Manual Testing

CODING BUGS NOTES GALLERY

CREATE BY - ATUL KUMAR (LINKEDIN)
@atulkumarx

What is the software?

Software is a collection of computer programs that helps us to perform a task.

Types of software:

1. **System Software** (e.g., Device drivers, OS, servers, etc.)
2. **Programming Software** (e.g., compilers, debuggers, interpreters, etc.)
3. **Application Software** (e.g., Web Applications, Mobile Apps, Desktop Applications, etc.)

What is Software Testing?

1. Software Testing is a part of the software development process.
2. Software Testing is an **activity** to identify the bugs in the software.
3. The **objective/motive** of testing is to release **quality products** to the client.
4. To ensures the client about the **correctness and completeness** of the software application.

Necessity of Software Testing:

1. To avoid risks (prevent)
2. To identify errors (detect)
3. To avoid extra costs
4. To gain customer confidence
5. To accelerate software development
6. To optimize business
7. To check software adaptability

Objectives/Motive of Software Testing:

1. To **prevent defects.**
2. **Finding defects** that **may get** created by the programmer while developing the software.
3. Helps to provide a quality product.
4. To ensure that it satisfies the BRS and SRS.
5. To ensure that the result meets the **business and user requirements.**
6. Gaining confidence and providing information about the level of quality.

What is Debugging:

1. Once the Development team receives the testing team's report, they will start debugging. This phase aims to **locate** the bug **and remove** it from the software. It is a one-off process and is done manually.
2. In this process, a special tool called a debugger is used in **locating the bugs**, most programming environments have the debugger.
3. Some popular Debugger tools: WinDbg, OllyDbg, IDA Pro...

- **What is the difference between Testing and debugging?**
  - Testing means **checking** the behavior of the application whether it is expected or not. And Testing can be done without any internal knowledge of software architecture. It's done by a tester.
  - Debugging means **finding out the main cause of defects** and debugging **requires** Internal knowledge of the software architecture and coding. It's done by the developer.

- **What defines (the quality of software) Software Quality:**
  1. Bug-free
  2. Delivered on time
  3. Within budget
  4. Meets the **requirements** and/or **expectations** of the customer
  5. Maintainable

- **Psychology of testing:**
  - In software testing, psychology plays an extremely important role.
  - It is one of those factors that stay behind the scenes **but has a great impact on the end result.**
  - It is mainly dependent on the **mindset** of the developers and testers, as well as the quality of communication between them. Moreover, the psychology of testing helps them work towards a common goal.
  - **The three sections of the psychology of testing are:**
    - The mindset of Developers and Testers.
    - Communication in a Constructive Manner.
    - Test Independence.

- **QA Vs. QC:**

3P's: Every Company has 3 Pillars:

- **P - People (QC - Testers)**
- **P - Process (QA - Involves throughout development/SDLC process.)**
- **P - Product**

QA	QC
A managing tool	A corrective tool
Process-oriented	Product-oriented
Proactive strategy	Reactive strategy
Prevention of defects	Detection of defects
Everyone's responsibility	Testing team's responsibility
Performed in parallel with a project	Performed after the final product is ready

Quality Assurance:

- QA is **process-oriented**.
- QA is a **proactive process**.
- QA focuses on **preventing defects.**
- QA team **works with** the development team to produce quality software.
- QA ensures that approaches and techniques are **implemented correctly** (**during** software development).
- QA is responsible for SDLC.
- E.g., Verification

Quality Control:

- QC is **product-oriented**.
- QC is a **reactive process**.
- QC focuses on identifying/detecting the defects.
- QC comes into the picture after Quality Assurance.
- QC verifies that the developed project meets the **defined** quality standards.

- QC is responsible for STLC.
- E.g., Validation

QE (Quality Engineering):

- Quality Engineer writes the code but for the software testing purpose.
- Quality Engineers are nothing but Automation Testers.

What is QAMS?

- A quality management system is a collection of business processes focused on consistently meeting customer requirements and enhancing their satisfaction. It is aligned with an organization's purpose and strategic direction.
- A quality management system (QMS) is a system that documents the policies, business processes, and procedures necessary for an organization to create and deliver its products or services to its customers, and therefore increase customer satisfaction through high product quality.

Capability Maturity Model (CMM-Levels):

It has long been accepted that **continuous process improvement** is based on many small evolutionary steps rather than larger revolutionary innovations. The Capability Maturity Model (CMM) provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement.

This methodology is at the heart of most management systems which are designed to improve the quality of the development and delivery of all products and services.

The five Software Capability Maturity Model levels have been defined as:

1. **Initial**

   The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

2. **Repeatable**

   Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3. **Defined**

   The software process for both management and engineering activities is documented, standardized, and integrated into all processes for the organization. All projects use an approved version of the organization's standard software process for developing and maintaining software.

4. **Managed**

   Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5. **Optimizing**

   Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

==========

Project Vs. Product:

- If a software application is developed for a specific customer based on their requirement, then it is called a Project.
- If a software application is developed for multiple customers based on the market requirements, then it is called a Product.

7 Principles of Software Testing:

1. Testing shows the presence of defects, not their absence
2. Exhaustive testing is impossible
3. Absence-of-errors is a fallacy
4. Early testing saves time and money
5. Testing is context-dependent
6. Defects cluster together
7. Beware of the pesticide paradox

✓ **BETA DET**

1. **Testing shows the presence of defects, not their absence:**

- Testing can show that defects are present **but cannot prove that** there are no defects. Testing **reduces** the **probability** of undiscovered defects remaining in the software, but testing is not proof of correctness even if no defects are found.

2. **Exhaustive testing is impossible:**
   - Testing everything (all combinations of inputs and preconditions) is **not feasible** except for trivial cases.
   - Rather than attempting to test exhaustively, risk analysis, test techniques, and priorities should be used **to focus test efforts.**

3. **Early testing saves time and money:**
   - To find defects early, both static and dynamic test activities **should be** started as early as possible in the software development lifecycle.
   - Testing early in the software development lifecycle helps reduce or eliminate **costly changes**.

4. **Absence-of-errors is a fallacy**
   - Some organizations **expect** that testers can run all possible tests and find all possible defects, but this is impossible. It is a fallacy (i.e., a wrong belief) to expect that just finding and fixing a large number of defects will ensure the success of a system.
   - For example, testing all specified requirements and fixing all defects found could still produce a system that is difficult to use but does not fulfill the **users' needs and expectations.**

5. **Testing is context-dependent**
   - Testing is done differently in different contexts.
   - For example, testing in an Agile project is done differently than testing in a sequential software development lifecycle project.

6. **Defects cluster together**
   - This is the idea that **certain components or modules** of software usually **contain the most number of issues** or are **responsible for most operational failures.**
   - Testing, therefore, should be **focused on these areas**.

7. **Beware of the pesticide paradox**

- If the same tests are repeated over again and again, eventually **these tests no longer find any new defects.**
- To detect new defects, **existing test cases and test data** may need changing, and new tests may need to be written.
- Tests are no longer effective at finding defects, just as pesticides are no longer effective at killing insects after a while. In some cases, such as automated regression testing, the pesticide paradox has a beneficial outcome, which is the relatively low number of regression defects.

==========

SDLC - Software Development Life Cycle:

- SDLC is a **process** used by the software industry to **design, develop** and **test software.**
- SDLC process **aims** to **produce high-quality software** that meets customer expectations.
- The software development should be complete in the **pre-defined time frame and cost.**
- SDLC consists of a **detailed process** that explains **how to plan, build, and maintain specific software.**

Why is SDLC Needed?

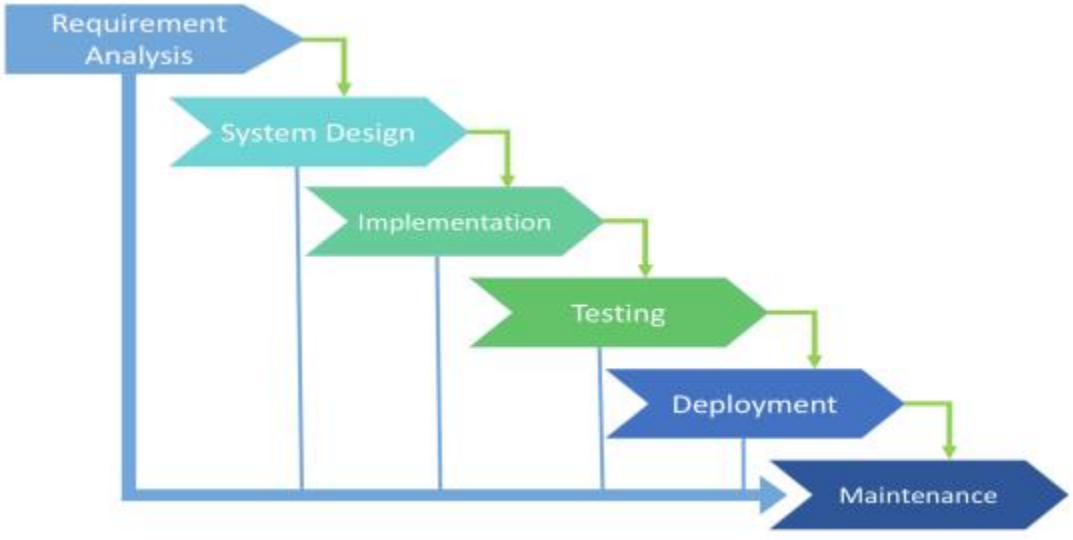
Here, are prime reasons why SDLC is important for developing a software system.

- ✓ It provides a **standard set of frameworks** for SDLC activities.
- ✓ SDLC is helpful for **project tracking and control.**
- ✓ Increased and enhanced **development speed.**
- ✓ Helps you to **decrease project risk** and project management plan overhead.
- ✓ Increases **visibility** of **project planning** to all involved stakeholders of the development process.
- ✓ Improved **client relations.**
- ✓ It offers a basis for project planning, scheduling, and estimating.

Phases in SDLC:

1. Requirement **Gathering** and **Analysis**
2. Design
3. Coding (Development/Implementation)
4. Testing
5. Deployment / Installation
6. Maintenance

RDC-TDM



1. Requirement Analysis:

- We have to **collect and understand** the requirement of the customer. Normally BA's, Project Managers, and Product Managers are involved in this phase.
- They will talk to the customer, get the requirements, and **prepare a certain number of documents (BRS/SRS).**
- This stage gives a clearer picture of the **scope of the entire project and forecasts issues.**
- This helps companies to **finalize** the necessary **timeline** to finish the work of that system.

2. Design:

- This helps to **define** the overall **system architecture.**
- In this phase, both **system and software design documents** are **prepared** as per the requirement specification document.
- This design phase serves as input for the next phase of the model.
- It contains documents like **DDS** (Design Document Specification) OR **TDD** (Technical Design Document).
- There are two kinds of design documents developed in this phase:
    a. **High-Level Design (HLD):**
        - ✓ Brief **description** and name of each module.
        - ✓ An **outline** of the **functionality** of every module.
        - ✓ **Integration** between modules.
        - ✓ **Database** tables are **identified** along with their key elements.
        - ✓ Complete **architecture diagrams** along with technical details.

b. Low-Level Design (LLD):

- ✓ **Actual Functional logic** of the modules.
- ✓ **Complete input and outputs** for every module.
- ✓ **Complete details** of the **integrations/interfaces.**
- ✓ Addresses all types of dependency issues.
- ✓ List of error messages.
- ✓ Database tables, which include type and size.

3. Coding:

- Once the system design phase is over, the next phase is coding. In this phase, developers start to build the entire system by writing code using the chosen programming language.
- In the coding phase, tasks are divided into units or modules and assigned to the various developers.
- It is the longest phase of the Software Development Life Cycle process.
- In this phase, the developer needs to follow certain **predefined coding guidelines.**
- They also need to use programming tools like compilers, interpreters, and debuggers to generate and implement the code.

4. Testing:

- Once the software is complete, it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done **to verify** that the entire application works according to the customer's requirements.
- During this phase, QA and testing team may find some bugs/defects which they communicate to developers. Then development team fixes the bug and sends it back to QA for a retest. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

5. Deployment / Installation:

- Once the software testing phase is over and no bugs or errors are left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

6. Maintenance:

- Once the system is deployed, and customers start using the developed system, the following three activities may occur:
    - ✓ Bug fixing - Bugs are reported because of some scenarios which are not tested at all.
    - ✓ Upgrade - Upgrading the application to the newer versions of the Software.
    - ✓ Enhancement - Adding some new features to the existing software.

==========

Which SDLC model is best?

- There is no model that we can consider as the best for software development process. But nowadays the
- Agile model is currently the most popular and widely used by all the software organization. In this model after
- every development stage (Sprint), the user is able to see whether the product meets his requirements. By this
- way the risks are reduced as continuous changes are done depending on client's feedback.

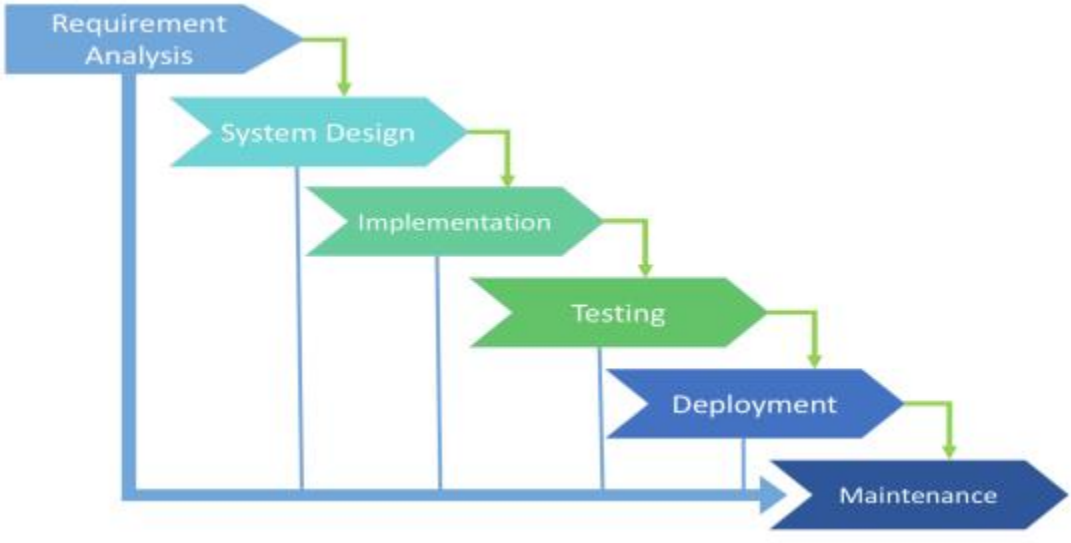
==========

Types (Models) of SDLC:

1. Waterfall Model
2. Spiral Model
3. Rapid Application Development (RAD) Model
4. Iterative or Incremental Model
5. Prototype Model
6. V Model
7. Agile Model

1. Waterfall Model:

- Waterfall is one of the earliest and most commonly used software development models (processes), in which the development process **looks like the flow**, moving **step by step** through the phases like analysis, design, coding, testing, deployment/installation, and support. So, it is also known as the "Linear Sequential Model".
- This SDLC model includes gradual execution of every stage completely. This process is **strictly documented and predefined** with features expected for every phase of this software development life cycle model.



- **Advantages of Waterfall Model:**
  1. Simple, easy to understand, and use.
  2. Easy to implement because of its linear process.
  3. Since requirement changes are not allowed, the chances of finding bugs will be less.

4. Initial investment is less since the testers are hired at the later stages.
5. Preferred for small projects where requirements are frozen (well understood at an early stage).

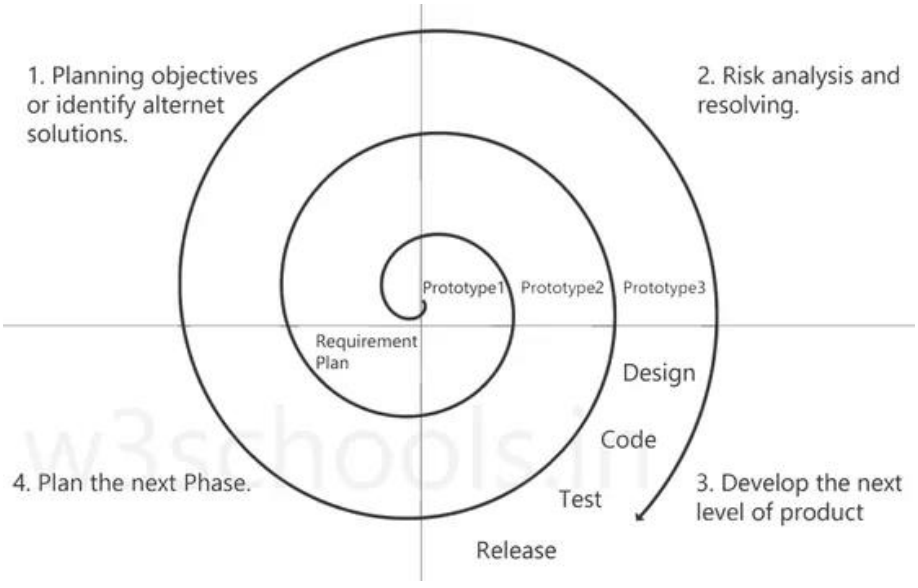
- **Disadvantages of Waterfall Model:**
  1. Requirement changes are not allowed.
  2. If there is a defect in the Requirement that will be continued in later phases.
  3. Poor model for a long and ongoing project.
  4. Testing will start only after the coding phase.
  5. Total investment is more because if the rework was done due to defects, leads to time-consuming and high investment.

- **Where we Use (Use Cases for) Waterfall SDLC Model:**
  1. The requirements are precisely documented.
  2. Product definition is stable.
  3. The technologies stack is predefined which makes it not dynamic.
  4. No ambiguous (doubtful) requirements.
  5. The project is short.

==========

2. Spiral Model: PREE

1. Spiral Model is also known as an iterative model.
2. It is a combination of the **Prototype** and **Waterfall model.**
3. Spiral Model is mostly used in **high-risk projects** (in large and complex projects).
4. It is used when the requirement is large also **unclear and complex.**
5. Spiral Model overcomes drawbacks of Waterfall Model.
6. It consists of 4 stages **Planning, Risk Analysis, Engineering, and Evaluation.**
7. In every cycle new software [module] version will be released to the customer.
8. Software will be released in multiple versions. So, it is also called a version control model.



Stages:

1. **Planning objectives** or identifying alternative solutions**:** In this stage, requirements are collected from customers, and then the aims are recognized and analyzed at the beginning of developing the project. If the iterative round is more than one, then an alternative solution is proposed in the same quadrant.

2. **Risk analysis and resolving:** As the process goes to the second quadrant, all likely **solutions are sketched**, and then the best solution among them gets selected. Then the different types of **risks linked with the chosen solution are recognized and resolved through the best possible approach.** As the spiral goes to the end of this quadrant, a **project prototype is put up** for the most excellent and likely solution.

3. **Develop the next level of product:** As the development progress goes to the third quadrant, the well-known and most required features are developed as well as verified with the testing methodologies. At the end of this third quadrant, new software or the next version of existing software is ready to deliver.

4. **Plan the next Phase:** As the development process proceeds in the fourth quadrant, the customers appraise the developed version of the project and report if any further changes are required. At last, planning for the next (subsequent) phase is initiated.

- **Prototype:** A prototype is a blueprint of the software.

  Initial requirements from the customer ---> PROTOTYPE ---> customer ---> Design, Coding, Testing....

- **Advantages of Spiral Model:**
  1. **Suitable for large projects:** Spiral models are recommended when the project is large, bulky, or complex to develop.
  2. **Risk Handling:** There are a lot of projects that have un-estimated risks involved with them. For such projects, the spiral model is the best SDLC model to pursue because it can analyze risk as well as handle risks at each phase of development.
  3. **Customer Satisfaction:** Customers can witness the development of the product at every stage and thus, they can let themselves habituate to the system and throw feedback accordingly before the final product is made.
  4. **Requirement's flexibility:** All the specific requirements needed at later stages can be included precisely if the development is done using this model.

- **Disadvantages of Spiral Model:**
  1. Requirement changes are NOT allowed in between the cycle.
  2. Every cycle of the spiral model looks like a waterfall model.
  3. There is no testing in the requirement and design phase.
  4. Does not work for smaller projects.
  5. It can be quite expensive.

==========

3. RAD Model:

RAD stands for "Rapid Application Development". As per the name itself, the RAD model is a model to develop fast and high-quality software products by Requirements using workshops.

1. Prototyping and early, reiterative user testing of designs.
2. The re-use of software components.
3. Less formality in reviews and other team communication.

==========

4. Iterative Model:

1. In an iterative model application will get divided into small parts and development will be done by specifying and implementing only small parts of the software, which can be reviewed to identify **further requirements.**
2. This process is repeated, creating a new version of the software for each cycle of the model. The iterative model is very simple to understand and use. In this model, we can't start developing the complete software with full specification of requirements

==========

5. Prototype Model:

- It is a trial version of the software. It is a sample product that is designed before starting actual testing.
- This model is used when user requirements are not very clear, and this software is tested based on raw requirements obtained from the user. The available types of prototyping are Rapid, Incremental, Evolutionary, and Extreme.
- **Prototype Model will work like --**
  1. We will take basic requirements
  2. Based on the discussion, we will create an initial prototype (A prototype – is a working model)
  3. Once the working prototype is built, we will ask the client to check and use it
  4. Next step will be to test and enhance
  5. Again, we will call the user to check and use it, and again we will make changes as per the user's feedback **until** we get all the requirements from the user.
  6. Once, all requirements are fulfilled and the client will agree, then the last step will be signed off (Sign off - Deliver the product and finish the contract)

Advantages of Prototype Model:

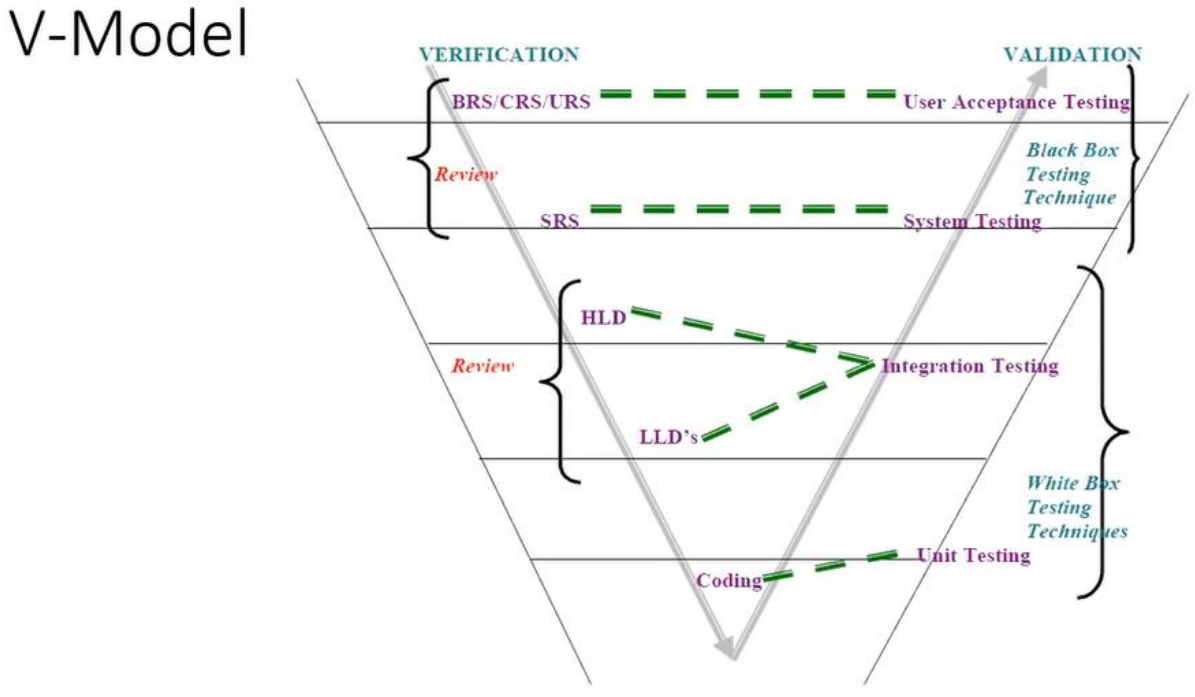
1. Users are actively involved in the development
2. Missing functionalities can be identified easily
3. Based on user feedback, the SRS document is finalized

Disadvantages of Prototype Model:

1. Sample model is not used for actual implementation
2. Scope of the system may expand beyond original plans

==========

6. **V Model:**
   - In parallel to the software development phase, a corresponding series of test phases also runs in this model. Each stage ensures a specific type of testing is done, and once that testing is passed, only then the next phase starts.
   - When the requirement is well-defined and ambiguous (uncertain), we use V-Model.
   - It is also known as Verification and Validation model.



- **Verification: Design Phase:**
  1. Verification checks whether we are building the right product (software) **(to check whether the specific requirements will meet or not).**
  2. Focus on documentation. Testing the project-related documents are called static testing.
  3. Verification does not involve the code execution.
  4. Verification typically involves Static Testing techniques like:
     - ✓ **Reviews**
     - ✓ **Walkthroughs**
     - ✓ **Inspections**
  5. **There are the various phases of the Verification Phase in the V-model,**
     - i. **BRS (Business Requirement Specification):**
       - This is the first step where product requirements are understood from the customer's side. This phase contains detailed communication to understand customers' **expectations** and exact **requirements.**
     - ii. **SRS (Software Requirement Specification):**

- In this stage **system engineers** analyze and **transfer** the business of the proposed system to SRS by studying the user requirements document (BRS).

iii. **HLD (High-Level Design):**

- High-level design (HLD) **explains the architecture** that would be used to develop a system.
- The **architecture diagram** provides an **overview** of an entire system, identifying the main components that would be developed for the product and their interfaces.

iv. **LLD (Low-Level Design):**

- In the module design phase, the system **breaks down into small modules.**
- The **detailed design** of the modules is specified, which is known as Low-Level Design

- **Coding Phase:**
  - After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

- **Validation: Testing Phase:**
  1. Validation checks whether we are building the product (software) right **(to check whether we are developing the right software).**
  2. **Focus on software.**
  3. Validation typically involves **actual testing.**
  4. Takes place after verifications are completed.
  5. Here we use the below Dynamic testing techniques:
     - ✓ Unit Testing
     - ✓ Integration Testing
     - ✓ System Testing
     - ✓ UAT

- **Advantages of the V Model:**
  1. Testing is involved in each and every phase.
  2. Every stage of the V-shaped model has strict results, so it is easy to control.
  3. Testing and verification take place in the early stages.
  4. Good for small projects, where requirements are static and clear.
  5. Testing Methods like planning, test designing happens well before coding.

- **Disadvantages of the V Model:**
    1. Documentation is more.
    2. If any changes happen in the midway, then the **test documents** along with the **required documents**, must be updated.
    3. Initial investment is more.
    4. Very rigid and least flexible.
    5. Software is developed during the implementation stage, so no early prototypes of the software are produced.

- **When to use V Model:**
    1. The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
    2. Experienced technical resources are available

- **Is Waterfall Model and V Model being the same.?:**

    V Model is the updated version of Waterfall Model.

==========

- **Static Testing Techniques:**

    We use Static Testing Techniques on the Verification side in V Model.

    1. Review
    2. Walkthrough
    3. Inspection

1. Review:

Review conducts on documents to ensure correctness and completeness.

- Requirement reviews
- Design reviews
- Code reviews
- Test Plan reviews
- Test cases reviews etc.

2. Walkthrough:

- It is an **informal** review.
- It is not pre-planned and can be done whenever required.
- Author reads the documents or code and discusses it **with peers.**
- Also, the walkthrough does not have minutes of the meeting.

3. **Inspection:**
    - It is the most **formal** review type.
    - Inspection will have a proper schedule which will be intimated via email to the concerned developer/testers.
    - In which at least 3-8 people sit in the meeting 1-reader, 2-writer, 3-moderator plus concerned people.

- **Dynamic Testing Techniques:**

  We use Dynamic Testing Techniques on the Validation side in V Model.

  1. Unit Testing
  2. Integration Testing
  3. System Testing
  4. User Acceptance Testing (UAT)

- **Why do we need testing in the SDLC process?**
  - The testing phase is very important to check whether the developed software is a quality product and matches with a design document. This stage confirms that the developed product meets the requirements defined by clients. In this stage, we perform different types of tests including unit testing, acceptance testing, system testing, etc. The software testing team works together with developers to identify and resolve software bugs and provides the Quality Application to the client.

- **Explain the software release process in brief:**
  - In the software release process, the project manager makes a release team consisting of developers, testers, project management executives. This team goes to customers' places and deploys software products or they just deploy the software at the client's server. The team also delivers some training to customers regarding the functioning of software if required.

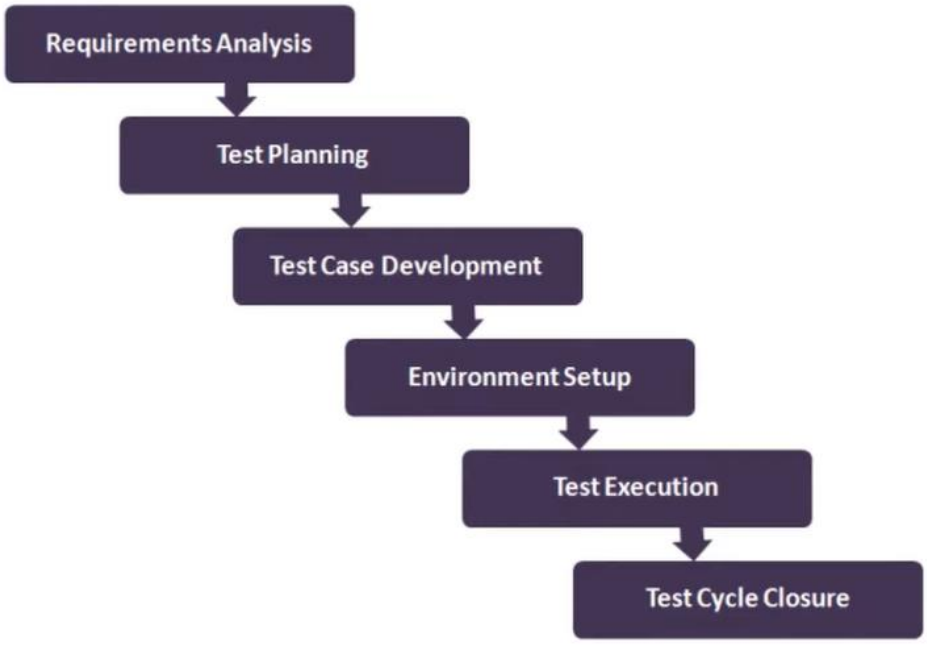
- **What is the difference between SDLC and STLC?**
  - Software Development Life Cycle **involves the complete Verification and Validation** of a Process or a Project.
  - Whereas Software Testing Life Cycle involves **only Validation.**

==========

STLC - Software Testing Life Cycle:

DESCRIPTION

1. Requirement Analysis
2. Test Planning
3. Test Design
4. Test Environment Setup
5. Test Execution
6. Test Closure



1. **Requirement Analysis:** In this phase, the requirements documents are analyzed and validated, and the scope of testing is defined.
2. **Test Planning:** In this phase, **test plan strategy** is defined, estimation of **test effort** is defined along with automation strategy and tool selection is done.
3. **Test Design:** In this phase test cases are designed; test data is prepared, and automation scripts are implemented.
4. **Test Environment Setup:** A test environment closely simulating the real-world environment is prepared.
5. **Test Execution:** To perform **actual testing** as per the test steps.
6. **Test Closure:** Test Closure is final stage of STLC where we will make all details documentations which are required submit to client at time of software delivery. Such as test report, defect report, test cases summary, RTM details, release note.

SNO	PHASE	Input	Activities	Responsibility	Out Come
1	**Test Planning**	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	**Test Designing**	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	**Test Execution**	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	**Defect Reporting & Tracking**	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	**Test Closure/Sign-Off**	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	**Test Summary Reports**
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

==========

What is Test Closure?

Test Closure is final stage of STLC where we will make all details documentations which are required submit to client at time of software delivery. Such as test report, defect report, test cases summary, RTM details, release note.

- **What is the list of test closure documents?**

  It includes,

  1. test case documents, (i.e., Test Case Excel sheet we prepare during actual testing)
  2. test plan, test strategy,
  3. Release note
  4. test scripts,
  5. test data,
  6. traceability matrix, and
  7. test results and reports like bug report, execution report etc.

- **Explain about each of test closure documents?**
  1. **test case documents:** Include detail test cases that can be reviewed.
  2. **test plan, test strategy:** Included Entire Test planning, Strategies, Schedules Etc.
  3. **Release note:** Includes all release related data like Software version, supporting hardware details, known issues, closed issues, added features etc.
  4. **test scripts:** It contain Scripts in terms of automation testing.
  5. **test data:** It contains which date you were used while testing.
  6. **traceability matrix:** traceability includes all pass and fail test cases matrix and defect records.
  7. **test results and reports like Bug Report and Execution Report:** Test Results shows entire summery on tested applications.

- **What is a Build?**
  - It is a number/identity given to Installable software that is given to the testing team by the development team

- **What is release?**
  - It is a number/ identity given to Installable software that is handed over to the customer/client by the testing team (or sometimes directly by the development team)

- **What is deployment?**
  - Deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to end-user/client/customer.

- **When will actual testing start?**

- Testing includes many activities in each phase of the development life cycle as requirement analysis, creating test plan, test design, etc. So, software testing activity should be started as soon as possible (After requirement baseline) in the development life cycle.

- **Which tools you are using to write review test cases?**
  - Generally, in our company we preferred to write down test cases in an excel sheet because it's very easy and user-friendly. After completion of the review, we will add that test cases in TestLink for future tracking purposes.
  - TestLink
    Version: -1.9.13 (Strombringer)

- **What is a different attribute of test cases?**
  1. TestCaseID - A unique identifier of the test case.
  2. Test Summary - One-liner summary of the test case.
  3. Description - Detailed description of the test case.
  4. Prerequisite or pre-condition - A set of prerequisites that must be followed before executing the test steps.
  5. Test Steps - Detailed steps for performing the test case.
  6. Expected result - The expected result in order to pass the test.
  7. Actual result - The actual result after executing the test steps.
  8. Test Result - Pass/Fail status of the test execution.
  9. Automation Status - Identifier of automation - whether the application is automated or not.
  10. Date - The test execution date.
  11. Executed by - Name of the person executing the test case.

- **What is the code freeze?**
  - Code freeze means Finalization of code and no further changes will be happening in code.

- **How would you define that testing is sufficient and it's time to enter the Test Closure phase? Or when we should stop testing?**
  - Testing can be stopped when one or more of the following conditions are met,
    1. **After test case execution –** The testing phase can be stopped when one complete cycle of test cases is executed after the last known bug fix with the agreed-upon value of pass percentage.
    2. **Once the testing deadline is met -** Testing can be stopped after deadlines get met with no high priority issues left in the system.

3. **Based on Mean Time Between Failure (MTBF)-** MTBF is the time interval between two inherent failures. Based on stakeholders' decisions, if the MTBF is quite a large one can stop the testing phase.
4. **Based on code coverage value –** The testing phase can be stopped when the automated code coverage reaches a specific threshold value with sufficient pass percentage and no critical bug.

- **What is your (QA/Tester) basic roles and responsibility?**
  - Understanding & analyzing Test Requirements.
  - Prepare Test Scenarios for individual modules.
  - Writing & review the Test Cases as per requirements.
  - Test cases execution.
  - Defect creation, Defect verification, and QA task management.
  - Weekly Test status reporting.
  - Continuous Communication with the team & client.
  - Preparation of Test Plan Documentation (if have 3+ years exp).

- **How to write effective test cases?**
  - Effective test cases have been considered
    1. Test cases should be simple to understand which helps to review and retest.
    2. Test cases should focus on end-user requirements.
    3. Test cases should be able to find defects.
    4. Test cases should contain proper steps and test data.
    5. Test cases should be Re-usable
    6. Test cases should express expected and actual results.

- **Mention what are the types of documents in QA?**
  - The types of documents in QA are
  - Requirement Document
  - Test Metrics
  - Test cases and Test plan
  - Task distribution flow chart
  - Transaction Mix
  - User profiles
  - Test log
  - User profiles
  - Test incident report

- Test summary report

- **Explain what should your QA documents include?**
  - QA testing document should include
  - List the number of defects detected as per severity level
  - Explain each requirement or business function in detail
  - Inspection reports
  - Configurations
  - Test plans and test cases
  - Bug reports
  - User manuals
  - Prepare separate reports for managers and users

==========

HERE WE NEED TO CREATE AND ADD TESTING TYPES IMAGES:

- **Types of Software Testing:**
  1. **Functional testing:** Functional testing involves validating the functional specifications of the system.
  2. **Non-Functional testing:** Non-Functional testing includes testing the non-functional requirements of the system like performance, security, scalability, portability, endurance etc.

- **Methods of Testing (Testing Methods) (White box, Black box, Grey box)**
  1. Black Box testing
  2. White Box Testing
  3. Grey Box Testing

Black Box Testing:

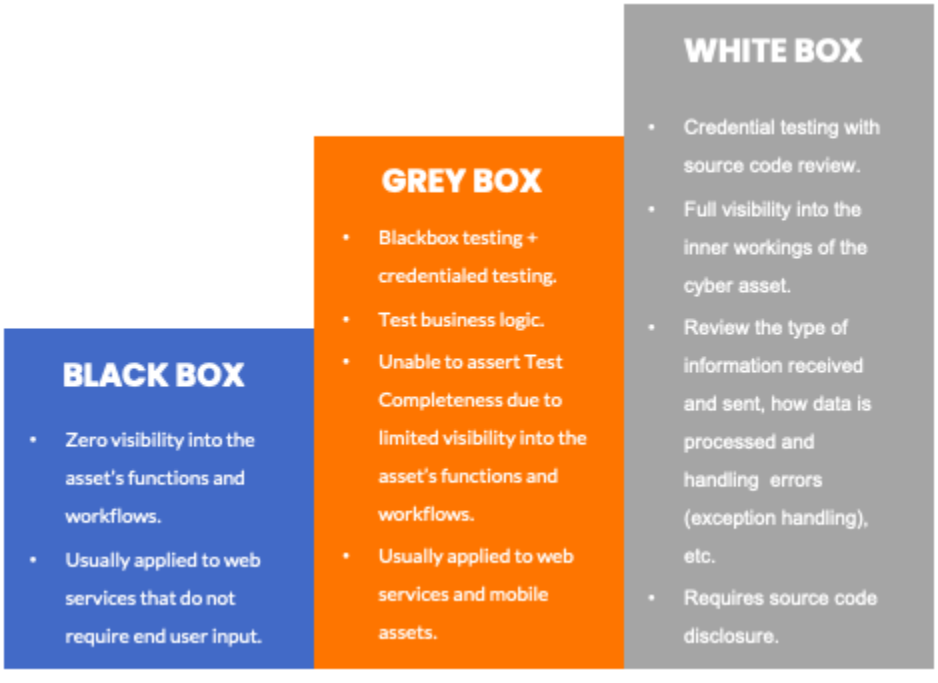
- ✓ Black box testing is the testing of requirements and functionality without knowledge of internal content. Inputs are fed into the system and outputs are determined expected or unexpected.

White Box Testing:

- ✓ White box testing is testing based on knowledge of the internal logic (algorithms) of an application's code. It's an approach that attempts to cover the software's internals in detail. White box testing is also known as 'glass box testing', 'clear box testing', 'transparent box testing', and 'structural testing'.

Grey Box Testing:

- ✓ Grey box testing uses a combination of black and white box testing. Grey box test cases are designed with knowledge of the internal logic (algorithms) of an application's code, **but the actual testing is performed as the black box.** Alternately a limited number of white-box testing is performed followed by conventional black-box testing



Levels of Software Testing:

1. Unit Testing
2. Integration Testing

3. System Testing
4. User Acceptance Testing (UAT)

1. Unit Testing:

- A unit is a single component or module of software.
- Unit testing conducts on a single program or single module.
- Unit testing is a white box testing technique.
- The developers conduct Unit testing.

- **Unit Testing Techniques:**
  - ✓ Basis path testing
  - ✓ Control structure testing
  - ✓ Conditional coverage
  - ✓ Loops coverage
  - ✓ Mutation testing.

2. Integration Testing:

- Integration testing performed between two or more modules.
- Integration testing focuses on **checking data communication** between multiple modules.
- Integration testing is a white box testing technique.
- Integration testing conducted **by the tester** at the **application level (at the UI level).**
- Integration testing conducted **by the developer** at the **coding level.**

- **Types (Techniques) of Integration Testing:**

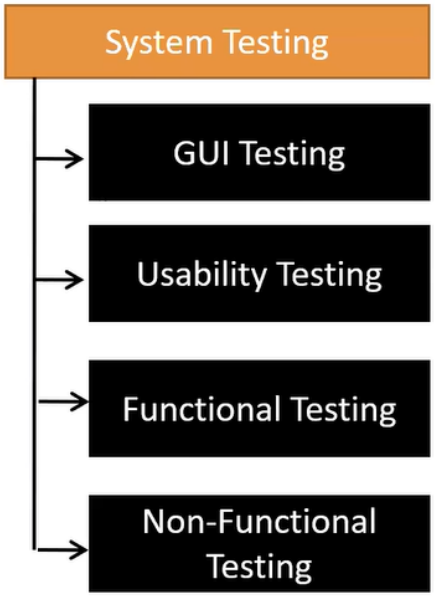
  - ✓ **Incremental integration testing**
    a. Top-down approach
    b. Bottom-up approach
    c. Sandwich/Hybrid approach

  - ✓ **Non-incremental integration testing**

3. System Testing: (This is the actual area where testers are mostly involved.)

- Testing the **overall functionality** of the application with respective client requirements.
- It is a black box technique.
- The testing team conducts System testing.
- After completion of component (unit) and integration level testing, we start System testing.

- **System Testing focuses on the below aspects (types):**
  - ✓ User Interface Testing (GUI)
  - ✓ Usability Testing
  - ✓ Functional Testing
  - ✓ Non-functional Testing



In-System Testing we both perform some functional, non-functional testing.

✓ **User Interface Testing (GUI):**

- Graphical user interface testing or GUI testing is a process of testing the user interface of an application.

- A graphical user interface includes all the elements such as menus, checkboxes, buttons, colors, fonts, icons, content, and images.
- GUI testing mostly **focuses on the front-end part.**
- Here we mainly focus on the look and feel of the application.

GUI Testing Checklist

- Testing the size, position, width, height of the elements.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen.
- Testing of the font whether it is readable or not.
- Testing of the screen in different resolutions with the help of zooming in and zooming out.
- Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
- Testing the colors of the fonts.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.
- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.
- Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs ,links etc.

✓ Usability Testing:

- During this testing validates application provide context-sensitive help or not to the user.
- Checks how easily the end-users can understand and operate the application is called usability testing.
- This is like a user manual so that the user can read the manual and proceed further.

✓ Functional Testing:

- In functional testing, we check the **functionality** of the software.
- Functionality describes **what** software **does**. Functionality is nothing but the **behavior of the application.**
- Functional testing talks about how your feature should work.
    I. Objective Properties Testing
    II. Database Testing: DML operations like insert, delete, update, select

III. Error Handling
IV. Calculation/Manipulations Testing
V. Links Existence and Links Execution
VI. Cookies and Sessions

i. Objective Properties Testing:

- ✓ Check the properties of objects like enable, disable, visible, focus, etc.

ii. Database Testing:

Checking database operations concerning user operations,

- ✓ DML operations like insert, delete, update, select
- ✓ Table and column level validations (Column type, Column length, number of columns...)
- ✓ Relation between the tables (Normalization)
- ✓ Functions
- ✓ Procedures
- ✓ Triggers
- ✓ Indexes
- ✓ Views, etc.

iii. Error Handling:

- ✓ The tester verifies the error messages while performing incorrect actions on the application.
- ✓ Error messages should be readable.
- ✓ Use understandable/simple language.

iv. Calculation/Manipulations Testing:

- ✓ The tester should verify the calculations.

v. Links Existence and Links Execution:

- ✓ Where exactly the links are placed - Links existence.
- ✓ Links are navigating to the proper or not - Links Execution
- ✓ **Types of Links:**
    - i. Internal Links

ii. External Links
iii. Broken Links

vi. Cookies and Sessions:

- ✓ Temporary files are created by Browser while browsing the pages on the internet.
- ✓ Sessions are time slots created by the server. The session will be expired after some time. (If you are idle for some time.)

Types of Functional Testing:

- Smoke
- Sanity
- Unit
- Integration
- System
- UAT (User Acceptance Testing)
- Retesting
- Regression Testing

✓ Non-functional Testing

Non-functional testing, we check the **performance** of the software under different conditions.

Types of Non-functional Testing:

a. Performance Testing
   - Load Testing
   - Stress Testing
   - Volume Testing

b. Security Testing
c. Recovery Testing
d. Compatibility Testing
e. Configuration Testing
f. Installation Testing
g. Sanitation / Garbage Testing
h. Endurance testing
i. Scalability testing.

a. **Performance Testing:** Speed of the application.
   - ✓ **Load: Gradually increase** the load on the application slowly then check the speed of the application. Here load means data.
   - ✓ **Stress: Suddenly increase/decrease** the load on the application and check the speed of the application.
   - ✓ **Volume:** Check how much data can handle by the application. Here we apply huge data to system until it gets hang. Generally, this test is performed to check how system is responding to bulk of user at a time.

b. **Security Testing:** To check how secure our application is.
   - ✓ **Authentication:** User is valid or not.
   - ✓ **Authorization / Access control:** Permissions of the valid user.

c. Recovery Testing:
   - ✓ Check the system change from abnormal to normal.

d. Compatibility Testing:
   - ✓ Forward Compatibility
   - ✓ Backward Compatibility
   - ✓ Hardware Compatibility (Configuration testing)

e. Configuration Testing:
   - ✓ It is a combination of hardware and software, in which we need to test whether they are communicating properly or not. In simple words, we check how the data is flow from one module to another.

f. Installation Testing:
   - ✓ Check screens are clear to understand.
   - ✓ Screens navigation
   - ✓ Simple or not.
   - ✓ Un-installation.

g. Sanitation / Garbage Testing

- ✓ If any application provides extra features/functionality, then we consider them a bug.

4. UAT (User Acceptance Testing):

- User Acceptance Testing (UAT) is a type of testing performed by the end-user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration, and system testing are done
- **Here are two types of UAT:**
  - **Alpha Testing:**
    Generally alpha testing is performed **in our side as per** client environment. Post Alpha Testing the software is ready for deployment into client Environment.
  - **Beta Testing:**
    Beta Testing Done into client side, where whole system is **tested as per the client Environment.** Generally, Beta and Alpha testing are done by two separate team. Largely it is done by Team who is functional user of that software. As a QA Team we assist them as per request from the client.

==========

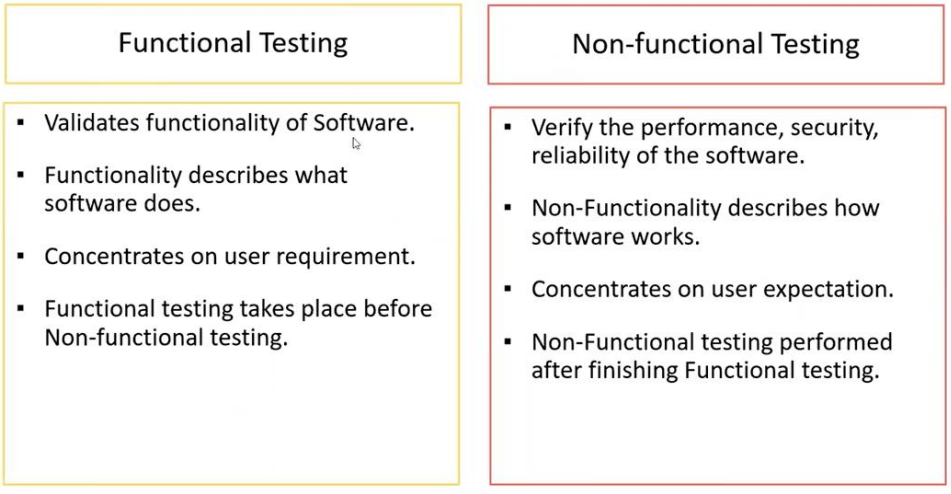
Functional Testing Vs. Non-Functional Testing:

Functional Testing:

- **Validates functionality** of Software.
- Functionality describes **what** software **does**.
- Concentrates on **user requirements.**
- Functional testing takes before Non-functional testing.

Non-Functional testing:

- **Verify** the **performance**, security, reliability of the software.
- Non-functional testing describes **how** the software **works**.
- Concentrates on **user expectations.**
- Non-functional testing performed after finishing functional testing.



==========

Test Techniques / Test Design Techniques / Test Data / Test Written Techniques:

Used to prepare data for testing.

- ✓ Data
- ✓ Coverage (cover every area/functionality of the feature)

- **Test Design Techniques (During Designing Test Cases) (for Black Box Testing):**
  1. Equivalence Class Partitioning (ECP)
  2. Boundary Value Analysis (BVA)
  3. Decision Table
  4. State Transition
  5. Error Guessing

1. Equivalence Class Partitioning (ECP):

a. Partition data into various classes and we can select data according to class then test. It reduces the number of tests cases and saves time for testing.
b. Value Check.
c. Classify/divide/partition the data into multiple classes.

Enter a Number: [ ] * Allow Digits from 1--500

Normal Test Data	Divide values into Equivalence Classes	Test Data using ECP
1		-50
2		30
3	-100 to 0 → **-50** (Invalid)	160
4	1 – 100 → **30** (Valid)	250
.	101 – 200 → **160** (Valid)	320
.	201 – 300 → **250** (Valid)	450
.	301 – 400 → **320** (Valid)	550
.	401 – 500 → **450** (Valid)	
500	501 – 600 → **550** (Invalid)	

Name: [ ] * Allow only alphabets

Divide values into Equivalence Classes

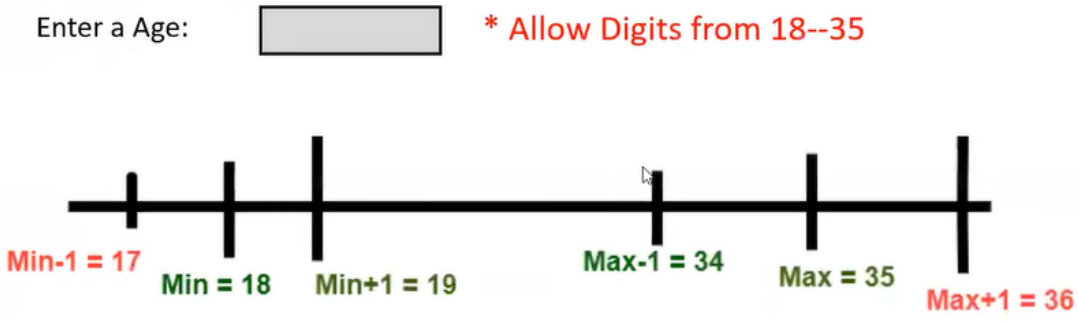
A..Z → (Valid)
a..z → (Valid)
Special Characters → (Invalid)
Spaces → **250** (Invalid)
Numbers → **320** (Invalid)

Test Data using ECP
XYZ
zyz
@#$%
Xy z
1234

2. Boundary Value Analysis (BVA):

a. Mainly we focus on boundaries of the value.
b. Here we consider 6 factors: Min, Min + 1, Min - 1, Max, Max + 1, Max – 1.

- BVA technique used to check Boundaries of the input.



Min = 18 (Pass)	Max = 35 (Pass)
Min-1 =17 (Fail)	Max-1 =34 (Pass)
Min+1 =19 (Pass)	Max+1 =36 (Fail)

***** In Input Domain Testing mostly we use ECP and BVA techniques.**

***** Input Domain testing:** The value will be verified in the textbox/input fields.

3. Decision Table:

a. Decision Table is also called a Cause-Effect Table.
b. This technique will be used if we have more conditions and corresponding actions.
c. In the Decision table technique, we deal with **combinations of inputs.**
d. To identify the test cases with a decision table, we consider conditions and actions.
e. If we have a greater number of conditions/actions, then we use the decision table technique.

Example for Decision Table:

- Take an example of transferring money online to an account that is already added and approved.
- Here the conditions to transfer money are,
    1. Account already approved.
    2. OTP (one-time-password) matched.
    3. Sufficient money in the account.

- And the actions performed are,
    1. Transfer money.
    2. Show a message as the insufficient amount.
    3. Block the transaction in case of a suspicious transaction.

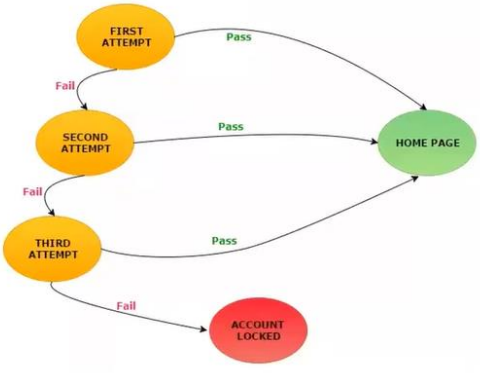
		TC1	TC2	TC3	TC4	TC5
Condition1	Account already approved	TRUE	TRUE	TRUE	TRUE	FALSE
Condition2	OTP Matched	TRUE	TRUE	FALSE	FALSE	X
Condition3	Sufficient Money in the Account	TRUE	FALSE	TRUE	FALSE	X
Action1	Transfer Money	Execute				
Action2	Show message 'Insufficient Amount'		Execute			
Action3	Block the transaction Incase of Suspicios Transaction			Execute	Execute	X

4. State Transition:

a. In State Transition Technique input is given in **sequence one step at a time.** Under this technique we can test for limited set of input values.
b. The technique should be used when the testing team **wants to test sequence of events** which happen in the application under test.
c. The tester can perform this action by entering various input conditions in a sequence.
d. In the State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.

State Transition Example

- Take an example of login page of an application which locks the user name after three wrong attempts of password.



STATE	LOGIN	CORRENT PASSWORD	INCORRECT PASSWORD
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	Display a message as "Account Locked, please consult Administrator"		

5. Error Guessing:

a. Error guessing is one of the testing techniques used to find bugs in a software application based on the tester's prior experience.
b. In Error guessing we do not follow any specific rules.
c. It depends on Tester's Analytical skills and experience.
d. Some of the examples are,
    - Submitting a form without entering values.
    - Entering invalid values such as entering alphabets in the numeric field.

==========

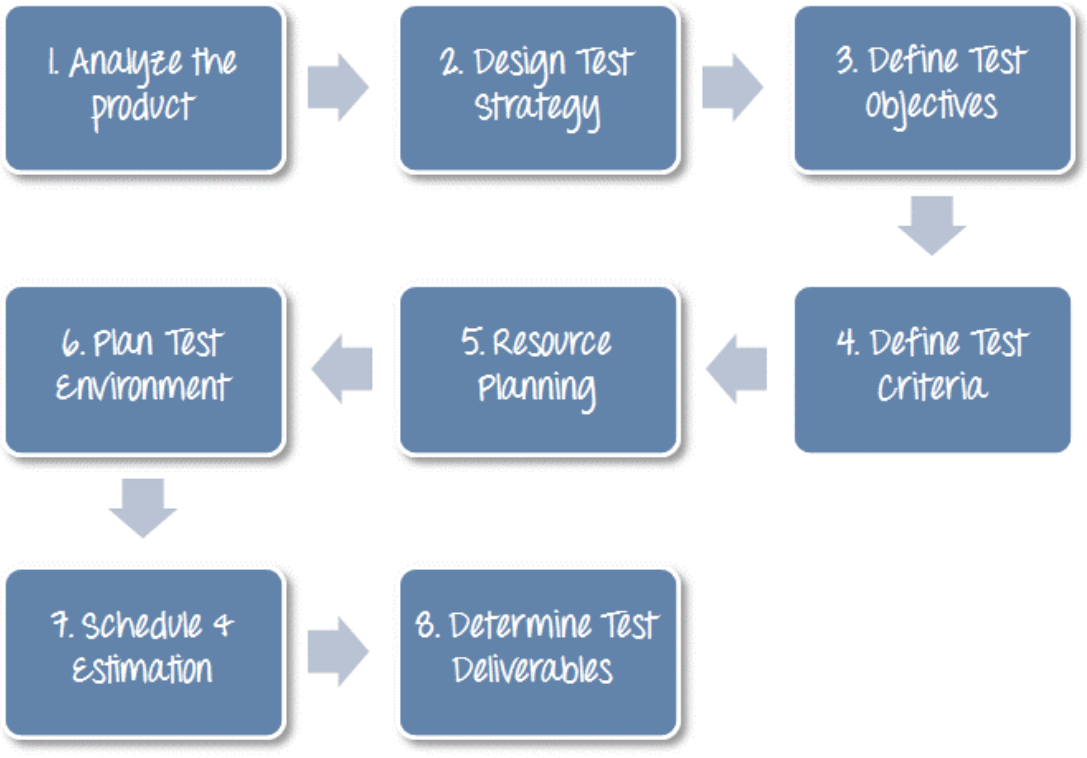
- **Test Plan Vs. Test Strategy:**

- **What is Test Plan/Test Planning:**
  - The test plan serves as a **blueprint** to conduct software testing activities as a defined process, which is monitored and controlled by the test manager.
  - A Test Plan is a **detailed/formal document** that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test.

- **Importance of Test Plan:**
  - It helps set out how the software will be checked, what specifically will be tested, and who will be performing the test. By creating a clear test plan all team members can follow, everyone can work together effectively.
  - Test Plan ensures that your Testing Project is successful, and it helps to control the risk

- **When To Write Test Plan:**
  - Software Testing should start very early stage in the project life cycle as soon as there's a Functional Requirements Document (FRD). The STLC consists of a series of phases carried out methodically to help certify the Application Under Test.

- **How to write a Test Plan:**
  - Analyze the product
  - Design the Test Strategy
  - Define the Test Objectives
  - Define Test Criteria
    - ✓ Suspension Criteria
    - ✓ Exit Criteria
  - Resource Planning
  - Plan Test Environment
  - Schedule & Estimation
  - Determine Test Deliverables



- **Test Plan IEEE 829 Format (Different Test Plan Attributes):**
  1. Test Plan Identifier
  2. References
  3. Introduction
  4. Software Risk Issues
  5. Test Items
  6. Approach
  7. Features to be Tested
  8. Features not to be Tested
  9. Item Pass/Fail Criteria
  10. Suspension Criteria and Resumption Requirements
  11. Environmental Needs
  12. Staffing and Training Needs
  13. Responsibilities
  14. Schedule
  15. Test Deliverables
  16. Approvals
  17. Glossary
  18. Remaining Test Tasks
  19. Planning Risks and Contingencies

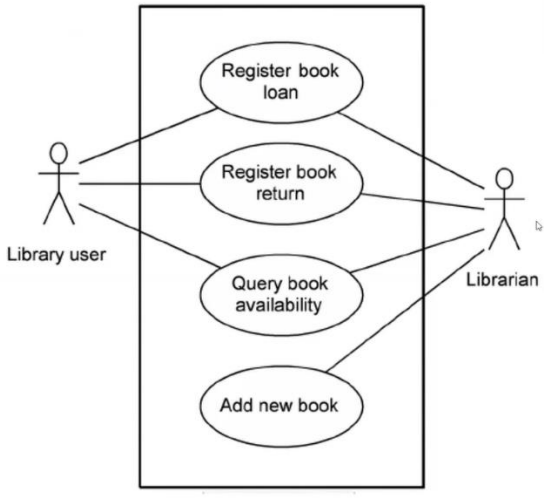
- **Test Plan template Contents:** (**WHAT** to test, **HOW** to test, **WHEN** to test):
    1. Overview
    2. Scope
        - ✓ Inclusions
        - ✓ Exclusions
        - ✓ Test Environments
    3. Test Approach/Strategy
    4. Defect Reporting Procedure
    5. Roles/Responsibilities
    6. Test Schedule
    7. Test Deliverables
    8. Pricing
    9. Entry and Exit Criteria
    10. Suspension and Resumption Criteria
    11. Tools
    12. Risk and Mitigations
    13. Approvals

==========

- **Use Case, Test Scenario, and Test Case:**

1. **Use Case: (Describes the requirement)**
    a. Use case describes the requirement.
    b. Use case contains three items.
        i. Actor, who is the user, who can be a single person or a group of people, interacting with a process.
        ii. Action, who is to reach the final outcome.
        iii. Goal/Outcome, which is the successful use outcome.

Sample Use Case



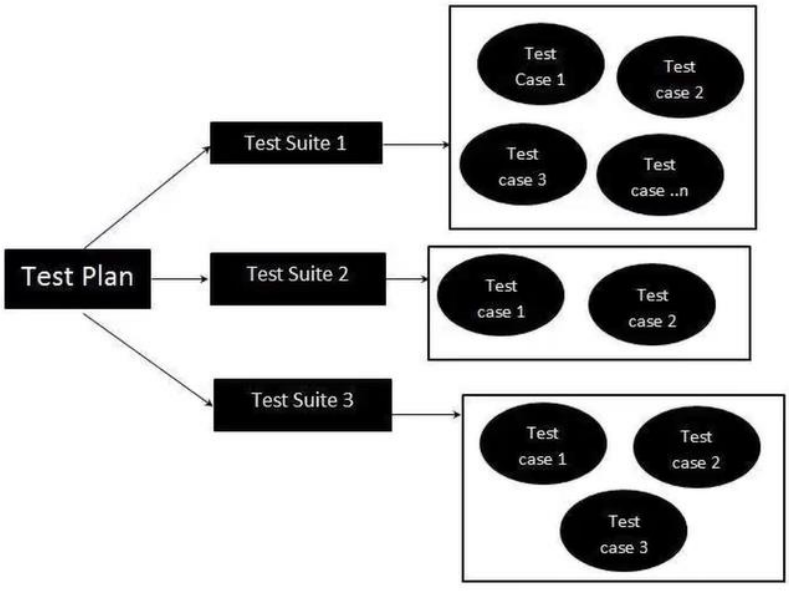
2. **Test Scenario: (What to test)**
    a. A possible area to be tested **(What to test).**
    b. Test Scenario is an overview of any functionality in a software application that is under testing.
    c. A test scenario is derived from a use case.
    d. We can say the test scenario contains all the conditions, which can be determined the testing coverage against the business requirement.

3. **Test Case: (How to test)**
    a. It is detailed description of any functionality for **how to test.**
    b. A Test Case is a **set of actions executed** to validate a particular feature or functionality of your software application.
    c. **Step by step actions** to be performed to validate the functionality of AUT (How to test).
    d. Test case contains test steps, expected result, and actual result.
    e. It contains many details like, pre-condition, test data, expected result, actual result, status etc.

- **Use Case V/s Test Case**
    a. **Use Case:** Describes functional requirement, prepared by Business Analyst (BA).
    b. **Test Case:** Describes Test Steps/Procedure, prepared by Testers.

- **Test Scenario V/s Test Case**
    a. Test Scenario is "**What to be tested**" and Test Case is "**How to be tested**".

- **Test Suite (Test Bed):**
    a. Test Suite is a group of test cases that belong to the same category.



==========

- **Test Case (Writing) Contents:**
    1. Test Case ID
    2. Test Case Title
    3. Description
    4. Pre-condition
    5. Priority (P0, P1, P2, P3) - order (P0 - Smoke and Sanity, P1 - Regression, P2 - Functional, P3 - UI)
    6. Requirement ID
    7. Steps/Actions
    8. Expected Result
    9. Actual Result
    10. Test Data

[ADD TEST CASE BHUSHAN TEMPLATE]

- **Test Case Template:**

Project Name:	**Google Email**
Module Name:	**Login**
Reference Document:	**If any**
Created by:	**Rajkumar**
Date of creation:	**DD-MMM-YY**
Date of review:	**DD-MMM-YY**

www.SoftwareTestingMaterial.com

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login	Gmail inbox is shown		
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					

==========

- **Test Case V/s Test Scenario**

TEST CASE VERSUS TEST SCENARIO

Test Case	Test Scenario
It's a set of variables or conditions which determine the viability of a software application.	It's a series of test cases executed one after the other to determine the functionality of the system or application.
It is a detailed document consisting of application requirements, preconditions, test data, post conditions and expected results.	It is a detailed test procedure consisting of test cases which help find problems in the system and evaluating results.
QA team and development team write test cases.	Reviewed by business analyst/ business manager.
It is important when development is done onsite and testing is done off-shores.	It is beneficial when time to build test cases is less.
More resources are required for writing test cases which is a waste of time and money.	It's a collaborative effort which reduces complexity thereby saving time and money.

==========

- **Test Environment:**
  1. Test Environment is a platform specially built for test case execution on the software product.
  2. It is created by integrating the required software and hardware along with proper network configuration.
  3. Test environment simulates production/real-time environment.
  4. Another name for the test environment is Test Bed.
  5. This is nothing, but an **environment created** to execute the Test Cases.

==========

- **Test Execution:**
  1. To perform actual testing as per the test steps. i.e., During this phase test team will carry out the testing, based on the test plans and the test case prepared.
  2. **Entry Criteria (Inputs):** Test Cases, Test Data, and Test Plan.
  3. **Activities:**
     - ✓ Test cases are executed based on test planning.
     - ✓ Status of test cases is marked, like passed, failed, blocked, run, etc.

- ✓ Documentation of the test results and log defects for failed cases are done.
- ✓ All the clocked and failed test cases are assigned bug ids.
- ✓ Retesting once they are fixed.
- ✓ Defects are tracked till closure.

4. **Deliverables (Outputs):** Provides defect report and test case execution report with completed results.

- **Guidelines for Test Execution:**
  1. The build being deployed to the QA environment is the most important part of the test execution cycle.
  2. Test execution is done in Quality Assurance (QA) environment.
  3. Test execution happens in multiple cycles.
  4. Test execution phase consists of Executing the test cases + test scripts (if automation).

==========

- **Requirement Traceability Matrix (RTM):**
  1. RTM Describes the mapping of Requirements with the Test cases.
  2. The main purpose of RTM is to see that all the Test cases are covered so that no functionality should miss while doing Software testing.
  3. RTM Parameters include:
     - ✓ Requirement ID
     - ✓ Requirement Description
     - ✓ Test Case IDs

Sample RTM

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

[Image: Another perfect example for RTM]

- **Which are the different types of RTM?**
  - **Forward Traceability:** This document is used to map the requirements to the test cases. This will help you in ensuring that your project is moving in the right direction. It will also ensure that all the requirements are tested thoroughly.
  - **Backward Traceability:** When you map your test cases with the requirements, you will be creating a Backward Traceability Matrix. This will help you in making sure that you are not expanding the scope of the project by adding features or functionality that was not a part of the original requirements.
  - **Bidirectional Traceability:** When you are mapping the requirements to test cases for Forward traceability and test cases to requirements for Backward traceability in a single document, it is called a Bidirectional Traceability Matrix. This document will help you in ensuring that all the specified requirements have corresponding test cases and vice versa.

- **How to create a traceability matrix for requirements?**
  1. **Set goals:** The first step you need to take before you actually create an RTM is to define your goals. It will help you in answering the question, what purpose will the RTM serve? Here is an example of a goal I want to create a traceability matrix to keep track of test cases and bugs that will be impacted if there are any changes made to the requirements.

2. **Collect artifacts:** As you have defined your goal, now you need to know which artifacts you will need in order to accomplish your goal. For creating a Requirements Traceability Matrix, you will need:
    - Requirements
    - Test cases
        - ✓ Test results
        - ✓ Bugs

    The next step will be to collect these artifacts. For this, you will have to access the latest version of requirements and make sure each requirement has a unique identification ID. You can then gather all the test cases from the testing team. If the testing is going on or it has been completed, you will have access to the test results as well as the bugs found.

3. **Prepare a traceability matrix template:** For a requirements traceability matrix template, you can create a spreadsheet in excel and add a column for each artifact that you have collected. The columns in the excel will be like: Requirements, Test cases, Test results

4. **Adding the artifacts:** You can start adding the artifacts you have to the columns. You can now copy and paste requirements, test cases, test results & bugs in the respective columns. You need to ensure that the requirements, test cases, and bugs have unique ids. You can add separate columns to denote the requirement id such as Requirement_id, TestCaseID, BugID, etc.

5. **Update the traceability matrix:** Updating the traceability matrix is an ongoing job which continues until the project completes. If there is any change in the requirements, you need to update the traceability matrix. There might be a case that a requirement is dropped; you need to update this in the matrix. If a new test case is added or a new bug is found, you need to update this in the requirements traceability matrix.

==========

- **Error, Bug/defect, and Failure:**
    - **Error:** It is a human mistake. We can say a mistake was made by the developer.
    - **Bug/defect:** It is the difference between the expected and actual results.
    - **Failure:** It comes in a customer/client environment. Where applications are unable to perform their functions.

- **Why the software has bugs?**
  - Software complexity
  - Programming errors
  - Changing requirements
  - Lack of skilled testers

- **Why does defect/bug occur?**
  - In software testing, the bug can occur for the following reasons:
    1. **Wrong coding**
    2. **Missing coding**
    3. **Extra coding**

==========

- **Defects/Bugs/Issues:**
  1. Any mismatched functionality found in an application is called a Defect/Bug/Issue.
  2. During Test Execution Test engineers are reporting mismatches as defects to developers through templates or using tools.
  3. **Defect Reporting Tools:**
     - Clear Quest
     - DevTrack
     - Jira
     - Quality Center
     - Bug Zilla etc.

- Test Management tools and Bug tracking tools are completely different.
- Test (Case) Management Tool Vs. Project Management Tool Vs. Bug Tracking Tool.

==========

- **Defect Report Contents:**
  1. Defect_ID
  2. Defect Description
  3. Version

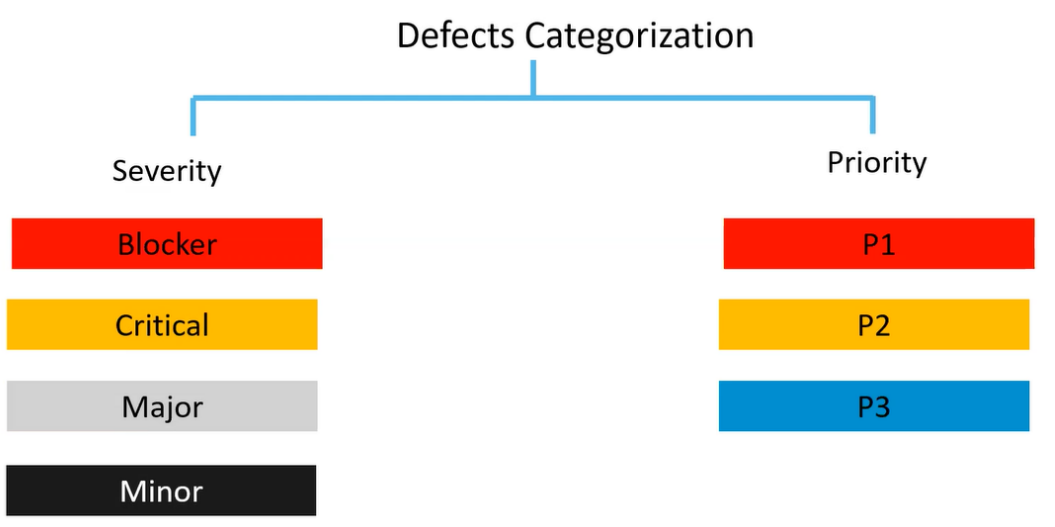
4. Steps
5. Data Raised
6. Reference
7. Detected By
8. Status
9. Fixed By
10. Date Closed
11. Severity
12. Priority

==========

- **Defect Classification / Categorization:**
  - **Severity**
    1. Blocker (Show stopper)
    2. Critical
    3. Major
    4. Minor

  - **Priority**
    1. P1 (High)
    2. P2 (Medium)
    3. P3 (Low)

Defect Classification



- **Defect Severity: S-T (S-S-T) (Severity-System)**
  - Severity is assigned/given by the QA **Testers.**
  - It affects the **functionality.**
  - Severity **describes** the **seriousness of the defect** and **how much** impact on **Business workflow (functionality).** It is categorized into Blocker, Critical, Major, Minor.

- **Defect Priority: D-P (D-I-P) (Priority-People)**
  - Priority is mostly given/assigned by the **developer.**
  - It affects business **values**.
  - Priority describes the **importance of defects.** The defect which is urgent for repair.
  - Defect Priority states **the order in which** a defect should be fixed.
  - **Defect priority can be categorized into three categories:**
    1. **P1:** The defect must be resolved immediately as it affects the system severely and cannot be used until it is fixed.
    2. **P2:** It can wait until a new version/build is created.
    3. **P3:** The developer can fix it in later releases.

[Image: Example]

==========

Defect Resolution:

After receiving the defect report from the testing team, the development team conducts a review meeting to fix defects. Then they send a Resolution Type to the testing team for further communication.

Resolution Types:

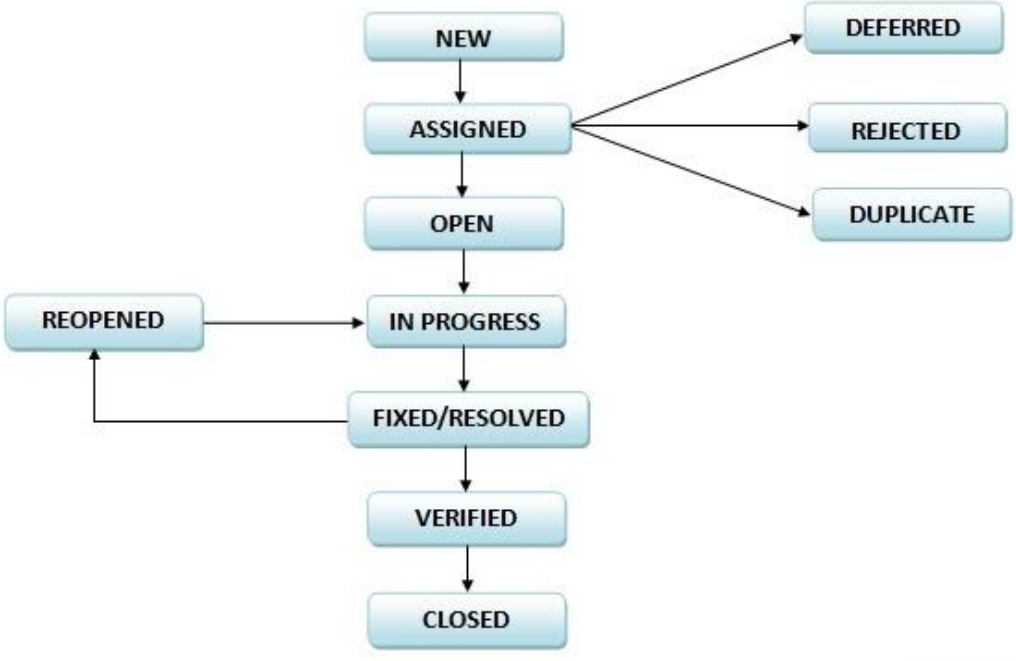
1. Accept
2. Reject
3. Duplicate
4. Enhancement
5. Need more information
6. Not Reproducible
7. Fixed
8. As Designed.

==========

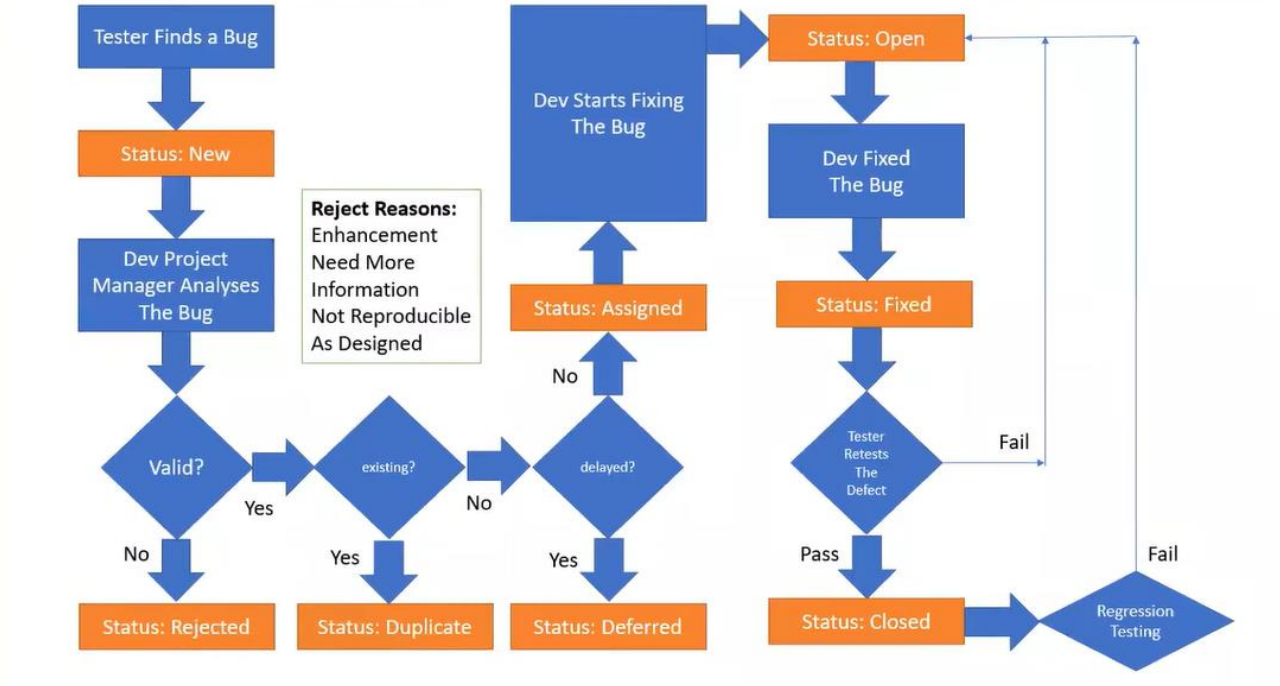
Defect Life Cycle/Bug Life Cycle:

- The Bug is the informal name of defects, which means that software or application is not working as per the requirement.
- In software testing, a software bug can also be an issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.

Defect Life Cycle



Bug Life Cycle



==========

Test Cycle Closure:

a. Activities:

- Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software Critical Business Objectives, Quality.
- Prepare test metrics based on the above parameters.
- Document the learning out of the project.
- Prepare Test summary report.
- Qualitative and quantitative reporting of quality of the work product to the customer.
- The result analysis to find out the defect distribution by type and severity.

b. Deliverables:

- Test Closure report.
- Test metrics.

c. Test Metrics:

Test Metrics

SNO	Required Data
1	No. Of Requirements
2	Avg. No. of Test Cases written Per Requirement
3	Total No.of Test Cases written for all Requirement
4	Total No. Of test cases Executed
5	No.of Test Cases Passed
6	No.of Test Cases Failed
7	No.of Test cases Blocked
8	No. Of Test Cases Un Executed
9	Total No. Of Defects Identified
10	Critical Defects Count
11	Higher Defects Count
12	Medium Defects Count
13	Low Defects Count
14	Customer Defects
15	No.of defects found in UAT

Test Metrics

- **% of Test cases Executed:**
  No.of Test cases executed / Total No. of Test cases written) * 100
- **% of test cases NOT executed:**
  (No.of Test cases NOT executed/Total No. of Test cases written) * 100
- **% Test cases passed**
  (No.of Test cases Passed /Total Test cases executed) * 100
- **% Test cases failed**
  (No.of Test cases failed / Total Test cases executed) * 100
- **%Test cases blocked**
  (No.of test cases blocked / Total Test cases executed) * 100

Test Metrics

- **Defect Density: Number of defects identified per requirement/s**
  No.of defects found / Size(No. of requirements)
- **Defect Removal Efficiency (DRE):**
  (A / A+B) * 100
  (Fixed Defects / (Fixed Defects + Missed defects)) * 100
  - A- Defects identified during testing/ Fixed Defects
  - B- Defects identified by the customer/Missed defects
- **Defect Leakage:**
  (No.of defects found in UAT / No. of defects found in Testing) * 100
- **Defect Rejection Ratio:**
  (No. of defect rejected /Total No. of defects raised) * 100
- **Defect Age:** Fixed date-Reported date
- **Customer satisfaction** = No.of complaints per Period of time

==========

- **QA/Testing Activities:**
  - Understanding the requirements and functional specifications of the application.
  - Identifying required Test Scenarios.
  - Designing Test Cases to validate the application.
  - Setting up Test Environment (Test Bed).
  - Execute Test Cases to valid applications.
  - Log Test results (How many tests cases pass/fail.)
  - Defect reporting and tracking.
  - Retest fixed defects of the previous build.
  - Perform various types of testing in the application.
  - Reports to Test Leas about the status of assigned tasks.
  - Participated in regular team meetings.
  - Creating automation scripts.
  - Provides recommendations on whether or not the application/system is ready for production.

==========

- **Software Testing Terminologies: (Other Types of Testing):**
  a. Regression Testing
  b. Re-testing
  c. Exploratory testing
  d. Adhoc Testing
  e. Monkey Testing
  f. Positive Testing
  g. Negative Testing
  h. End to End Testing
  i. Globalization and Localization Testing

a. Regression testing:

Testing conducted on modified build (updated build) to make sure there will not be an impact on existing functionality because of changes like adding/deleting/modifying features. Also, we can say Smoke Testing is a small part of regression testing.

1. Unit regression testing type:

i. Testing only the changes/modification done by the developer.

2. Regional Regression testing type:

i. Testing the modified module along with the impacted modules.
ii. Impact Analysis meeting conducts to identify impacted modules with QA and developer.

3. Full Regression type:

i. Testing the main feature and remaining part of the application.
ii. Example: The developer has done changes in many modules, instead of identifying impacted modules, we perform one round of full regression.

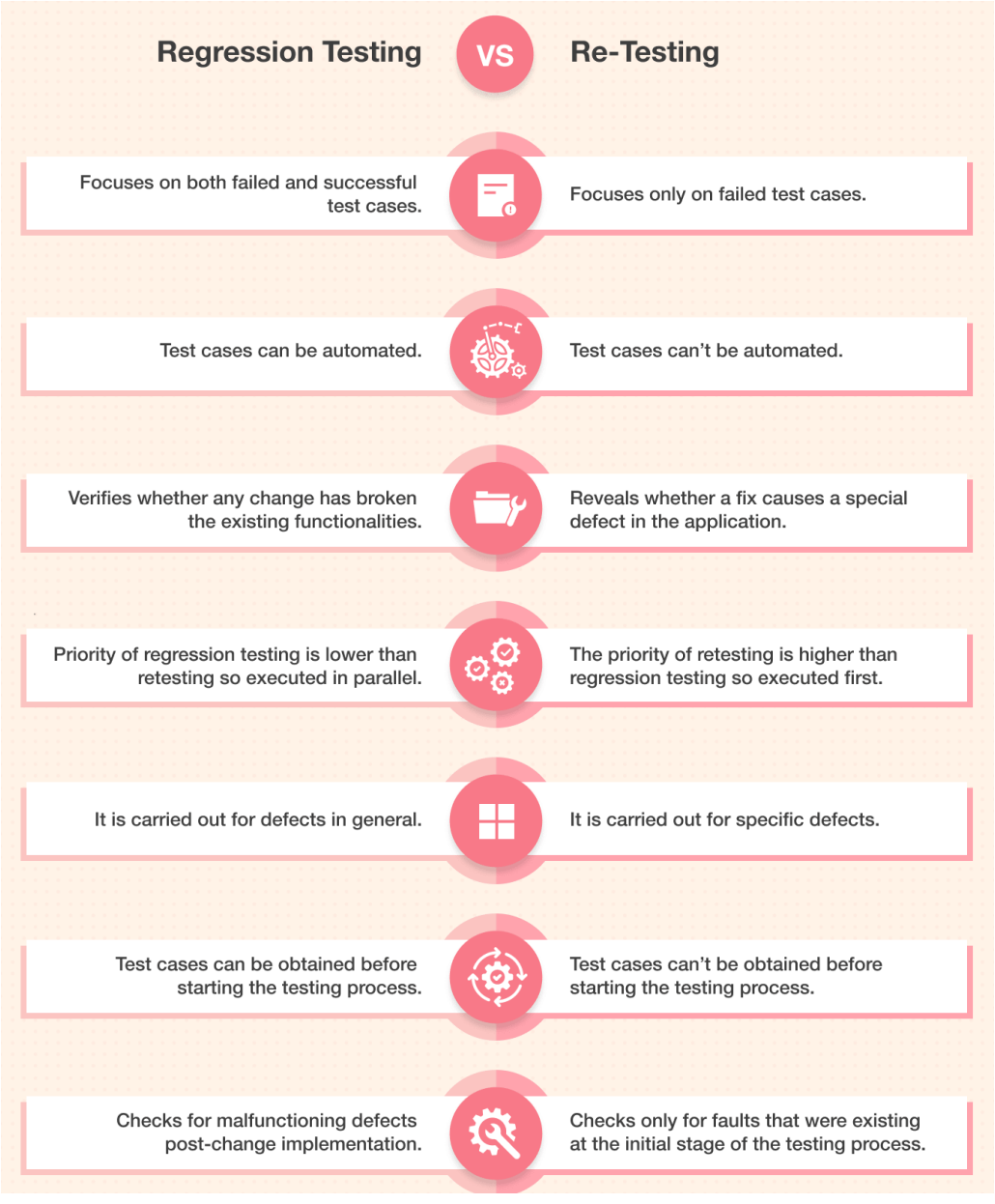
- **When we performed regression testing:**
  - ✓ After Defect get Fixed and Retesting is done.
  - ✓ Modification, Updating, change request into Existing functionality.
  - ✓ After addition of new functionality.
  - ✓ Before release.
  - ✓ After release.
  - ✓ Data Migration.

- ✓ After change in Environment

b. Re-Testing:

1. Whenever the developer fixed a bug, the tester will test the bug fix called re-testing.
2. Tester closes the bug if worked otherwise re-open and send to a developer.
3. To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
4. Example:
    i. Build 1.0 was released, test team found some defects (Defect ID 1.0.1, 1.0.2) and posted them.
    ii. Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

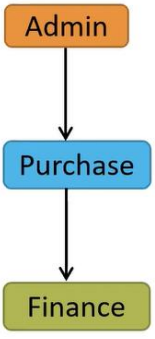
- **Re-Testing VS. Regression Testing:**



- **Example of Re-Testing Vs. Regression Testing:**

Example: Re-Testing Vs Regression Testing

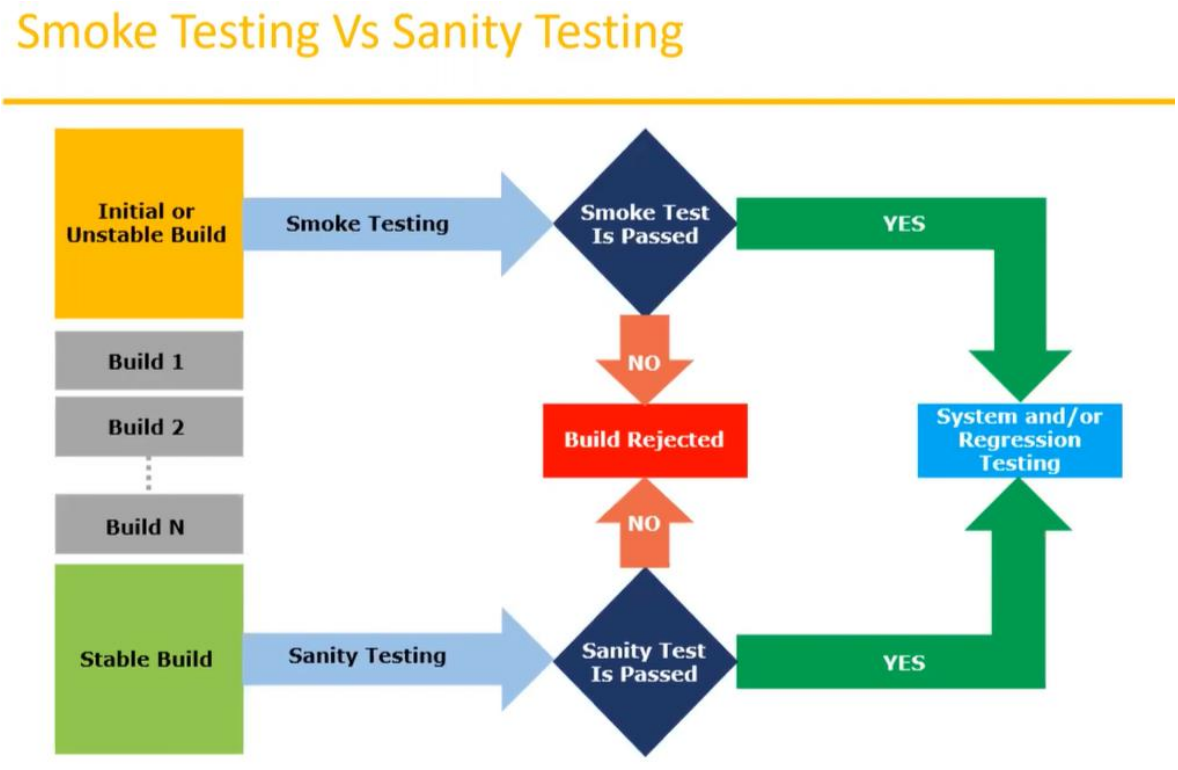
- An Application Under Test has three modules namely **Admin**, **Purchase** and **Finance**.
- Finance module depends on Purchase module.
- If a tester found a bug on Purchase module and posted. Once the bug is fixed, the tester needs to do **Retesting** to verify whether the bug related to the Purchase is fixed or not and also tester needs to do **Regression Testing** to test the Finance module which depends on the Purchase module.



Smoke Vs. Sanity Testing:

- Smoke and Sanity Testing come into the picture after build release.

Smoke Testing	Sanity Testing
Smoke Test is done to make sure the build we received from the development team is testable/stable or not	Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper.
Smoke Testing is performed by both Developers and Testers	Sanity Testing is performed by Testers alone
Smoke Testing, build may be either stable or unstable	Sanity Testing, build is relatively stable
It is done on initial builds.	It is done on stable builds.
It is a part of basic testing.	It is a part of regression testing.
Usually it is done every time there is a new build release.	It is planned when there is no enough time to do in-depth testing.



c. Exploratory testing:

- We have to explore the application, understand completely and test it.
- Understand the application, identify all possible scenarios, documents it, then use it for testing.
- We do exploratory testing when the application is ready but there is no requirement.
- Test Engineer will do exploratory testing when no requirement is.

Drawbacks of Exploratory testing:

- You might misunderstand any feature as a bug (or) any bug as a feature since you do not have requirements.
- time-consuming.
- If there is any bug in the application, you will never know about it.

d. Adhoc Testing:

- Testing application randomly without any test cases or any business requirement document.
- Adhoc testing is an informal testing type with an aim to break the system.

- Tester should have knowledge of application even though he does not have requirements/test cases.
- This testing is usually an unplanned activity.

e. Monkey/Gorilla Testing:

- Testing applications randomly without any test cases or any business requirement.
- Adhoc testing is an informal testing type with an aim to break the system.
- Tester does not have knowledge of the application.
- Suitable for gaming applications.

Adhoc Testing Vs. Monkey Testing Vs. Exploratory Testing

Adhoc Testing	Monkey Testing	Exploratory Testing
No Documentation	No Documentation	No Documentation
No Plan	No Plan	No Plan
Informal testing	Informal testing	Informal testing
Tester should know Application functionality	Testers doesn't know Application functionality	Testers doesn't know Application functionality
Random Testing	Random Testing	Random Testing
Intension is to break the application/find out corner defects	Intension is to break the application/find out corner defects	Intension is to learn or explore functionality of application
Any Applications	Gaming Applications	Any Applications which is new to tester

f. Positive Testing:

- Testing the application with valid inputs is called positive testing.
- It checks whether an application behaves as expected with positive inputs.

g. Negative Testing

- Testing the application with invalid inputs is called negative testing.
- It checks whether an application behaves as expected with the negative testing.

- **Positive Vs. Negative Test Cases:**

  For example, if a text box is listed as a feature and in FRS it is mentioned as a Text box that accepts 6-20 characters and only alphabets.

  a. Positive Test Cases:

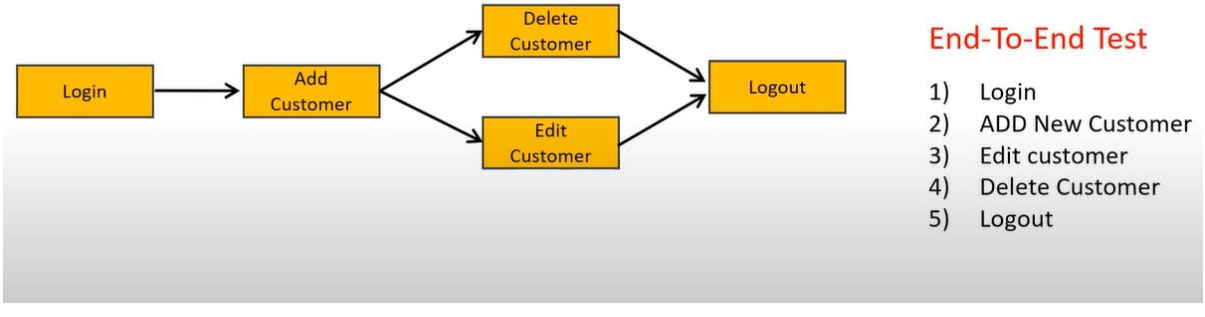
  1. Textbox accepts six characters.
  2. Textbox accepts up to 20-character lengths.
  3. Textbox accepts any value between 6-20 characters long.
  4. Textbox accepts all alphabets.

  b. Negative Test Cases:

  1. Textbox should not accept less than 6 characters.
  2. Textbox should not accept characters more than twenty characters.
  3. Textbox should not accept special characters.
  4. Textbox should not accept numerically.

h. End to End Testing:

1. Testing the overall functionalities of the system including the data integration among all the modules are called end-to-end testing.



i. Globalization and Localization Testing:

- **Globalization Testing:**
- Performed to ensure the system or software application can run in any cultural or local environment.
- Different aspects of the software application are tested to ensure that it supports every language and different attributes.
- It tests the different currency formats, mobile number formats and address formats are supported by the application.
- For example, Facebook.com supports many of the languages and it can be accessed by people of different countries. Hence it is a globalized product.

- **Localization Testing:**
- Performed to check system or software application for a specific geographical and cultural environment.
- Localized product only supports the specific kind of language and is usable only in specific region.
- It testes the specific currency format, mobile number format and address format is working properly or not.
- For example, Baidu.com supports only the Chinese language and can be accessed only by people of few countries. Hence it is a localized product.

==========

Agile – Scrum

Agile Model/Agile Methodology/Agile Process:

- **Agile is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer/end customer.**
- Both development and testing activities are done parallelly and collaboratively.
- In Agile, instead of BA's, the Product Owner collects and discusses with the customer/end client.
- It's a Quick assign Quick Design model.
- Agile model is combination of **Iterative and Incremental model.**

Agile Principles:

- Requirement changes are accepted from the customer.
- So, the customer no need to wait, which provides customer satisfaction.
- We develop, test, and release pieces of software to the customer with some features.

- The Whole team works together toward achieving one goal.
- Here we focus on F2F conversation.
- We follow iterative and incremental nature.

Advantages of Agile Model:

- We can accept/accommodate the required changes in the middle of development.
- Release will be very fast.
- So, the customers no need to wait for a long time.
- Create good communication between teams.
- Easy model to adapt.
- It is suitable for both small and large software development projects.

Disadvantages of Agile Model:

- Less focus on design and documentation, as we focus on fast delivery of the software.
- Project can easily go out of track if the customer representative is not clear about what final outcome they want.

Development Task:

- Review the story
- Estimate the story
- Design
- Code
- Unit Testing
- Integration Testing etc.

QA Task:

- Review the story
- Test Cases
- Test Scenarios
- Test Data
- Test Review
- Test Environments
- Execute Test Cases
- Report Bugs etc.

==========

- **Scrum:** Scrum is a **framework** through which we build the software product by following Agile Principles.

- **Scrum workflow -**

  User Stories → Product Backlog → Sprint Planning → Sprint Backlog → Sprint (1 – 3 weeks and daily scrum meetings) → Product → Sprint Review and retrospective Meeting

- **Scrum Team:**

  Scrum Team involves,

  1. Product Owner
  2. Scrum Master
  3. Development Team
  4. QA Team

  1. Product Owner:
  - Product Owner is the First Point of Contact.
  - He will get input from the customers.
  - Defines the feature of the product.
  - Prioritize the features according to the market value.
  - Adjust features or priority every iteration, as needed.
  - Product Owner can Accept or Reject the work result.
  - He will define features of the product in the form of User Stories or Epics.

  2. Scrum Master:
  - Scrum Master has the main role of facilitating and driving the Agile Process.
  - He acts like a manager/team lead for Scrum Team.
  - He leads over all the Scrum ceremonies.

  3. Developers and QA:
  - Develop and test the software.

- **Advantages of Scrum (i.e., Agile Scrum):**
  - The quality can be ensured because each and every Sprint will be tested multiple times.
  - The requirement can be accepted at any level of project maintenance.
  - All the participating in Scrum meetings so that transparency can be maintained.
  - Each and every Sprint we are delivering to the client so we can maintain the customer's satisfaction and we can avoid the delivery risk of the project.
  - Helps to save time and cost of the project.

==========

- **Definition of Ready (DOR):**

If the below points are ready or clear **regarding User Stories, is DOR.**

- **User Story is clear.**
- **User Story is testable.**
- **User Story is feasible.**
- **…….**
- **…….**
- **…….**
- **…….**

- **Definition of Done (DOD):**

It is achieved when,

- The story is developed completely
- Testing (QA) complete
- Regression around the story is complete
- The story meets and satisfies the acceptance criteria
- The feature is eligible to be deployed in production.

==========

Scrum Terminologies:

- **Product Backlog:** Contains a list of all requirements (like user stories and epics). Prepared by Product Owner.

- **Epic:** Collection of related user stories. Epic is nothing but a large (high level) requirement.
- **User Story:** A feature/module in a software. Define the customer needs. It is nothing but the phrasing of the requirement in the form of a story.
- **Task:** To achieve the business requirements development team create tasks.
- **Sprint/Iteration:** **Period/time to complete** (means development and testing) the user stories, decided/selected by the Product Owner and Team. It is usually for 2-4 weeks of time.
- **Sprint Backlog:** List of committed stories by the Developers and QAs for a specific Sprint.
- **Sprint Planning Meeting:** This is the meeting with the team, **to define** what can be delivered in the Sprint and its duration.
- **Sprint Review Meeting:** Here we walkthrough and demonstrate the feature or story implemented by the team to the stakeholder.
- **Sprint Retrospective Meeting:** Conducts after completion of Sprint only. The entire team including the Product Owner and Scrum Master should participate.

  They discuss majorly on 3 things,
  - ✓ What went well?
  - ✓ What went wrong?
  - ✓ Improvements are needed in the upcoming sprint.
- **Backlog Grooming Meeting:**

  - In this meeting, the scrum team along with the scrum master and product owner.
  - The product owner presents the business requirements and as per the priority team discussed over it and identifies the complexity, dependencies, and efforts.
  - The team may also do the story pointing at this stage.

- **Scrum Meeting/Scrum Call/Standup Meeting:** This happens every day.

  Points to be discussed in Daily stand-up meetings:
  - ✓ What are we going to do today?
  - ✓ What have I done yesterday?
  - ✓ Any blockers/issues, due to which we are not able to proceed.

- **Story Point:** Rough estimation given by Developers and QA in the form of the Fibonacci series.
- **Time Boxing in Scrum:** Timeboxing is nothing but the Sprint which is the **specific amount of time to complete the specified amount of work.**
- **Scum of Scrums:**
  - Suppose 7 teams are working on a project and each team has 7 members. Each team leads its particular scrum meeting. Now to coordinate among the teams a separate meeting has to be organized, thatmeeting is called Scrum of Scrums.
  - An ambassador (team leads) (a designated person who represents the team) represents the team in the scrum of scrums.
  - Few points discussed in the meeting are:
    - ✓ The progress of each team, after the last meeting.

- ✓ The task is to be done before the next meeting.
- ✓ Issues that the team had faced while completing the last task.

- **Spike in Agile Scrum:** It is a story that cannot be estimated**.**
- **Sprint Zero:**
  - Sprint zero usually takes place **before the formal start of the project**.
  - This Sprint should be kept lightweight and relatively high level. It is all about the origination of project exploration and gaining an understanding of where you want to head while keeping velocity low.

- **Scrum Board/Task Board:** It is showing the status of the story.

  Scrum Board contains points like,

  - **User Story:** It has the actual business requirement.
  - **To Do:** Tasks that can be worked on.
  - **In Progress:** Tasks in progress.
  - **To Verify/Testing:** Tasks pending for verification or testing.
  - **Done:** Completed tasks.

- **Release Candidate:** The release candidate is a code/version/build released to make sure that during the last development period, no critical problem is left behind. It is used for testing and is equivalent to the final build.

- **Velocity:**
  - Velocity (is a metric) **used to measure** the **units of work done (completed)** in the given **time frame.**
  - We can say it is the **sum of story points** that the Scrum team **completed over a sprint.**

- **Burndown Chart:**
  - Shows that how much work is pending/remaining in the Sprint. Maintain by the Scrum Master daily. The progress is tracked by a Burndown chart.
  - Burndown chart is nothing but, it shows that estimated Vs. actual efforts of the Scrum Task.
  - To check whether the stories are doing **progress** towards the completion of the committed story points or not.
- **Burnup Chart:** Amount of work completed within a project.

==========

- **Difference between Scrum and Waterfall:**
    - Feedback from the customers is received at an early stage in Scrum than Waterfall.
    - New changes can easily accommodate in Scrum than the Waterfall.
    - Rollback or accommodate new changes is easy in Scrum.
    - Testing is considered a phase in the Waterfall, unlike Scrum.

- **Difference between Scum and Iterative Model:**
    - Scrum is a type of Iterative model, but it is an Incremental + Iterative model. As we break the product into small Incremental builds and then small pieces of builds are completed in multiple iterations.

- **Any other Agile Methodology apart from Scrum:**
    - Kanban, XP (Extreme Programming), Lean is another Agile Methodologies than Scrum.

- **Other frameworks used in the agile model:**
    - There are some development and methodologies where you can use agile like feature-driven development,lean software development, crystal methodologies, dynamic development.

- **Different ceremonies perform in the Scrum:**
    - Planning Meeting
    - Review Meeting
    - Retrospective Meeting
    - Backlog Grooming Meeting

- **Different Amigos/person involved in the Scrum:**

  There are three types of persons involved in Scrum which are, Product Owner, Scrum Master, and Scrum Team which includes Developer, Tester, and BA.

- **Ideal Size of Scrum team is 7 to 9 resources.**
- **Ideal duration of Sprint is 2-4 weeks.**
- **Requirements are defined in Scrum as User Stories.**

Different artifacts in Scrum:

Two artifacts are maintained in the Scrum are,

- **Product Backlog:** Contains a list of all user stories. Prepared by Product Owner.
- **Sprint Backlog:** Contains the User Stories committed by the Developer and QAs for a specific Sprint.

==========

Extra Questions:

- **Types of User Acceptance Testing**
    1. Alpha Testing
    2. Beta Testing

- **Different Tools for Functional Testing:**
    1. Selenium
    2. QTP (HP Product)
    3. SoapUI (This is API tool)

- **Different Tools for Non-Functional Testing:**
    1. JMeter
    2. Loadster
    3. LoadRunner
    4. LoadStorm
    5. NeoLoad
    6. Forecast
    7. Load Complete

So, in the scrum, which entity is responsible for the deliverables? Scrum Master or Product Owner?

- Neither the scrum master nor the product owner. It's the responsibility of the team that owns the deliverable.

- **How do you create the Burn-Down chart?**
  - It is a tracking mechanism by which for a particular sprint; day-to-day tasks are tracked to check whether the stories are progressing towards the completion of the committed story points or not. Here, we should remember that the efforts are measured in terms of user stories and not hours.

- **What do you do in a sprint review and retrospective?**
  - During Sprint review we walk-through and demonstrate the feature or story implemented by the scrum team to the stakeholders.
  - During Retrospective, we try to identify collaboratively what went well, what could be done better, and action items to have continuous improvement.

- **What qualities should a good Agile tester have?**
  - One should be able to understand the requirements quickly.
  - One should know Agile concepts and principles.
  - As requirements keep changing, one should understand the risk involved in it.
  - The agile tester should be able to prioritize the work based on the requirements.
  - Communication is a must for an Agile tester as it requires a lot of communication with developers and business associates.

- **How does agile testing (development) methodology differ from other testing (development) methodologies?**
  - In agile testing methodology, the entire testing process is broken into a small piece of codes and in each step, these codes are tested. There are several processes or plans involved in this methodology like communication with the team, short strategical changes to get the optimal result, etc.

- **Describe the places where 'Scrum' and 'Kanban' are used?**
  - 'Scrum' is used when you need to shift towards a more appropriate or more prominent process while if you want improvement in running the process without any changes in the whole scenario, you should use 'Kanban'.

- **Why aren't user stories simply estimated in man-hours?**
  - Estimation of user stories based on man-hours can be done but it is not preferred. If we do so we will concentrate on the cost and budget of the management while using man-hours.

- Instead of that, we can use story points, as it provides the complete idea about both the complexity of work and required efforts which can't be derived from Man-hours.

Do you think scrum can be implemented in all the software development processes?

- Scrum is used mainly for
  - ✓ Complex projects.
  - ✓ Projects which have early and strict deadlines.
  - ✓ When we are developing any software from scratch.

Tell me one big advantage of using scrum?

- The major advantage is – Early feedback from the client so if any changes are required those can be done at the initial stage and producing the Minimal Viable Product to the stakeholders.

Do you see any disadvantages of using scrum?

- I don't see any disadvantage of using scrum. The problems mainly arise when the scrum team does not either understand the values and principles of the scrum or are not flexible enough to change.

In case you receive a story on the last day of the sprint to test and you find there are defects, what will you do? Will you mark the story as done?

- No, I will not be able to mark the story as done as it has open defects and the complete testing of all the functionality of that story is pending. As we are on the last day of the sprint, we will mark those defects as Deferred for the next sprint and we can spill over that story to the next Sprint.

How do you measure the complexity or effort in a sprint? Is there a way to determine and represent it?

- Complexity and effort are measured through "Story Points". In Scrum, it's recommended to use the Fibonacci series to represent it. Considering the development effort+ testing effort + resolving dependencies and other factors that would require to complete a story.

When we Estimate with story points, we assign the point value to each item.

- To set the Story Point- Find the simplest story and assign the 1 value to that story and accordingly on basis of complexity we can assign the values to user stories.

- **During Review, suppose the product owner or stakeholder does not agree with the feature you implemented what would you do?**
  - First thing we will not mark the story as done.
  - We will first confirm the actual requirement from the stakeholder and update the user story and put it into the backlog. Based on the priority, we would be pulling the story in the next sprint.

- **Apart from planning, review, and retrospective, do you know any other ceremony in scrum?**
  - These three meetings are the ones which occur on regular basis, apart from these We have one more meeting which is the Product Backlog Grooming Meeting where the team, scrum master, and product owner meet to understand the business requirements, splits it into user stories, and estimating it.

- **Can you give an example of where scrum cannot be implemented? In that case, what do you suggest?**
  - Scrum can be implemented in all kinds of projects. It is not only applicable to software but is also implemented successfully in mechanical and engineering projects.

- **What is MVP in scrum?**
  - A Minimum Viable Product is a product that has just the bare minimum required feature which can be shown to the stakeholders and is eligible to be deployed to production.

- **How do you calculate a story point?**
  - A story point is calculated by considering the development effort + testing effort + resolving dependencies and other factors that would require to complete a story.

- **You are in the middle of a sprint and suddenly the product owner comes with a new requirement, what will you do?**
  - In an ideal case, the requirement becomes a story and moves to the backlog. Then based on the priority, the team can take it up in the next sprint.
  - But if the priority of the requirement is really high, then the team will have to accept it in the sprint, but it has to be very well communicated to the stakeholder that incorporating a story in the middle of the sprint may result in spilling over few stories to the next sprint.

- **Which are the top agile matrices?**
  - **Sprint burndown matric:** Shows that how much work is pending/remaining in the Sprint. Maintain by the Scrum Master daily. The progress is tracked by a Burndown chart. It shows that estimated Vs. actual efforts of the Scrum Task.
  - **Velocity:** Velocity (is a metric) **used to measure** the **units of work done (completed)** in the given **time frame.**
  - **Work category allocation:** This factor provides us a clear idea about where we are investing our time or where to set priority.
  - **Defect removal awareness:** Quality products can be delivered by active members and their awareness
  - **Cumulative flow diagram:** With the help of this flow diagram, the uniform workflow can be checked, where X-axis shows time and Y-axis shows no. of efforts.
  - **A business value delivered:** Business value delivered is an entity that shows the team's working efficiency. This method is used to measure, which around 100 points are associated with each project. Business objectives are given value from 1,2,3,5 and so on according to complexity, urgency, and ROI.
  - **Defect resolution time**: It's a process where team member detects the bug and priority intention by the removal of the error. A series of processes are involved in fixing the bug:
    - ✓ Clearing the picture of a bug
    - ✓ Schedule fix
    - ✓ Fixation of Defect is done
    - ✓ Report of resolution is handed
  - Time coverage: Amount of time given to code in question in testing. It is measured by the ratio of no. of the line of code called by the test suite by the total no. of the relative lines of code (in percentage).

- **How do you define a user story?**
  - Generally, the User stories are written by BA. So, I am not sure about the Syntax, but it is a process of Phrasing the requirements in a form of a story, it is a way of describing a feature set/Requirement like – **"I need …, So that …".**

- **Which SDLC model is the best?**

SDLC models are selected according to the requirements of the development process. Each model provides unique features for software development. Due to that, it may vary software-to-software to decide which model is best. But nowadays the Agile Model is the most popular and widely adopted by software firms.

Different Types of test plans in software testing?

Test plans can be used as supporting documentation for an overall testing objective (a master test plan) and specific types of tests (a testing type-specific plan).

- **Master test plan:** A master test plan is a high-level document for a project or product's overall testing goals and objectives. It lists tasks, milestones, and outlines the size and scope of a testing project. It encapsulates all other test plans within the project.
- **Testing type-specific plan:** Test plans can also be used to outline details related to a specific type of test. For example, you may have a test plan for unit testing, acceptance testing, and integration testing. These test plans drill deeper into the specific type of test being conducted.

What are the important tasks in the Test Planning stage?

- Understanding and analyzing the requirements
- Risk Analysis
- Test Estimations (Scope of the project, Time, Budget, Available resources)
- Team formation
- Test Approach (Strategy) Implementation
- Defining Test Environment setup
- Traceability Matrix
- Test Plan Documentation

What are the reference documents for Test Planning?

- Reference Documents for Test Planning stage:
  - ✓ Requirements
  - ✓ Project Plan
  - ✓ Test Strategy

—

- Design docs
  - ✓ Process guidelines docs
  - ✓ Corporate standards docs, etc.

What is Test Point Analysis?

- A formula-based test estimation method based on function point analysis.

What is Software Test Process?

- The fundamental test process comprises test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting, and test closure activities.

What is Function Point Analysis?

- A Method aiming to measure the size of the functionality of an information system. The measurement is independent of the technology. This measurement may be used as a basis for the measurement of productivity, the estimation of the needed resources, and project control.

What are the Entry criteria in Test Plan?

- The set of generic and specific conditions for permitting a process to go forward with a defined task, e.g., test phase.
- The purpose of entry criteria is to prevent a task from starting which would entail more (wasted) effort compared to the effort needed to remove the failed entry criteria.

What is Exit criteria in Test Plan?

- The set of generic and specific conditions, agreed upon with the stakeholders, for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task that have not been finished. Exit criteria are used to report against and to plan when to stop testing.

What is a Test deliverable?

- Any test (work) product must be delivered to someone other than the test (work) product's author.

What are the Test deliverables in Software Test Process?

Test deliverables are nothing, but the documents prepared after testing. Test deliverables will be delivered to the client not only for the completed activities but also for the activities, which we are implementing for better productivity.

Some Test Deliverables in the project are,

- Test deliverables

- Test plan document,
- Test case document,
- Test Result Documents (will be prepared at the phase of each type of testing,
- Test Report or Project Closure Report (Prepare once we rolled out the project to the client),
- Coverage matrix, defect matrix, and Traceability Matrix,
- Test design specifications,
- Release notes,
- Tools and their outputs,
- Error logs and execution logs,
- Problem reports and corrective action