phase:

Once the SRS is created, it will be shred with Architect.
The Architect/ Sr. Dev will go through the SRS & creates a Design Doc.
It is of 2 types:

1) HLD: High Level Design Doc

It contains a plan for Modules, Sub-modules, interfaces, front end and back end technologies.

2) LLD: Low Level Design Doc

It consist of controls to all the components inside the HLD

---

6. Coding:

The dev will start converting design into actual software by writing programming languages (High level languages). Once the s/w is ready it is given to tester for testing purpose.

---

7. Testing:

Tester will understand the requirements, write the test cases & test the application as per the test case doc. If tester finds any defects during testing it will be reported to dev through a defect tracking tools (Ex: Bugzilla, JIRA, TFS etc)

---

8. Deployment:

Its a process of deploying/installing the s/w in customer location. It can be done by deployment/Build engineers OR dev OR QA.

All the deliverables are kept in FTP machine & sendout a Release mail to the customer.

Customer will have 2 environments (computer OR set of computers OR Servers in which the software is installed is k.a., environments) namily:

(a) stagging OR pre-production environment

(b) production environment

Whenever we make a release to customer, they deploy the s/w in staging environment first & perform one round of end-to-end testing which is k.a., User Acceptance Testing (UAT). If UAT goes fine the same s/w will be moved to production(PROD) environment. If UAT fails, it will be escallated back to the company.

---

9. Maintenance:

As per the agreement the company will give support to the customer for a given period of time.

Adv of PM:

**1. We can set the customer expectation in the very early phase of application development.
2. Protoypes can be reused (Dynamic prototypes).
3. Suitable for dynamic applications.
4. Suitable for long term applications.
5. Both Dev & QA will have better understanding of the application with the help of prototypes.
6. Cost of fixing the defect in the later stages is low bcoz most of the issues are addressed at the prototype level itself.

Dis-Adv of PM:

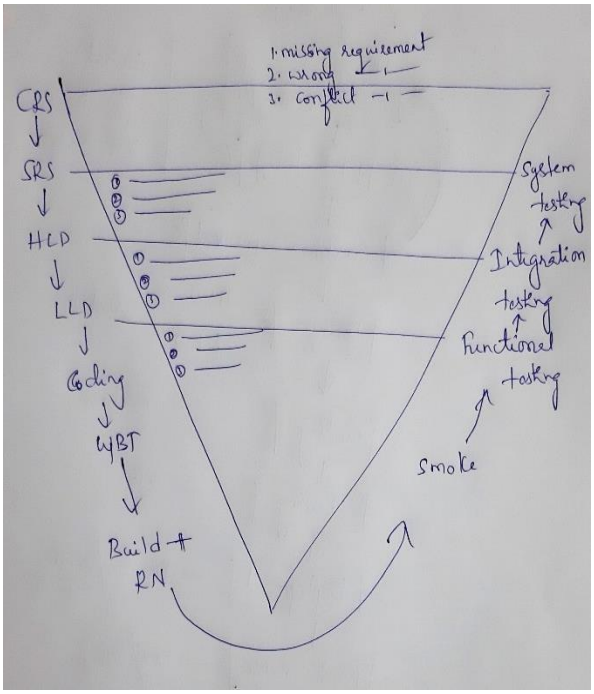
1. Initial high investment is required.
2. Wastage of prototypes
3. wastage of time and cost in the process of prototype development
4. Any changes in the later stages should route through prototypes

V & V Model:(Verification & Validation Model)

Verification: Its a development activities. It is defined as "Are we building the right product".
Verification is a process of evaluating software at the development phase. It helps you to decide whether the product of a given application satisfies the specified requirements
Validation: Its a QA activities. It is defined as "Have we built the product right"
Validation is the process of evaluating software at the after the development process and to check whether it meets the customer requirements.
Q: When to go for V&V model:
Ans: When customer wants high quality product within a short period of time.

Different phases in V&V model:
1) Requirement phase
2) SRS
3) Design

(a) HLD
(b) LLD
4) Coding
5) Testing
6) Deployment



1) Requirement phase:

The BA from different companies goes to customer location to gather the requirements after feasibility study. Once the requirements (CRS) are collected it will be equally shared between Both Dev & QA team.

QA will start testing/reading the CRS doc to find:

(a) Missing requirements:

Some info is missing from the CRS doc. While reading CRS, QA didn't find complete information about the given features.

Ex: While reading "Compose" requirements, QA didn't find info for attachments.

(b) Wrong requirements

QA found that the requirement was not correct as per the business.

Ex: While reading "User Type" requirement, QA found that All the users viz., Admin, Paid-user & non-paid user are having same access rights. But as per business only Admin user should have all the privileges. But doc tells something else. Hence it might be a wrong requirement.

(c) Conflict requirements:

The same details are put it in 2 different ways.

Ex: The date format was different in different pages of the given requirement viz., 'dd-mm-yyyy' in few places & 'mm-dd-yyyy' in other place. Hence its a conflict bcoz we are not sure which one is correct.

Note: If QA finds any issues in the CRS doc, then the defect must be logged against BA as a 'Documentation Defect'.

---

2) SRS

Once BA converts CRS into SRS, it will be shared among Dev & QA. QA will goes through SRS and starts preparation for System Testing which is as follows:

1. Review SRS v/s CRS
2. Write System Test plan
3. Write System Test cases

Note: If QA finds any issues in the SRS doc, then the defect must be logged against BA as a 'Documentation Defect'.

3a) HLD:

When QA gets HLD doc they will start preparation for Integration testing which is as follows:

1. Review HLD v/s SRS
2. Write Integration Test plan
3. Write Integration Test cases

Note: QA will attend the developer's design review meeting to understand more about HLD.

3b) LLD:

When QA gets LLD doc they will start preparation for Functional testing which is as follows:

1. Review LLD v/s HLD
2. Write Functional Test plan
3. Write Functional Test cases

Note: QA will attend the developer's design review meeting to understand more about LLD.

Coding & Testing:

Team will prioritize the requirements & as per the prioritization dev will start developing one by one features. Once the features are developed, the dev will create a build (Its a s/w having few set of features implemented/developed) & given to testing team along with Release Note(RN) doc for testing purpose.

Note:

1. Build naming convention will be usually <projectName_Version>
2. Release Note is a doc which consist of current build scope. It is created by Dev.

Current Build scope means: The purposes of giving the current build OR the list of features developed in the current build.

What QA supposed to do upon getting every build?

1. QA should go through the RN & understand the build scope
2. Select the test cases as per the build scope
3. Execute the test cases against the build. During execution If QA finds any defects it will be logged against dev in a defect tracking tools (Ex: Bugzilla, JIRA, TFS, ALM etc)
4. Prepare a consolidated test execution report and share with the team.

The same procedure is repeated for every subsequent builds until we reach the release scope.

Adv of V&V model:

1. Deliverables happens parallel hence both dev & QA are in sync
2. Each & every phases are tested. Hence downward flow of defect is negligible.
3. The cost of fixing the defect in later stages is low bcoz most of the issues are addressed in the intital stage only.
4. Highly suitable for dynamic applications
5. Suitable for long term applications.
6. Within a short period of time we can develop robust application.

Dis-adv of V&V model:

1. High initial investment
2. Its slow compare to agile model.

Test Case Example.xlsx

Hybrid Model:

Combination of more than one model put together makes hybrid model.

Q: Why Hybrid Model?

Ans: If user wants to have advantages of more than one models in single place then go for hybrid model.

Ex: Hybrid between V&V + Prototype:

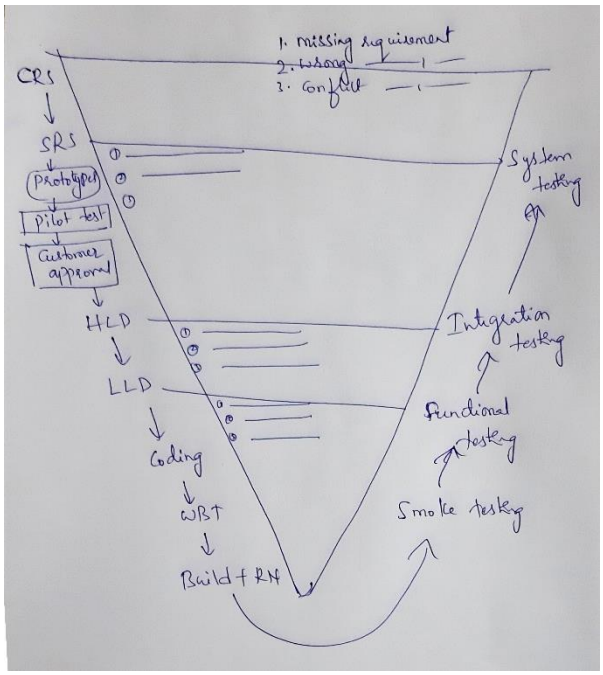
Q: When to go for V&V + Prototype model?
Ans:
1. When customer wants high quality s/w within a short period of time.
2. When customer is new to the s/w.
3. When company is new to the domain
4. Interaction between customer OR client is a challenge.

Different Phases in Hybrid model:
1) Requirement phase
2) SRS + Design & development of prototypes
3) Design
   (a) HLD
   (b) LLD
4) Coding
5) Testing
6) Deployment
7) Maintenance

---

1) Requirement phase:

The BA from different companies goes to customer location to gather the requirements. Once the requirements (CRS) are collected it wil be equally shared with Both Dev & QA team.



QA will start testing/reading the CRS doc to find:
(a) Missing requirements:
Some info is missing from the CRS doc. While reading CRS, QA didn't find complete information about the given features.
Ex: While reading "Compose" requirements, QA didn't find info for attachments.

(b) Wrong requirements
QA found that the requirement was not correct as per the business.
Ex: While reading "User Type" requirement, QA found that All the users viz., Admin, Paid-user & non-paid user are having same access rights. But as per business only Admin user should all the privileges. But doc tells something else. Hence it a wrong requirement.

(c) Conflict requirements:
The same details are put it in 2 different ways.
Ex: The date format was different in different pages of the given requirement viz., 'dd-mm-yyyy' in few places & 'mm-dd-yyyy' in other place. Hence its a conflict bcoz we are not sure which one is correct.

Note: If QA finds any issues in the CRS doc, then the defect must be logged against BA as a 'Documentation Defect'.

---

2) SRS

Once BA converts CRS into SRS, it will be shared among Dev & QA. QA will goes through SRS and starts preparation for System Testing which is as follows:

1. Review SRS v/s CRS
2. Write System Test plan
3. Write System Test cases

Meanwhile the content dev. will start design & development of prototypes. Once the prototypes are created it will be tested which is also k.a., Pilot testing. All the tested prototypes will be sent to customer approval. Whichever prototypes customer approves will be moved to actual design phase. If customer wants any changes it will be addressed by the content dev.
Note: If QA finds any issues in the SRS doc, then the defect must be logged against BA as a 'Documentation Defect'.

3a) HLD:
When QA gets HLD doc they will start preparation for Integration testing which is as follows:

1. Review HLD v/s SRS
2. Write Integration Test plan
3. Write Integration Test cases

Note: QA will attend the developer's design review meeting to understand more about HLD.

3b) LLD:
When QA gets LLD doc they will start preparation for Functional testing which is as follows:
1. Review LLD v/s HLD
2. Write Functional Test plan
3. Write Functional Test cases
Note: QA will attend the developer's design review meeting to understand more about LLD.

Coding & Testing:
Team will prioritize the requirements & as per the prioritization they will start developing one by one features. Once the features are developed, the dev will create a build (Its a s/w having few set of features implemented/developed) & give it to testing team along with Release Note(RN) doc for testing purpose.
Note:
1. Build naming convention will be usually <projectName_Version>
2. Release Note is a doc which consist of current build scope. It is created by Dev.
What QA suppose to do upon getting every build:
1. QA should go through the RN & understand the build scope
2. Select the test case as per the build scope
3. Execute the test cases against the build. During execution If QA finds any defects it will be logged against dev in a defect tracking tools (Ex: Bugzilla, JIRA, TFS, ALM etc)
4. Prepare a consolidated test execution report and share with the team.

The same procedure is repeated for every subsequent builds until we reach the release scope.

Adv of V&V+ Prototype model:
1. Deliverables happens parallel hence both dev & QA are in sync
2. We can set the customer expectations in the very early stage of application development.
3. Each & every phases are tested. Hence downward flow of defect is negligible.
4. The cost of fixing the defect in later stages is low bcoz most of the issues are addressed in the initial stage only.
5. Prototypes can be reused.
6. Both Dev & QA will have better understanding of the s/w with the help of prototypes.
7. Highly suitable for dynamic applications
8. Suitable for long term applications.
9. Within a short period of time we can develop robust application.

Dis-adv of V&V + Prototype model:
1. High initial investment
2. Its slow compare to agile model.
3. Wastage of prototypes
4. Wastage of cost & time in the preparation of prototypes
5. Any changes in the later stages should route through prototype development.

Agile model:

"able to move quickly and easily"

OR

"A method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans."

It is a customized model. It is a combination of Waterfall + Spiral model.

Different Agile Methodologies:

1. Scrum
2. Crystal Methodologies
3. Dynamic Software Development Method (DSDM)
4. Feature Driven Development (FDD)
5. Lean Software Development
6. Extreme Programming
7. Kanban

In Agile model we use SCRUM methodology.

==========

Agile process

1. Agile team formation
2. User stories collection (Requirements)
3. Sprint planning
4. Product backlog review
5. Sprint kick-off meeting
6. Sprint
   (a) Daily Scrum meeting OR Daily Standup meeting
   (b) Implementations OR Coding
   (c) Unit Testing OR White Box Testing
   (d) Review
   (e) Testing
   (f) Release
7. Sprint Review OR Sprint Retrospect meeting
8. Next Sprint

---

1. Agile team formation:

A scrum Master (SM) will play a vital role in the Agile Team Formation.

A typical Agile/Scrum Team consist of min 5-7 Dev & atleast 1 QA put together. The no. of Agile/Scrum teams depends on:

(a) Project Responsibilities
(b) Complexity of the project
(c) Team Size/Head count

All the team members should have mindset of owning complete product responsibilities (Holistic Approach).

The team members should be a motivated individual & having required skill set.

---

2. User stories (US) collection:

In Agile, requirements are also k.a., User Stories. The Product owner will collect the US from the customer. He is a SPOC (Single Point Of Contact) between company and Customer.

Product owner goes to customer location, explains about the agile process and collects the US from the customer. Once the US are collected it will be handed over to the SM.

---

3. Sprint planning:

Once US are given to SM he will perform the Sprint Planning. Which includes:

(a) No. of sprints required to develop the s/w

(b) Spint Window will be defined

(c) Segregation of US in each sprint.

Sprint: Its is a total duration period of time in which all the selected US are developed is k.a., Sprint.

Sprint Window: A fixed duration for the sprints. Usually it will be 3 weeks.

Scrum Master Responsibilities:

1. Agile team formation
2. Defining sprint window
3. Defining # of sprints
4. Defining daily standup meetings, status meetings, kick-off meetings etc.
5. Prioritization & Allocation of the tasks to the individuals.
6. Solving OR resolving the challenges/impediments/Crux among the scrum team members.
7. Sharing the work status with the stake holders (Any one who is directly OR indirectly involved with your project. Ex: Sales, Marketing, Customer, Dev, QA, Product Owner etc).

Note: He acts as a facilitator.

---

4. Product backlog review:

It is process of identifying the feature/User stories/fixes which are part of the current sprint.

It may include new user stories, Defects from the previous sprint, the user storeis pending from the last spint etc

What are the activities takes place in Product backlog review:

1. Task prioritization and allocation to the individuals.
2. All the team members will go through the US & understands them. During the process of understanding of US we can talk/discuss with BA, Product Owner, Dev, PEER etc.
3. Both Dev & QA must chalk out/Write down risk assessments.
4. All the team members has to provide the estimation to the tasks allocated to them by including both Dev & QA efforts put together.
5. SM will collects the estimations from all the team members and create a Story Point **(A unit of estimation measuring complexity and size is k.a., Story Point)**
6. Once the given estimation is sustainable, then the same thing will be communicated to the stake holders & then the current sprint US will be freezed. (No more new US will be added to the current sprint after freezing the reqts.)

---

5. Sprint kick-off meeting:

In sprint kick-off meeting we will show the preparation checklist

QA checklist:

1. Documentation is ready (Test case & Testplan)
2. Risk outlined and understood
3. Preparation for the QA activities

Dev checklist:

1. Design understood
2. Ready with plan and documentation etc

SM will share the meeting details with all the stake holders

---

6. Sprint:

All the scrum team members will attend the Daily scrum meeting OR Daily standup meeting. This meeting should be completed within 15-20 min.

The following things will be discussed by the individuals during the Daily Scrum meeting:
1. What we have done yesterday
2. What we are going to do today
3. Any challenges OR impediments OR Crux

After the meeting QA goes to their desk, write/update the test cases as per the discussion.

Once the test cases are written, it should undergo review process. In Agile BA, Dev will involve in the review meetings along with the QA.

QA has to talk to BA & Dev, get their availability & setup a review meeting with them. After the review, the QA has to fix all the review comments, Baseline the test cases and store them in test case repository. **(It is a centralized place in which the test cases are stored is k.a., Test case repository).**

Once Dev completes the coding for the discussed features they will create a build & give to QA for testing purpose along with Release Note (RN) doc.

Once QA gets the build + RN:
1. QA should go through the RN & understand the build scope **(What are the features are implemented/developed in the current build is k.a., Build scope).**
2. Select the test cases as per the current build scope
3. Execute/Run all the selected test cases against the build. During execution If QA find any defects it should be logged against the dev in a defect tracking tool.
4. Finally prepare the consolidated Test execution report.

The same process will be repeated for the next subsequent days until all the sprint features are developed and tested.(usually 3 weeks)

SCRUM: It is a development activity.
"It is an iterative & incremental way of the developing all the selected use stories which are part of the current sprint is k.a., SCRUM"

---

7. Sprint Review OR Sprint Retrospect meeting OR Post Marton meeting:

Once the sprint is completely developed and tested, we will perform deployment process OR release the s/w to the customer.

After the release of the software, all the agile team members will perform a Sprint Review OR Sprint retrospect meeting.

During Sprint review meeting all the individuals should talk about following things:
1. what are the best practices we have followed
2. Any process which requires improvement.
3. Any challenges faced and how did you overcome.
4. Any suggestions

The SM will document the retrospect discussions. So that the documentation will act as a base for the next upcoming sprints.

After completion of Sprint 1 we may perform product backlog review & Sprint kick-off meetings for Sprint2. So that the moment Sprint1 completes we are ready for the sprint2 activities.

In Sprint2/subsequent sprints we can have new features, defects from previous sprint, backlog from previous sprint etc.

Recommended Practices:
Development:
TDD - Test Driven Development
Weekly Code Reviews for critical fixes with the Agile team (including QA).

QA:
Review of Risk analysis with the Agile team.
Review of test coverage/plan with the Agile team.
Discuss all critical defects found, at the earliest, with the Agile team, and with stake holders

Note:
a) It is highly desirable to have automated test cases both at unit level (Dev) and at system level (QA). This helps in becoming agile.

Advantages OR Principles of Agile:

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
Continuous attention to technical excellence and good design
9. Simplicity—the art of maximizing the amount of work not done—is essential
10. Self-organizing teams
11. Regular adaptation to changing circumstances

Dis-Advantages:
1. it is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. cost of agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.
==========
Q & A
==========
1. What is Daily scrum meeting? What you do in Daily stand up meeting?

2. How the user stories will be assigned to you?
Ans: The user stories will be assigned to each individuals in a JIRA tools. Once done all team members will get notification.

3. What is Sprint?

4. What is SCRUM?
Ans: Scrum is a development phase. It defines "It is an iterative and incremental approach of developing a s/w product"

5. How many sprint you have completed in current project?
Ans: around 4-5 in the current project.

6. What is your role in Agile? OR Explain Agile process in your organization?
Ans: As we are practicing agile our day starts with Daily standup meeting (DSM). Here we discuss following things:
A. What we have done yesterday
B. What we are going to do today
C. Any blocker OR challenges OR impediments
2. After the DSM, QA goes to his desk update/write the test cases. Once done it should under go review process. In agile review should be done F2F. Hence QA will setup a meeting with Dev & BA.
3. After review, fix all the review comments, baseline the test cases and store them in Test case repository
4. When build comes select and Execute the test cases against the build based on the Build scope. During execution we perform Manual execution (wherever required) and automation execution as well.
5. If you find any defects, log the defect against dev in a defect tracking tool
6. Finally prepare a consolidated test execution report.

Similarly multiple builds can happen in every sprint until all the sprint features are implemented.

Q: How do you project the work status?
Ans: Through the daily status report & as well as thorugh "Burndown chart".
Note: Burndown charts are graphs that give an overview of progress over time while completing a project. As tasks are completed, the graph “burns down” to zero.
A burn down chart is a graphical representation of work left to do versus time. The outstanding work is often on the vertical axis, with time along the horizontal. Burn down charts are a run chart of outstanding work. It is useful for predicting when all of the work will be completed

Q: Who will assign the task to you in Agile?
Ans: Scrum Master
Q: Why do you do standup OR Scrum meeting?
Ans:
1. It establishes a proper sync between dev & QA with respect to project requirements OR task.
2. We can understand with move rapidly.
3. Mis-understandings with respect to requirements will be reduced drastically.

Functional Testing:

Testing the functionality and behavior of each & every components regoursly is k.a., Functional Testing.

The extended level of functional testing is also k.a, GUI Testing.

When Dev completes the coding of the selected requirements, they will create a build & give it to testing team along with Release Note (RN) for testing purpose. QA will go through RN, select the test cases as per the current build scope.

Pick functional test cases & start executing them as part of functional Testing.

Note: Always perform +ve & -ve testing’s.

The following Test cases will be selected.
1. Functional Test cases
2. Integration Test cases
3. System Test cases
4. Smoke test cases

For example we need to perform Functional Testing on the FB registration page.

FB Reg Page Reqts will talk about the following components:

1. First Name Edit field: It should accept characters, Mandatory & 8-15 digits
2. SurName edit field: It is an Optional field & accepts characters, special characters
3. Email OR Phone number Edit field: It should accept Number, special characters (@,., _ etc) & characters
4. Password edit field: It should accept characters, special character, numbers, 8-15 digits
5. Retype Password edit field
6. Gender Radio buttons
7. DOB drop downs
8. Submit Button

Sample functional Test cases for FB Reg Page:

1. All the below components should exist in FB Reg page (+)

a). First Name Edit field:
b). SurName edit field:
c). Email OR Phone number Edit field:
d). Password edit field:
e). Retype Password edit field
f). Gender Radio buttons (Male, Female & Other)
g). DOB drop downs
h). Submit Button

2. Enter valid deails in all the fields and click on "Submit" button & verify FB Reg happens successful.(+)

3. Functional test cases for below components:

(a) First Name Edit field
- a.1: Astrix(*) symbol should be present (+)
- a.2: Enter characters only (+)
- a.3: Enter characters bet 8-15 digits (+)
- a.4: Enter <8 digits OR >15 digits (-)
- a.5: Enter only numbers (-)
- a.6: Enter alpha numeric OR special characters (-)
- a.7: Enter blank and click on Subit button(-)

(b) SurName edit field:
- b.1: Keep it blank (+)
- b.2: Enter characters of 8-15 digits (+)
- b.3: Enter Special characters OR alpha numeric (+)
- b.4: Enter numers (+)

(c) Email OR Phone number Edit field:
- c.1: Enter characters & special characters (+)
- c.2: Enter 10 digits number (+)
- c.3: Enter special characters @, _ and . (+)
- c.4: Enter other special characters (, #$ etc) (+)
- c.5: Enter mail id without @ (-)
- c.6: Enter phonenumber less than 10 digits (-)

Note: These below points are applicable to all the testing's

1. Always perform +ve & -ve testings for the requirements.
2. Always perform +ve testing first. If s/w is working for +ve testing then only perform -ve testing.
3. While writing test case write both +ve & -ve test cases
4. Always perform optimized testing. Never perform under testing OR over testing.

Q: What is +ve testing?
Ans: Testing the application with valid input data is k.a., +ve testing.
Q: What is -ve testing?
Ans: Testing the application with invalid input data is k.a., -ve testing.

During Functional testing (while executing functional test cases) if QA finds any defects it should be logged immediately against the dev in a defect tracking tools. (EX: Bugzilla, JIRA, ALM, TFS, Mantis etc). Once all the functional test cases are executed, prepare consolidated test execution report.

Consolidate report should consist of:
# of test cases selected for execution: 100
# of test cases executed/Completed: 50
# of test cases passed: 45
# of test cases failed: 5
# of test cases blocked: 10
# of test cases pending: 40
# of defects logged: 3

Integration Testing (IT):

Testing the interface OR bridge OR gateway between 2 or more modules or features is k.a., IT.
IT of 2 types:
1. Incremental Integration Testing
   (a) Top down
   (b) Bottom up

2. Non-Incremental Integration Testing
   (a) Bigbang
   (b) Sandwitch
---
1. Incremental Integration Testing:
(a) Top down:
This type of integration testing is performed when you have a modules in which the data is flowing from higher modules towards its lower modules.

Arrange the modules in such a way that the dataflow is happening from higher modules towards its lower module. Verify Data flows from higher module successful & lower module receives the data successful.

(b) Bottom Up:
This type of integration testing is performed when you have a modules in which the data is flowing from lower modules towards its higher modules.

Arrange the modules in such a way that the dataflow is happening from lower modules towards its higher module. Verify Data flows from lower module successful & higher module receives the data successful.
---
2. Non-Incremental IT
(a) Big bang: Here a single module can have multiple interfaces exposed to multiple modules.
Drawbacks of Big band from testing point of view:
1. We may miss to test some of interfaces among many.
2. It is very difficult to identify the root cause of the defects.
3. It takes more time to test as we need to test all the interfaces.
(b) sandwich:
It is a combination of incremental integration testing & non-incremental integration testing.

Stub and Driver:
Stub: A dummy low level module in top down incremental integration testing is k.a., Stub. It simulates the activities of low level model.
Driver: A dummy high level module in bottom up incremental integration testing is k.a., driver. It simulates the activities of high level model.

Q: Who provides the stub/Driver?
Ans: The dev will provide the stub/Driver to QA for testing purpose until the actual module is ready.

Q: There are 4 modules are present. What kind of integration testing you perform?
Ans: A B C D
First understand the relationship between each modules & then based on the relationship we decide what kind of integration testing need to be done.
Q: What is system Integration Testing?
Ans: Integration between 2 independent s/w's is k.a., system Integration Testing.
Ex: Amazon + Payment modules

System Testing:

It is an end-to-end (E2E) testing in which the QA will test the actual customer/business scenarios using customer data is k.a, System Testing.
As we are performing actual customer business scenarios we do need actual customer test data. For that we ask data OR DB dump from the customer.

Requirement for System Testing:
1st time apply for O.D (Over Draft)
Rs. 50,000/-
Interest: 3,000/-
Activation: 1,000/-

54,000/-

2nd Time apply O.D
Rs. 50,000/-
Interest: 3,000/-

53,000/-

Smoke Testing (ST):

Testing the basic & critical features of an application at a very high level before performing formal testing's is k.a., ST.

It is also k.a., Skim testing, Dry run testing & quick testing.

Formal testing's: Functional, Integration & System Testing's

Note:
(1) ST is a +ve tetsing. Never perform -ve testing under smoke testing.
(2) If ST works fine, and then only continue with further testing otherwise we should stop our testing & Reject the build.

Q: Why ST is required OR Advantages of ST?
Ans:
- It is done to make sure build is eligible for further testing.
- It is done to catch/find all blocker & critical defects at the earliest.
- Finding defects at the earliest helps developer to spend sufficient time to fix the defects.
- Project Schedule will not be affected.

Q: When to perform ST?
Ans: It should be done at the initial stage before the start of formal testings.

Q: Does ST requires Test cases?
Ans: Yes. We do write Test cases for the ST.

Q: What is the time duration for ST?
Ans: It should be completed within 30-45 mins only.

Q: Example of performing ST?
Consider ST on gmail

Features	Time taken
- Login	5 min
- Registration	5 min
- Compose	5 min
- Inbox	5 min
- Draft	5 min
- Thrash	5 min
- Spam	5 min
- Settings	5 min
- Sent Item	5 min
- Logout	5 min

Q: Does dev also perform ST?
Ans: Yes. It is performed at Unit level by dev.

Q: What If ST fails?
Ans:
(a) Stop testing & log the blocker/Critical defect against dev.
(b) Ask dev to provide patch/hot fix.
(c) When dev provides patch/hot fix, apply them in QA environment by following the steps mentioned by the dev in the mail.
(d) After applying patch, verify that the issue is addressed. If yes, continue further testing. If No, again ask dev to provide the patch/hot fix.

Types of ST
(a) Formal ST: Test cases are present. We do refer the test cases are perform ST.
(b) Informal ST: Very rare scenario in which we might not have test case for ST. But still we perform ST.
Q: can we automate ST?
Ans: Yes

Q: Does ST covers whole application?
Ans: Yes

Q: What is the order of testing?
Ans:
Smoke--> Functional--> Integration--> System


SystemTestCase.xlsx

Ad-Hoc Testing (AT):

Testing the application randomly without referring to the requirements is k.a., AT.

It is a -ve testing. Here we don't refer to requirements. We just randomly perform the testing.
It is also k.a., Random Testing, Monkey Testing OR Gorilla Testing

Q: When to do AT?
Ans: AT is not a mandatory Testing. But it is recommended. Hence it should be done after completion of all the testing's provided If you have a time.

Q: Does Test cases are required for AT?
Ans: No.

Q: Why ad-hoc Testing is required?
Ans: Customer can encounter the issue when they use the s/w randomly. We don't want our customer to face the issues when they use the s/w randomly. Hence we act as end user & perform AT.
Note: Most of the ad-Hoc defects are invalid. It will we considered valid only when the issue is having impact on the business.

Q: What QA has to do If they find blocker OR critical defects during AT?
Ans:
(a) QA should log the blocker defects immediately against the dev.
(b) QA must document the defect (Write the Test case). So that the same idea will be executed over all the builds.

Q: What is the order of testing?
Ans: Smoke--> Functional--> Integration--> System--> Ad-Hoc

Regression Testing (RT):

It is a process of re-executing the same test cases in every release just to make sure the new modifications are not affecting the existing functionalities of an application is k.a., RT.

Note: Regression testing is a bridge between manual & automation testing.

Characteristics of Regression features:
1. The feature which is completely implemented & stable.
2. The feature which is completely defect free.
3. The feature which does not have any CR (Change Request) nearby.
4. The feature which has reached the PROD.

Q: Do we Write test cases for RT?
Ans: No. Instead we use existing test cases for the rergession.

Types of RT:
1. Unit Regression
2. Regional Regression OR Risk Based Testing
3. Full Regression

1. Unit Regression:

Some cases the new modifications are not going to affect the existing functionalities of an application. Which is termed as Unit Regression.

Ex: Spelling mistake. Alignments etc.

QA Actions:
(a) Perform complete smoke testing
(b) Test the new changes

---
2. Regional Regression OR Risk Based Testing:

In some cases we can easily predict the affected areas which are due to the new modifications. Hence we take test cases of the impacted regression feature & perform regression Testing. which is termed as Regional Regression.

As most of the cases the impact areas are predicted/guessed. Hence going by our prediction and performing RT might be a risky approach. But due to the time crunch situations we might go for it.

QA Actions:
(a) Perform complete smoke testing
(b) Test the new changes
(c) Select and execute the impact area regression test cases.

---
3. Full regression:

In some cases it is very difficult to predict the impact areas caused due to new modifications. Under such situations we go for Full Regression.

QA Actions:
(a) Perform complete smoke testing
(b) Test the new changes
(c) Select and execute all the regression test cases.

Note: We always perform full regression

Q: How many Regression test cases you have?
Ans: Around 1700+ regression test cases we have right now & counting.
Q: How many Regression Test cas you have written?
Ans: We don't write Regression Test cases. Instead we use existing test cases for the rergession.
Q: What is the start point of automation?
Ans: Regression test cases influence the start of the automation.

Q: If you are a test automation engineer? How do you start your automation?
Ans:
-> Firsts we will assigned with Regression Test cases.
-> Read OR execute, and understand the Regression Test cases.
-> Findout the regression test cases which can be automated.
-> Create a POC (Proof of Concept) & give the demo to the team
-> If approved start adding the new test cases.

Q: Order of testing?
Ans:
Smoke-> Functional-> Integration-> System-> Regression-> Ad-Hoc

Re-Testing:

It is a process of validating the defect fix to make sure the fix (Defect fix) is working as expected is k.a., ReTesting.

It is also k.a., Confirmation Testing.

QA has logged 3 defects in the last build.
1. Logo is missing
2. Cancel button is not working
3. Attachment is not working for excel file.

Dev has fixed all the 3 defects in the next build & given to QA for testing purpose.

Now the new build has 3 things:
1. Old feature OR regression features
2. Newly implementation OR code change
3. Defect fix.

Validating the defect fix to check that the fix is working or not is k.a., Re-Testing.
Q: What is the order of testing?
Ans:
Final order of testing:
Smoke-> Functional-> Integration-> System-> Retesting-> Regression-> Ad-hoc

Exploratory Testing (ET):

Explore OR understand the application first, write the test cases and then test the application is k.a., ET.

OR

Exploratory testing is an approach to software testing, wherein testers learn simultaneously about the test design and test execution.

Q: When to go for ET?
1. When reqts are not clear OR not updated
2. Reqts are present but no time to go through it.
3. The old application came for updating.
4. When you join the organization in the middle of the project.

Q: Dis-Advantages of ET?
1. There is no baseline to test the application.
2. We may understand feature as a defect.
3. We may understand defect as a feature.
4. If dev fails to implement any feature, QA will never come to know about it.
5. More rework both at Dev & QA side.
6. Wastage of time in exploring the root cause of the defects.
7. We may miss to find some defects.

Q: How to perform ET?
Ans: To perform ET, we need to follow "Exploratory Testing life cycle".

Exploratory Testing Life Cycle:

1. Identify the need for ET
   |
   V
2. Explore OR understand the application to the fullest
   |
   V
3. Write Test cases based on your understanding
   |
   V
4. Test case Review process ******
   (a) Internal Review (Done by BA, Dev & Peer)
   (b) External Review (Done by Customer)
   |
   V
5. Fix all the review comments, baseline the testcases and store the test cases in test case repository
   |
   V
6. Select & execute the testcases based on the relase note scope whenever build comes. During test case execution If you find any defects it should be logged against dev in a defect tracking tool.
   |
   V

7. Prepare consolidate test execution report & attach all the test cases which QA has used to perform the testing.
Note: Attaching test cases is very important here. Also mention that the application is tested using only these attached test cases.
Q: How to make ET very effective?
Ans:
1. Explore the application to the fullest.
2. Effective interaction with BA.
3. Effective interaction with Dev.
4. Effective interaction with Customer.
5. Use your previous domain experience.
6. Proper documentation both at Dev & QA level.
7. Effective & timely review process at every steps.

Compatibility Testing (CT):

Testing the same s/w in different hardware OR software Flavours (combinations) is k.a, CT.

CT is devided into 2 groups:
(i) Hardware Compatibility
Ex: Input devices, output devices, Processor, RAM, VGA card etc.
(ii) Software Compatibility
(a) Browsers
(b) Opearting Systems
(c) Apps/Devices

Q: How to decide the flavors?
Ans: It is decided mainly with respect to the ROI (Return on Investment)

Note:
1. Initially the s/w is developed on 1 std flavour. Once the application is stable then we will migrate the s/w to remaining flavours.
2. In CT the version is very important.
3. Nowadays we use VMwares OR Dockers to perform the CT on the required flavours.
4. It is mandatory that we need to perform CT on a fresh OS or Browsers.
5. When we log the defects in CT it is highly mandatory to mention the exact version of the OS & Browsers are used.
6. It is recommended that the same person has to perform the testing on all the flavours. So that they can find the behavioural difference of the s/w on different flavours. But in practical it is highly impossible.

Types of defects in CT:
(a) Functionality defects (FD): The issue which is persist in all the selected falvours is k.a., FD.
(b) Compatibility defects (CD): The issue which is persist in few flavours but working fine in the remaining flavours is k.a., CD.

Note:
1. Performing testing of the same s/w on different browsers is k.a., Browser compatiblity OR Cross Browser Testing
2. Performing testing of the same s/w on different O.S is k.a., O.S compatiblity OR Cross O.S Testing

---

Project Class Questions:
Q: What are the browsers you have used?
Q: What are the OS you have used?
Q: What are the behaviour difference of the s/w acorss different OS & Browsers (both manual & Automation)?
Q: What are the challanges you have faced while performing CT (both manual & Automation)?

Q: Who used to deploy the s/w?
Q: Which tools you have used to automate CT?
Q: Tell me 3-4 critical functionalities OR features in your application?
Q: Time taken to perform CT both manual & Automation separately

---

Sanity Testing:

Sanity testing is the subset of regression testing and it is performed when we do not have enough time for doing testing.

A subset of regression test cases are executed after receiving a build or code with small or minor changes in the functionality or code, to check whether it resolved the issues or software bugs and no other software bug is introduced by the new changes.

Sanity testing is performed after the build has cleared the smoke tests and has been accepted by QA team for further testing. Sanity testing checks the major functionality with finer details.

Sanity testing is performed when the development team needs to know quickly the state of the product after they have done changes in the code, or there is some controlled code changed in a feature to fix any critical issue, and stringent release time-frame does not allow complete regression testing.

Q: Difference between Smoke & Sanity testing?

See the "Smoke & Sanity.xlsx"

Smoke & Sanity.xlsx

Note:

1) Both sanity tests and smoke tests are ways to avoid wasting time and effort by quickly determining whether an application is too flawed to merit any rigorous testing.

2) Sanity testing performed on a particular build is also known as a build verification test.

3) As per the needs of testing, you may have to execute both Sanity and Smoke Tests in the software build. In such cases, you will first execute Smoke tests and then go ahead with Sanity Testing.

User Acceptance Testing (UAT):

It is an end-to-end testing in which the actual business scenarios will be tested by the customer in customer environment (Stagging) is k.a., UAT.

- It is also k.a., UAT OR Acceptance Testing

Usually customer will have 2 environments

(a) Staging OR pre-production

(b) Production OR PROD environment

Once final tesitng is completed and the s/w is ready for the release, then we place all the deliverables into FTP machine & sendout a mail (Release mail) to the customer.

The release mail includes:

(a) FTP server name/IP address

(b) Credentials

(c) Location in which deliverables are kept

(d) List of deliverables kept in FTP machine.

Q: What is Deliverables?

Ans: Any artifacts which will be shared with the customer is k.a., Deliverables.

Common Deliverables would be:

1. Software (It consist of Source Code(.jar files), pages/jsp (.war file) & DB tables (.sql files)
2. Release Note Doc
3. Installation Guide Doc
4. Consolidated Test Execution Report
5. Consolidated Defect Report

Whenever we make a release to the customer, the customer will install the s/w in staging environment by following the installation guide doc. Once the installation/deployment of s/w is done, they will perform one round of end-to-end testing in staging environment which is k.a., UAT.

Q: What are the types of System testing's?
Ans:
1. System Testing:
   - It is done by QA in testing environment.
   - It is an End-to-End testing
   - Actual customer business scenarios are tested
2. UAT:
   - Its done by customer in staging environment
   - It is an End-to-End testing
   - Actual customer business scenarios are tested

During UAT testing in staging Environment, If UAT goes fine then the same s/w will be moved to PROD environment. But If customer finds any defects in UAT then the defect will be escalated to the company.

Q: What is defect Leakage?
Ans: Finding the defects by the customer in customer environment which was missed out by the QA is k.a., Defect Leakage.

Q: How to handle the escalations OR What QA has to do when customer finds the issues in PROD OR Stagging environment?
Ans:
I. Check whether the issue is part of the current release scope
  Ans: If No, then you just update the issue with below comments "This issue will be addressed in the next upcoming release as it is not part of the current release"
  But If the answer is Yes, then perform 2nd step.
II. Try to replicate/reproduce the same issue in QA environment.
  Ans: If No, then update the issue with below comments "The issue is not reproducible in QA environment. Hence closing the defect". Also provide screenshot OR server logs to proove the same.
   But If the issue is reproducible in QA environment then we need to perform 3rd step.
III. Check whether Test cases are written & selected for testing.
  If No, then you will asked to write the testcases for the same feature.
  If Yes, Action will be taken against the QA.

Different ways of doing UAT:
1. Done by customer
2. Done by customers customer (Inherent Acceptance Testing)
3. Done by QA.

1. Done by customer: It is the widely used scenario. When we make a release to the customer, the customer will deploy the s/w in staging environment & perform UAT.
2. Done by customers customer: In some business scenario's the UAT will be done by customer's customer.
EX: KSRTC has given requirement to wipro for developing online bus reservation s/w. Wipro has developed the s/w, tested it & now making a release to the KSRTC.

The KSRTC will not perform UAT instead they will give the s/w to the outlet dealers who are the customer of the KSRTC where they are also booking KSRTC buses.

Once the s/w is given to outlet, they will start using the s/w for booking the KSRTS buses. When they are using the s/w If they find any defects it will be reported back to KSRTC.

KSRTC consolidate all the complaints form their customer & send it to wipro as a UAT defects. This is k.a., Inherent Acceptance Testing.

3. Done by QA: It is not recommended. If situation demands QA has to do UAT by following User Acceptance Life Cycle:

1. Identify the need for UAT
   |
   V
2. Get customer approval
   |
   V
3. Write UAT test cases.
   |
   V
4. UAT test case review
   (a) Internal Review (Done by BA, Dev & Peer)
   (b) External Review (Done by customer)
   |
   V
5. QA should fix all the review comments, baseline the test cases and store the test cases in test case repository.
   |
   V
6. whenever Release comes, select the UAT test cases & execute them in Customer stagging environment.

Performance Testing (PT):

Testing the stability & Response Time of an application by applying Load is k.a., PT.

The performance testing will be done for web applications (any application which opens through browser) only.

Stability: Ability to withstand for which it is designed for is k.a., stability.

Response Time (RT): A total time taken to receive the link or page is k.a., Response Time.

RT = T1 + T2 + T3

But T1 & T3 are influenced by many factors. Such as:

1. Network speed
2. Network Traffic
3. Service Provider
4. Hardware capacity of the computer.

Hence we need to standardize the T1 & T3 before performing PT. While publishing the PT results also we should say under the specific speed of internet & specific Hardware capacity the application response time is 2 sec.

Load: Number of users working on the Software.

Types of PT:

1. Load Testing
2. Stress OR Torture Testing
3. Volume Testing
4. Soak Testing

1. Load Testing: Testing the stability and Response Time of an application by applying load which is equal OR less than the designed number of users is k.a., Load Testing.

---

2. Stress OR Torture Testing: Testing the stability & Response Time of an application by applying the load which is more than the designed number of users is k.a., Stress/Torture Testing.

This type of testing is usually done to find the scalability limit for the s/w.

It can be done in 2 ways:

1) Restrict the load & reduce the hardware capacity

2) Restrict the hardware & increase the load.

Note: The stress testing must be followed by Load testing.

---

3. Volume Testing: Testing the stability & response time of an application by transforming huge volume of data through it is k.a., Volume Testing.

---

4. Soak Testing: Testing the stability & Response Time of an application by applying the designed number of load continuously over a period of time is k.a, Soak Testing.

It is required to simulate the real time scenario where the continuous load will be present on the servers.

To do that we continuously apply the load to the server over a period of time

Ex: 72, 92 OR 120 hours.

Note:

1. Not all the application requires PT. We do PT on those s/w which provides ROI (Return On Investment)

2. Not all the pages requires PT. We do PT on those pages which are frequently accessed by the end users.

3. Based on Architecture the web applications are devided into 2 types

(a) Client Server Architecture-> 2 tier architecture

(b) Web Application-> n tier architecture

Both requires PT if there is a ROI.

4. Based on the client resource consumption the applications are devided into 2 types

(a) Thin client: The application which uses less client resource & whole process takes place at server side.

Ex: Gmail. FB etc

(b) Thick client: The application which utilizes more client resources and as well as server resources.

Ex: Yahoo messanger, Skype

Both requires PT if there is a ROI.

Tools used for PT are:
Load Runner
Jmeter
SoapUI
Silk Performer
Ex:
(a) Through RDC (Remote Desktop Connection) access the customer staging environment & perform the testing.
(b) If it is a web application OR Client server application then access the s/w which is deployed in staging environment through the URL.

Globalization Testing (GT):

Testing the application, which is designed for multiple languages OR regions is k.a., GT

GT is devided into 2 groups:
(a) Internationalization OR I18N testing
(b) Localization OR L10N testing

(a) Internationalization OR I18N testing: Testing the application which is designed for multiple languages is k.a., I18N testing.

(b) Localization OR L10N testing: Testing the application to verify that the country standards are displayed correctly is k.a., L10N testing.

Usually the software will be developed for gloabal language i.e., English
Developer will create English property file. This file contains english content which has to be displayed on the page.
Similarly we need to develop different property OR .yml files for different languages.

Q: Who will develop the property OR yml files?
Ex: Hindi property file.
Ans: Language experts. These are the certified people as language experts. These people must know English + their local language.
The English property file will be given to the corresponding language experts and ask them to create a property of corresponding by referring to the english property file.

Once all the property files are ready, all of them will be deployed in the server. The code is writen in such a way that if the request is for english, then connect to english property file and so on.
Note: For the applications which are having static content, we can test by adding prefix & suffix
But for the applications which are having dynamic content then cannot test by adding prefix & suffix. Hence we use tools

Common Issue in GT:

1. The selected language is not displayed
2. Content overlapping
3. Issue due to language type
   Ex: Uni directional language
   Bi-Directional Languages
4. Image should not contain text in it. If present we need to have multiple images for different languages.

5. The content should not be hardcoded in the source code. Instead the content should be hardcoded inside the property files.

Usability Testing (UT):

Testing the User friendliness of an application is k.a., UT.

User friendliness: Effort needed to learn the s/w or application is k.a., User friendliness

In UT we don't use the term test cases. Instead we use the term "CheckList".

Sample UT check Lists:

1. The look & feel should be good
2. All images should have alt tag
3. Page should not have big horizontal OR vertical scrolls instead it should have pagination.
4. All pages should have bread scrumb.
5. All the features should be accessed thorugh keyboard shortcuts
6. TAB order should work fine.
7. The text used should be very simple.
8. All pages should have Back & Next button
9. The clicked link color should get change.
10. All the important features should be located within 3rd depth
11. Text format & size should be uniform across the application
12. All the important links should be located in the left side.
13. Cursor should blink in the user name field
14. Logout link should be visible.

Different ways of performing UT:

1) Done by end user
2) Done by Usability experts
3) Done by QA

1) Done by end user:

Most of the time the gaming s/w undergoes UT. We invite the trusted end user to the company premises & ask them to perform UT. We can capture the user actions with the help of few tools viz., Camptasia, Jing, webEx etc.

The end user will perform testing, after testing he will share his findings OR defects to developer.

2) Done by Usability experts:

Usability experts are the certified people in UT. When they have to perform UT, they will call all the end users and ask them to use the s/w (Especially gaming s/w). During usage, If people faces any issues, they will report to Usability Experts. Usability experts picks up the actual defects and ask dev to fix it.

3) Done by QA:

It is not recommended. But if situaltion demands the QA can also perform UT. When QA has to do UT they need to follow Usability Life Cycle.

Usability Life Cycle:

1. Identify the need for UT
   |
   V
2. Get customer approval
   |

V

3. Derived/Write the Checklist

|

V

4. Checklist reviews

(a) internal review (BA, Dev, PEER etc)

(b) External review (Customer/ end user)

|

V

5. Fix all the review comments given by the reviwer. Baseline the checklist doc & store them in repository (Its a centralized place where all the project artifacts are stores. Ex: ALM, PC, SVN, Sharepoint etc.)

|

V

6. Whenever build comes select the checklist and execute them against the each pages. During testing If any issues found, it should be reported to dev.

|

V

7. Prepare consolidated test execution report

Note:

1. Not all the application requires UT. Its done only If there is a ROI
2. Not all the pages requires UT. Its done only If there is a ROI
3. The application used by end-user must undergo UT.

Tools used for UT:

Optimizely

Qualaroo

Crazy Egg

Click Heat

Accessability Testing (AT):

Testing the s/w from physically challenged persons point of view.

America made one rule viz., ADA (American Disability Act).

Its also k.a., 508 Compliance Testing.

It is to make sure the s/w which is developed for end user should be used by physically challenged people as well.

AT should be done for web based applications (Open through browser) only.

AT is a part of Usability testing. Here also we use checklists.

Sample checklists:

1. Text colour shouldnot be in red OR green colour(bcoz 0.02% people can't recognize red & green)
2. All features should be accessed through keyboad short cut
3. TAB order should work fine
4. All images should have alt tag.

to perform AT we use tools like AATT (Automated Accessability Testing Tool). The AATT tool works only for web applications.

This tools go through the html (DOM) source code and checks each & every tags/elements.

This AATT tool checks single pages & as well as multiple pages. It is a product of PayPal. It is a open source tool.

Ex:
If <img> tag then it checks for the presence of alt tag.
If text, it checks the colour of the text etc.
Other AT tools:
DYNOMapper.com, A11Y Compliance Platform

Reliability testing (RT)

Testing the consistency & stability of the application over a period of time is k.a., RT.

RT should be done only for window OR standalone OR Desktop applications. It is not performed for web applications.
It is done to make sure the window applications uses/consumes the client hardware resources effectively.
It is possible that using the window applications (Ex: Eclipse, Photoshop, adobe pdf etc) over a period of time viz., 72 OR 92 OR 120 hours, might create a huge backup files (Ex: temporary files, cookies etc). If it exceeds the limit, your client machine (Operating System) can hang due to out of memory problems.

As part of RT we test that the window application:
- should create only min backup files
- should reuse the files which are created already
- should not create unnecessary files
- should use min client resources.

Q: How to perform RT?
Ans:
1. Take 10-15 system OR end-to-end test cases which are having core business scenarios
2. Take any automation tools which are used for window applications viz., UFT, Test Complete, AutoIT, Sikuli etc & automate all the above test cases
3. Run the test cases continuosly over a period of time viz., 72 OR 92 OR 120 hours uninterrupted. During testing analyse the local resource consumption parameters as per the requirements.

Recovery Testing (RT):

It is the ability to restart the application after integrity of application is lost is k.a., as RT

OR

It is the activity of testing how well an operation is able to recover from crashes, hardware failures and any other similar problems is k.a., as RT

Objectives of RT:

1. To ensure operation can be continued after disaster
2. RT verifies recovery process and effectiveness of recovey process.
3. Adequate backup data is preserved & kept in secure location
4. Recovery procedures are documented
5. Recovery personnel have been assigned and trained.
6. Recovery testing tools are developed & available.
   Ex: Recuva, TestDisk etc

Examples for recovery scenarios:
1. While an application is running suddenly restart the computer & afterwards check the validness of data integrity
2. When application is receiving data from the network unplug the connection cable. After some time plug the cable back & analyses the applications ability to continue

3. Restart the computer when the browser has definite number of sessions. Afterwards check that the browser is able to recover all of them.

Penitration Testing (PT):

It is the process of testing a computer system, network OR web application to find Vulnerabilities (weakness in the s/w) that an attacker could exploit is k.a., PT.

It is also k.a., Pen Testing OR white-hat-attack. Bcoz a good guy is trying to hack the s/w.

Objectives of PT:

It is to determine the security weakness.

Penitration Testing Stategies:

1. Target Testing (Lights-turned-on)
2. Internal Testing
3. External Testing
4. Blind testing
5. Double blind testing.

1. Target Testing (Lights-turned-on): In this approach the PT is done on the well known s/w. Here the PT personnel can take the help of IT team /admin/QA etc to perform the PT.

2. Internal Testing: Perform the PT on those s/w which are used OR exposed to internal employees. It is done to avoid the impact which can be done by the internal employees.

3. External Testing: Performing PT on those s/w which are exposed to external world. ex: DNS (Domain Name Server), emails, firewalls etc

4. Blind testing: a limited or no information is provided to the PT personnel who conduct PT. This takes considerable amount of time for the PT personnels to identify the location, technology, resource etc. It simulates the actual real time scenarios.

5. Double blind testing: It takes the blind test and carries it to a step further. It simulates the actual real time scenarios.

Security Testing (ST):

It is the process that determines that the confidential data stays confidential & user can perform only those tasks which they are authorized to perform is k.a., ST.

Key terms used in ST:

1. Vulnerability
2. URL manipulation
3. SQL injection
4. Cross Site Scripting (XSS)
5. Spoofing

1. Vulnerability: It is the weakness in the s/w. it can be due to presence of virus, SQL injection, presence of defects in the s/w.

---

2. URL manipulation: Some web applications interacts with the servers with the help of information stores in the client URL. The URL may contains few data in the form of query OR Path parameter.

Hackers take the URL modify the query String/parameter, send the request back to server in order to get additional information from the server.

By manipulating certain parts of a URL, a hacker can get a web server to deliver web pages that they are not supposed to have access to.

On dynamic websites, parameters are mostly passed via the URL as follows:

http://target/forum/?cat=2

The data present in the URL is automatically created by the site. When navigating normally, a user simply clicks the links proposed by the website. If a user manually modifies the parameter, they can try different values, for example:

http://target/forum/?cat=6

---
3. SQL injection: Inserting SQL statement through GUI into some query which is then executed by the server is k.a., SQL injection
---
4. Cross Site Scripting: In some intra connected network the hackers can steal the data from the adjacent computers by writing programms in their local machines is k.a., Cross Site Scripting.
---
5. Spoofing: creation of hoax-alike (looks like) website/emails is k.a., spoofing.

How to prevent:
1. Password cracking mechanism:
   The password should be a combination of
   1. atleast 1 Upper case letter
   2. atleast 1 lower case letter
   3. atleast 1 number
   4. atleast 1 special character
   5. atleast 8-15 digits
   The password should be expire in every configured period of time.
 Ex: 45 days
   The last 4 used passwords should not be set as new password
2. How to prevent URL manipulations:
   (a) Hide the address bar
   (b) make the address bar read-only
   (c) any redirections should be rejected by the server.

3. How to prevent SQL injection:
   (a) apply length restriction to the GUI components
   (b) The GUI should not accept special characters viz., *, ',"",, etc

4. How to prevent Cross Site scripting:
    By blocking <html> & <script> tag in the webpages we can avoid XSS.
5. Spoofing : use common sense
Tools to perform ST:
web inspector (hp)
Parrot

Failover testing (FT):

Verify of redundancy mechanisms while the application is under load. This is in contrast (compare to) to load tests which are conducted under anticipated load with no component failure is k.a., FT.

FT is mainly a server side validation. To perform this we need a cluster (Multiple servers are connected) environment setup.

Here we apply the load to the server and purposefully shut down few servers during testing to make sure the load is distributed to other active/running servers in the cluster setup. So that the end user can continuously use the s/w without getting the problem even when few servers goes down.

Test Plan:

A TEST PLAN is a dynamic document describing software testing scope and activities. It is the basis for formally testing any software/product in a project.

OR

A test plan is a document describing the scope, approach, objectives, resources, schedule etc of a software testing effort. It identifies the items to be tested, items not be tested, resource, the test approach followed, training needs for team, the testing schedule etc.

Contents of Testplan:

1. Objectives
2. Scope
3. Approach
4. Testing Methodologies
5. Deliverables
6. Assumptions
7. Risk
8. Contingent OR Mitigation plan
9. Entry & Exit Criteria
10. Test Environment
11. Defects
12. Templates
13. Schedules

1. Objectives

It describes the purpose of creating the test plan doc.

---

2. Scope:

it describes the testing scope for the project.

It is 2 types:

(a) Features to be tested.

(b) Features not to be tested.

(a) Features to be tested:

List of features tested as part of functional testing:

1.
2.
3.

List of features tested as part of Integration testing:

1.
2.
3.

List of features tested as part of System testing:

1.
2.
3.

(b) Features not to be tested.

Except for the above mentioned requirements, no other requirements will be considered for testing.

OR

1. The third party features are not in testing scope
2. The feature/functionality which requires PROD setup will not be tested offshore.

---
3. Approach:
The approach which we follow to test the s/w.
It is of 2 types:
(a) By writing high level scenarios
(b) By writing flow graphs
---
4. Testing Methodologies:
What are the types of testings which we are going to perform on the applications.
(a) Functional testing
(b) UAT
(c) Compatibility
(d) Performance
(e) Security
(f) Globalization Testing
(g) Performance Testing
---
5. Deliverables:
Any artifacts which will be shared with the customer is k.a., deliverables.
The common deliverables would be:
(a) Software (which consist of source code(.jar file), pages OR jsp (.war) and DB tables (.sql) files)
(b) Release Note doc
(c) Installation Guide Doc
(d) Consolidated Test Execution Report
(e) Consolidated Defect Report
(f) Unit testing reports
---
6. Assumptions
7. Risk
8. Contingent OR Mitigation plan
All these 3 points are inter-related.
Ex: 1
*Assumption is that the 3 modules will be tested by 3 Testers.
*If any resource quits the job then there is a risk.
*Mitigation plan is that:
- M1 is a primary module for Tester1 & secondary module to Tester2
- M2 is a primary module for Tester2 & secondary module to Tester3
- M3 is a primary module for Tester3 & secondary module to Tester1

Ex: 2
* Assumption is that the test data OR DB dump will be given by customer for system testing.
* if customer does not provide the test data then there is a risk.
Mitigation plan is that:
- Ask customer to train the resource on test data preparation
- K.T (Knowledge Transfer) should be provided to the offshore team members
- Customer should Provide POC (Point Of Contact) for the offshore team members for test data related queries.
---

9. Entry & Exit Criteria

When to start & when to stop the testing's is termed as Entry & Exit Criteria.

(1a) Entry criteria for functional Testing:

1. Build should be ready with unit test results from dev.
2. Test Bed should be ready & Build should be deployed for testing.
3. Smoke testing should be completed with +ve results.
4. Functional test cases should be written, reviewed and approved.
5. Resource should be available.

(1b) Exit criteria for functional testing:

1. All the selected test cases should be executed and passed

OR

2. All the High priority test cases are executed and passed.
3. All the blocker & Critical defects should be fixed and validated.

(2a) Entry criteria for Integration Testing:

1. The exit criteria of functional testing should be met.
2. The build should be deployed in test environment.
3. The build should be ready with atleast 3-4 interfaces are implemented along with Unit test reports from dev.
4. Integration test cases are written, reviewed and approved.
5. Resources should be available.

(2b) Exit criteria for Integration testing:

1. All the selected Integration test cases should be executed and passed.

OR

2. All the High & medium test cases should be executed & Passed
3. All the blocker, critical & medium defects should be fixed and validated.

(3a) Entry criteria for system testing:

1. Exit criteria of integration tesitng should be met.
2. Build should be ready with atleast 5-6 end to end functionalities are implemented.
3. Test data OR DB dump from the customer should be ready
4. System test cases are written, reviewed and approved.
5. Resources should be available.

(3b) Exit criteria for System Testing:

1. All the selected test cases are executed and passed
2. All the defects should be fixed and validated.

---

10. Test Environment:

A software/build consist of 3 segments (Web applications)

(a) source code/business logic: (in the form of .jar file)
(b) Pages/jsp (in the form of .war file)
(c) DB tables (in the form of .sql file)

So to install the s/w:
we need to have appServer (Weblogic OR Jboss) for deploying .jar file.
We need to have webserver (Apache Tomcat) for deploying .war files.
We need DB server (Oracle, DB2, SQL Server, Mongo DB, Timber etc) to run the .sql files.

All these servers are connected to each other in a systemmatic manner. By this we can run & access the s/w.

Q: Explain application Architecture?
Ans: Explain 'n' tier architecture
Q: What is test Bed OR Test Environment OR Test Suite?
Ans: Its server OR set of servers in which the s/w is deployed for testing purpose is k.a., Test environment.
Q: Who creates a test bed?
Ans: Dev/QA/Build Engineer/Deployment Engineer can create a test bed.
** In interview preferred to say in our organization we have Build/Deployment Engineer.

Q: Who will deploy the s/w?
Ans: Dev/QA/Deployment Eng/Build Engineer can deploy the s/w.
** In interview preferred to say in our organization we have Build/Deployment Engineer.

---

11. Defects:

This section contains following:

(a) Which defect tracking tool will be used in the project
(b) Defect Severity & Its levels
(c) Defect Priority & Its levels

(a) Defect Tracking Tool: There are so many defect tracking tools are available in the market. The globally known tools are Bugzilla, JIRA, ALM, TFS, Mantis, DevTrack etc.

(b) Defect Severity & Its levels:

Defect Severity: How badly the functionality is affected is k.a., Defect Severity.

Severity Levels:

(a) Blocker/Show stopper	(a) S1	(a) Sev1
(b) Critical	(b) S2	(b) Sev2
(c) Major	(c) S3	(c) Sev3
(d) minor	(d) S4	(d) Sev4
(e) Cosmotic	(e) S5	(e) Sev5
(f) Trivial	(f) S6	(f) Sev6
(g) Enhancement	(g) S7	(g) Sev7

(c) Defect Priority & Its levels:

Defect Priority: How fast the defect should be get it fixed is k.a., Defect Priority.

Priority Levels:

(a) V. High	(a) P1
(b) High	(b) P2
(c) Medium	(c) P3
(e) Low	(d) P4
(f) V. Low	(e) P5

Q: Who will define the severity & priority while logging the defect?
Ans: In our organization the QA who logs the defect will assign the Severity & Priority.

Q: To whom you assign the defect?
Ans: In our organization we will assign the defects to the dev.

Q: How QA knows that to whom the defect should be assigned to?
Ans:
(a) In stand up meeting we can discuss
(b) In JIRA we can find the owner for the perticular requirements. If you find the defect in the requirements, check that who is the owner for that requirements & assign the defect to them
---
12. Templates:
It is a company std format used to create/write QA related documents. (Ex: Test case templates, Defect template, RTM templates etc)
OR
A system that helps you arrange information on a computer screen.
All the Test case templates, Defect template, RTM templates etc are attched in the Test plan doc. OR atleast they will mention the location in the server where all the templates are stored.
---

13. Schedules:
A tentative estimation/deadlines for all the project related tasks.
Ex:
By Jan 2019:
01st - 05th: understand all the requirements
06th - 10th: Write functional (System, Integration & Functional) test case.

Q: Who creates the Test plan?
Ans: QA should prepare the test plan especially Test Lead/Sr.QA/Manager.

Q: Have you involved in Test plan Creation?
Ans: Yes. Along with my Test Lead/ QA manager I have also involved in test plan creation.
EX: Updated Test Environment section, defect section etc
Q: Do we need to create Test plan for every Sprint?
Ans: Yes. As every sprint will have different scope so need separate test plan for each sprint.

Q: Does Dev can prepare test plan?
Ans: No. Its created by QA only.

Test Case:

A test case is a document, which has a set of test data, preconditions, expected verify compliance against a specific requirement. results and post conditions, developed for a particular test scenario in order to

OR

It is a dynamic, step by step procedural doc contains Steps, expected results, pre-condition, test data etc to validate the functionality of the given requirements.

Q: Advantages of writing the testcases?

1. Testcases provides Reusability
2. Better test coverage.
3. Consistency in test execution
4. Helps to find more defects which leads to high quality of the s/w.
5. A new comer can easily rampup on the s/w with the help of test cases.
6. Resource dependency will be avoided.
7. It is a base for the testing of the given functionality or application.

Q: What If no test cases are written?

1. There no reusability
2. No consistency in test execution
3. No better test coverage.
4. Fail to find OR miss to find the defects in the s/w
5. It is very difficult for the new comer to rampup on the application
6. Resource dependency will be more.
7. We cannot achieve better quality in the s/w application.
8. There is no base to test the s/w or application

Q: Points to be considered while writing test cases?

1. All the test cases should have all its complete steps.
2. We should write both +ve and -ve test cases.
3. Don't explain all the steps. Explain/Elaborate only those steps for which you are writting the test cases.
4. All the important words OR sentences should be highlighted either by underlining them, making them bold etc
5. All the steps in the test case must have expected results.
6. There should be a proper mapping for requirement ID's and test case ID's for easy tracking.
7. Should have appropriate pre-conditions whenever required.
8. All the test cases should be given with priority (Ex: High, Medium & Low test cases etc)
9. Write test cases with end-users perspective
10. Write test steps in a simple way that anyone can follow them easily
11. Make the test cases reusable
12. Provide a test case description, test data, expected result, precondition, postcondition.
13. Follow proper naming conventions
14. Review the test cases regularly and update them if necessary.

Test case Design Technique:

(a) Error Guessing
(b) Equivalent Partition
(c) Boundary Value Analysis (BVA)

(a) Error Guessing: Identify OR guess all the possible -ve scenario both at component level & as well as scenario/functionality level.

(b) Equivalent Partition:

According to Press men, the Equivalent partition can be done under following scenarios:

(i) When the input is a range of values then write 1 +ve & 2 -ve test cases.
(ii) When the input is a set of values then write 1 +ve & 2 -ve test cases.
(iii) When the input is a boolean value then write 1 +ve & 1 -ve test cases.

Note: But for machine critical s/w (viz., banking etc) we will devide the range into equal parts & then apply the press men rule.

(c) Boundary Value Analysis (BVA):
Derive the boundary values only.
Ex: If the input is A, then write 3 test cases
A
A+1
A-
Q: Explain the test cases templates?
Ans: In our organization we are using excel sheet for writing the test cases. The Test cases template as follows:
(a) Sl. No
(b) TestCaseID: Same as Requirement ID
(c) Description: Test case name
(d) Priority: How important this test case based on business (H, M, L)
(e) Pre-Requisite: Condition required to start the test cases
(f) Step No.: no. of steps in every test case. This is used for estimation
(g) DetailedDescription: complete Steps in the test cases
(h) ExpectedResult: Every step should have expected result.
(i) ActualResult: It should be blank while writing test case, filled while execution.
(j) ExecuteTest: What are the test cases has to be executed.
(k) Status: Passed/Failed/Blocked/Pending etc
(l) ExecutedBy: Tester name who has executed the test case.
(m) DateOfExecution: Mention the date
(n) Author: Who has written the test case.
(o) Reviewer: Who has reviewed the test cases.
(p) ExecutedOnBuild: MEntion the build number

Q: Are you still using excel?
Ans: There are so many tools are available to write the test cases viz., ALM, TFS, TestRail, JIRA, TestLink, SPIRATEST etc.
Yes. you are absolutely correct. We are planning to migrate from excel sheet to ALM in the next month. Hence training will be planned in the coming days.
Q: What is test scenario?
Ans: Writing a end-to-end workflow OR functionalities in a given requirements is k.a., Test scenario.
Q: How to proceed with writing testcases when the requirements are given?
Ans: Follow the below process to write the test cases:
(a) Understand the requirements
(b) Write the test scenarios for the given requirements. Send for reviews. Once approved,
(c) Write Test cases for all the derived scenarios by applying test case design technique.
Ex: Amount transfer is the Requirements. The example scenarios includes:
(i) Amt transfer between same bank
(ii) Amt transfer between different banks
(iii) Amt transfer using Third party apps
(iv) Amt transfer using blocked account
(v) Amt transfer with insufficient funds

After completion of writing scenarios send it for review OR set up a meeting for review. After review, write test cases for these scenarios by applying Test Case Design Techniques.

Q: Who should write the test cases?
Ans: All the QA members should write the test cases based on the User stories/requirements assigned to them.
Q: How many test cases you can write per day?
Ans: Around 15-30 testcases per day based on the complexity of the requirements.Ex: Each test cases may have around 15-40+ steps
Q: How many test cases you can execute manually per day?
Ans: Around 20-40 test cases per day based on the complexity of the requirements.
Q: What process you follow after writing the test cases?
Ans: In our organization after writing the testcases it will under go review process. First we perform internal review process. It will be done by BA, Dev, PEER etc. Once the internal review is completed & all the internal review comments are fixed, the same test cases will be sent to external review. It will be done by customer.
Q: Did you involve in reviewing the test case?
Ans: Yes. As part of PEER review I have involved in test case review process.

Q: What are the factors you conside while reviewing test cases?
Ans: Following key points has to bo considered:
(a) Test case coverage.
(b) Minimum steps with maximum coverage.
(c) Usage of proper test data for the test cases.
(d) proper pre-requisites whenever required.
(e) Usage of proper test case templates
(f) Every steps should have appropriate expected results
(g) All the test cases should have proper priority set.
(h) we make sure no duplicate test cases are written
Q: On what basis you set the priority for the test cases?
And: Based on:
(a) complexity of the functionality is depends on impact on the business.
(b) How frequently the customer uses that feature
Q: Write a test case for Pen?
Q: Write a test case for Water bottle?
Q: Write a test case for Fan?
Q: Write a test case for ATM machine?
Q: Write a test case for Paper?
Q: Write Test case for FB login?
Q: They will show you one picture of the application OR page and then ask you to write test cases for the same?


TestCaseTemplate.xlsx

Q: What is defect?
Ans: It is a deviations from the requirement is k.a., Defect.
(a) Missing implementation
(b) Wrong implementation
(c) Extra implementation

Q: What is bug?
Ans: Informal/Fancy name given to the defect is Bug.

Q: What is Failure:
A failure is the inability of a software system or component to perform its required functions within specified performance requirements
---
Q: What is error:
An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of developer we include software engineers, programmers, analysts, and testers.
(a) Syntax
(b) Logical
(c) Runtime

Q: What is Fault:
An incorrect step, process or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner. A fault is introduced into the software as the result of an error

==========
========Defect Template==========
defect Template: The important, standard set of information which we should provide while logging the defect.
The sample Defect is:
The "Send" button in compose page is not working.
The defect template includes following sections:
1. Title
2. Environment details
3. Build Details
4. Steps to replicate/Reproduce
5. Expected Result
6. Actual Result
7. Additional Info

Example:
Title: Getting "Null pointer" exception upon clicking "Send" button in the compose page after entering all the details.

Environment Details: QA Environment
App Server: 10.154.15.110 (ip address)
DB Server: 10.154.15.101 (ip address)
Web Server: 10.154.15.114 (ip address)

Build details: Provide the exact build number on which the defect is encountered
<ProjectName>_<Version>
EX: ActiTime_1.01

Steps to Replicate/Reproduce:
1. Open browser and navigate to the URL (Ex: www.gmail.com)
2. Login with valid credentials
3. Navigate to "Compose" page by clicking the "Compose" link on the left top corner
4. Enter To, cc, Bcc, subject and body & then click on "Send" button

Actual Result:
1. Getting "Nullpointer" exception upon clicking "Submit" button in the compose page after entering all the details.
2. Mail was not composed.

Expected Result:
1. The compose should be successful
2. Appropriate confirmation message should display
3. The compose window should get close
4. The mail should reach the destination without any data loss.

Additional Info:
The server error logs are attached.
The error screenshot is attached.

==========
====Questions==========

1. Who will defines the severity & priority?
Ans: In our organization the QA will defines the severity & priority.

2. To whom you will assign the defect?
Ans: In our organization we assign the defect to the respective developer.

3. How will you find the exact dev?
Ans: In standup meeting
OR
in JIRA tool the dev name will be their against the reqt.

4. What if your defect is not addressed?
Ans: bring into Scrum Master notice.

5. Does the same QA should retest his defect?
Ans: Primarily the QA who logged the defect has to retest. But it can be done by others also.

5. Can someone modify the severity & priority of the defect?
Ans: Yes. After updating the defect we must provide the appropriate comment.

Q: How to make sure the failure is a defect? OR Defect logging process?
Ans:
During test cases execution If QA finds any failure, then we should make sure the failure is a valid defect by following below steps:
1. Try to replicate the issue with different test data.
2. Try to replicate the issue in different environment OR computer.

By performing above steps we can confirm whether the failure is a valid defect OR not. Once it is confirmed that the failure was a valid issue then follow the next step
3. Check for duplicate

If nobody has logged the issue, then go ahead & log the new defect
4. Log the defect in defect tracking tool with proper defect templates.

Bugzilla URL: https://bugzilla.mozilla.org/
Here we will discuss about Defect Tracking tool BUGZILLA:
Bugzilla components:
Product/Project: name of the project
Component: the module where you found the defect.
Version: The exact build number in which you found the defect.
Severity: How badly the defect is affected.
Priority: How fast it should be fixed.
Platform: in which OS you found the defect. Its mandatory for compatibility testing.
Browsers: in which browser you found the defect. Its mandatory for compatibility testing.
Target Milestone/Fix Version: In which build the defect has to be fixed
Status: NEW
Assignee: The dev mail id to whom you are assigning the defect.
Sprint Number: The exact sprint number
QA Contact: who logged the defect OR Point of contact for the defect.
Summary: Title for the defect
Description: Defect template
Attachment: screenshot OR defect logs

Q: Project related Questions?
1. Give me min 3 examples for H. Severify & H. priority defects in your project?
2. Give me min 3 examples for H. Severify & L. priority defects in your project?
3. Give me min 3 examples for L. Severify & L. priority defects in your project?
4. Give me min 3 examples for L. Severify & H. priority defects in your project?
5. Give me examples of defects which you logged are directly affecting the business?
6. Brief, how did you justify with dev when they tell your defect is invalid?

Defect Life Cycle: (Happy Path)

When QA finds the failure, they make sure the failure is a valid defect. If its a valid defect, then log the defect against the dev in a defect tracking tool. The initial status of the defect will be NEW. While logging the defect assign the defect to respective dev who is owning that functionality.

Dev goes through the defect & changes the defect status to IN-PROGRESS. Once they finds the root cause of the issue, it will be fixed and changes the defect stutus to FIXED with appropriate comments.

The fix will be given to the QA in the next subsiquent build for retesting. During retesting If QA finds that the fix is working as expected, then close the defect by changing the defect status to CLOSED with appropriate comments.

Ex: "Validated the defect in the build <build Version> & found that the fix is working as expected. Hence closing the defect."

During retesting If QA finds that the fix is not working as expected then reopen the defect by changing the defect status to REOPEN with appropriate comments.

EX: "Validated the defect in the build <build Version> & found that the issue still persist. Hence reopening the defect."
OR
"Validated the defect in the build <build Version> & found the following observations:
The below points are working fine
1.
2.

where as the
1.
2.
are not working as expected. Hence reopening the defect.
==========
Other invalid defect status:
(1) Invalid OR Not a defect OR Rejected.
Reason:
1. Dev/QA might be refering to old requirement Doc.
2. Dev/QA might have mis-understood the requirements.
During investigation dev will change the defect status to INVALID with appropriate comments.
QA Action:
1. Go through the Dev comments carefully. If you agreed to the dev comments close the defect by changing the defect status to CLOSED with appropriate comments.
EX: "As per developer comments closing the defect".

If QA not agreed to the dev comments then reopen the issue by changing the defect status to REOPEN with proper justifications.

**(2) NOT REPRODUSIBLE:
Reason: The issue logged by the QA is working fine in Dev environment but not working in QA environment. Under such situation Dev will change the defect status to NOT REPRODUSIBLE with appropriate comments.

QA Actions:
1. Call the Dev to your desk, replicate the issue in QA environment by following the steps & show to him.
2. If reprodusible, then Provide all the additional information which dev is asking for you for better understanding of the issue.
Ex: DB details, QA environment configuration details etc.

Note: If dev is in Onsite then discuss, replicate the issue & show them through WebEx OR team viewer etc.

---

(3) Duplicate:

QA has logged one defect which was already logged by some other QA. Hence the newly logged defect becomes duplicate.

During investigation dev will change the defect status to DUPLICATE with appropriate comments.

Ex: The defect id 1251 is the duplicate of the defect id 1200.

QA Actions:

1. QA should go through Dev comments. Open both the defects and compare them carefully. If QA agreed to the dev comments then close the defect by changing the defect status to CLOSED with appropriate comments.

EX: "As per dev comments closing the defect".

2. If QA feels both are unique then reopen the issue by changing the defect status to REOPEN with proper justifications.

---

(4) Need Info:

Reason: The QA has not followed proper defect template OR has not provided proper details about the defect. Then dev will change the defect status to NEED INFO with proper comments.

QA Action: Act immediately, provide all the information what dev is asking for & then change the defect status to INFO PROVIDED.

**(5) Deffered:

Unconditionally post-poning the defect is k.a., deferred.

Reason: The defect logged by the QA is not in the current scope

OR

Dev team is very busy in fixing critical PROD issues. During such situation If QA logs the low severiy defect (which doesnot affect the business) then it will be deferred.

---

(6) Enhancement:

Any suggestions (nice to have) given by the Team members & which is not part of the requirements will be logged as Enhancements. It is also know as RFE (Request For Enhancement).

This RFE will be given to customer for approval. If customer approves it, the RFE will converted to requirements. Otherwise it will be closed.

---

(7) Cannot fix OR Product limitations:

Some time to fix the issues (which doesnot direct affect the business & moreover it has a work around for it) dev needs huge code change OR architectural changes. Which might affect the functionality of the whole product OR s/w. Under such situation we may mark the defect as CAN'T FIX.

Q: When you find the defect:

Which point is highly recommended.

(a) Log it immediately in the tool

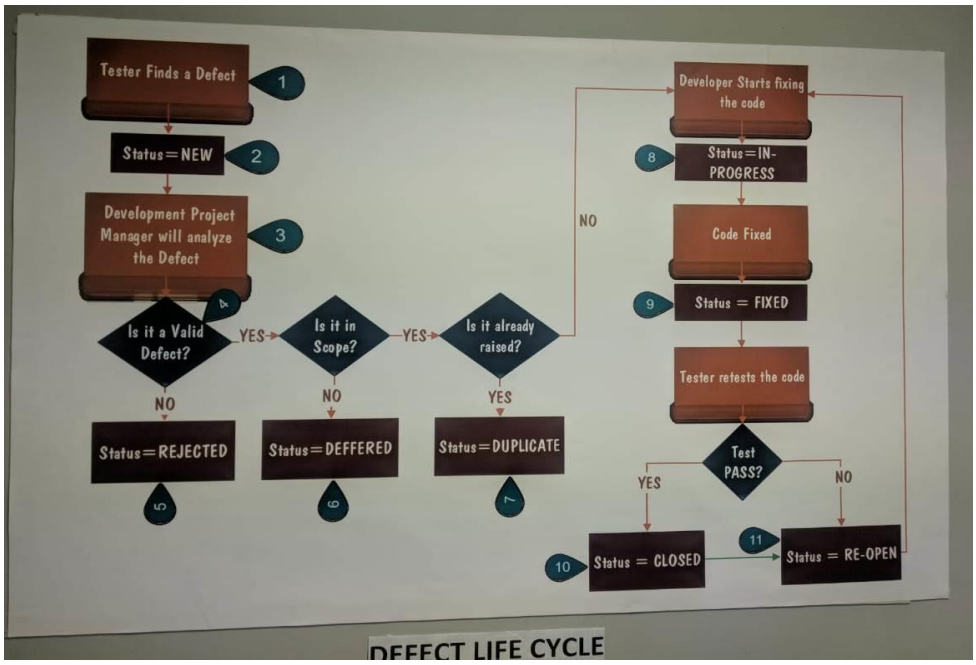
(b) Collect all the defects you found in the day & in the EOD log all of them.

(c) Dont log the issue in a tool. Just inform it in the stand up meeting
Ans: (a)

Q: Can anybody edit the defect?
Ans: Yes. Anybody can update/Edit the defect If required with proper commnets. Automatically mail will be triggered to the members who are part of that defect.
==========
Q: When QA is checking for duplicate issue, he found the same issue already logged with following status?
Case1: Found Same issue but the status is Closed
Ans: Log the new defect OR Reopen the defect

Case2: Found Same issue but the status is fixed
Ans: (a) Check the target milestone. If it is set to future build then no action for QA.
(b) If the target milestone is set to the current build then QA should reopen the defect.

case3: Found Same issue but the status is Open
Ans: Just update the comment



STLC: STLC stands for 'Software Test Life Cycle'.

It is an systematic process/approach to test the s/w. It starts from requirement collection and ends with quality deliverables.

Different phases in STLC:

1. Requirement Collection & creation of SRS doc
   |
   V
2. Understand the requirements OR system study.
   |
   V
3. Write the Test plan (written by QA Lead/ Sr. QA)
   |
   V
4. Identify the test scenarios by applying "Test Case Design Techniques" for writing the test cases (Approach: Write high level scenarios)

Note: Test scenario: It is an end-to-end workflow in the given requirements/application.

|

V

5. Write test cases for all the test scenarios

|

V

6. Create a RTM (Requirement Traceability Matrix) document. So that we can map all the requirements with the test cases

|

V

7. Test case review. All the written test cases will be sent to review. Usually review will be done in 2 cycles.

(a) Internal Review (Done by BA, Dev, PEER etc)

(b) External Review (Done by Customer)

|

V

8. QA should fix all the review comments, baseline the test case document and store them in the test case repository.

Note:

1. Baseline means finalizing the docs by fixing all the errors/review comments.
2. Test Case Repository: It is a centralized location in which all the test cases are stored.

Ex: ALM, SVN, git, Share Point, dedicated PC etc.

|

V

9. Whenever build comes, select the test cases as per the build scope (RN) & execute them against the build. During execution If you find any defects it should be logged against Dev in a defect tracking tools. (Bugzilla, ALM, TFS, JIRA etc)

Note: Please see the attachment (Execution flow.png) for the better understanding execution flow

|

V

10. Prepare consolidated test execution report

Sample Consolidated Test Execution report must have following information:

1. Total # of test cases selected : 120
2. Total # of test cases executed : 70
3. Total # of test cases Passed: 55
4. Total # of test cases failed : 15
5. Total # of test cases blocked : 0
6. Total # of test cases pending : 50
7. Total # of defects logged : 2

------ End of project from customer point of view----

But from company point of view, the team will attend the Retrospect meeting OR Project/Sprint closure meeting OR post-morton meeting:

Output of Retrospect meeting:

1. What are the best practices we have followed
2. Any process which requires improvement.
3. Any challenges faced and how did you overcome.
4. Any suggestions

The QA Manager/SM will document the retrospect discussions. So that the documentaion will act as a base for the next upcoming sprints.

RTM (Requirement Traceability Matrix)

We need a test case to test the application. Hence we have to make sure all the requirements are having the sufficient test cases so that it can be tested during our testing.

So I need a mechanism to identify that all the requirements are having test cases. The answer for this is RTM.

RTM is a matrix & it is used to map the requirements with the test cases & other parameters.

Adv of RTM:

1. It is primarily used to map the reqts with the test cases.
2. The RTM can also be used to map the reqts with other parameters viz., test cases executed, Automated , Defects logged etc.
3. RTM helps to achieve better test coverage.
4. RTM make sure all the requirements are having atleast 1 test case written
5. Team can get/Track all the project related information against each requirements in a single place.

Dis-Adv of RTM:

1. RTM doesnot tells the user that how many test cases must be written for the selected requirements.

Types of RTM:

1. Forward RTM*: We start from the requirements and map to all other parameters
2. Backward RTM: We start from other parameters and map to the requirements

Q: What is RTM?

Q: Are you using RTM in your project?

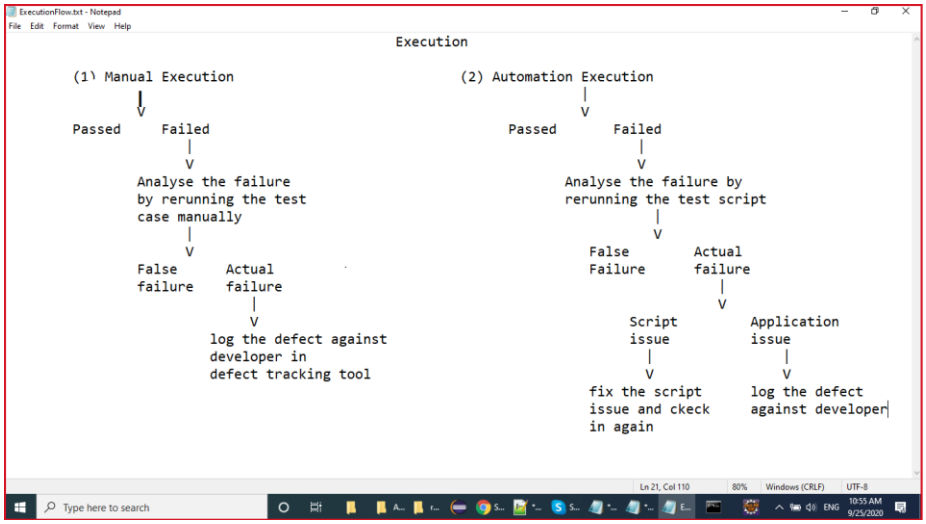
Q: What is the RTM template you are using?

RTM Template.xlsx

Q: Advantages of RTM?

Q: When do you start using the RTM in your project?

Ans: Once we start writing the test cases.



White Box Testing (WBT):

It is type of testing done by the dev to verify the source code.

To perform this type of testing we need to have understanding of complete requirements and as well as complete code knowledge. It is also k.a, Glass Box Testing OR Unit Testing.

Usually WBT will be performed by Dev. QA can also perform WBT provided they should have proper coding knowledge.

Dev will write the code then he will test their code (White box testing). Once white box testing results are passed, then they will create a build + RN and give it to QA for testing purpose.

Types of WBT:

1. Path Testing
2. Conditional Testing
3. Loop Testing
4. Testing from performance point of view
5. Testing from memory point of view
6. Integration testing

1. Path Testing:

Find/derive all the workflows OR paths for the given requirements. Take the individual workflow, combine all the code belongs to that flow and run them in a required order.

Usually to do that we use Unit testing tools (Ex: Junit & TestNG)

2. Conditional Testing:

Check each and every conditional statement in your source code whether they have properly validated OR not. Every conditional statements should be validated for both its true & false sides.

3. Loop Testing:

Make sure the source code doesnot contains any infinite loops, unwanted iterations, unwanted loops etc.

4. Testing from performance point of view:

Make sure the execution of code is very fast. To achieve this we have to do code optimization &/OR DB tuning.

5. Testing from memory point of view:

If we want to reduce the size of the source code in order to burn them in the chip we need to reduce OR remove unused variables, unused blocks/methods, unused objects etc. We need to close and release all the objects effectively to reduce the size of the source code.

6. Integration testing:

(i) Incremental Integration Testing
   (a) Top down
   (b) Bottom Up

(ii) Non-Incremental Integration Testing
   (a) Bigband
   (b) Sandwitch.