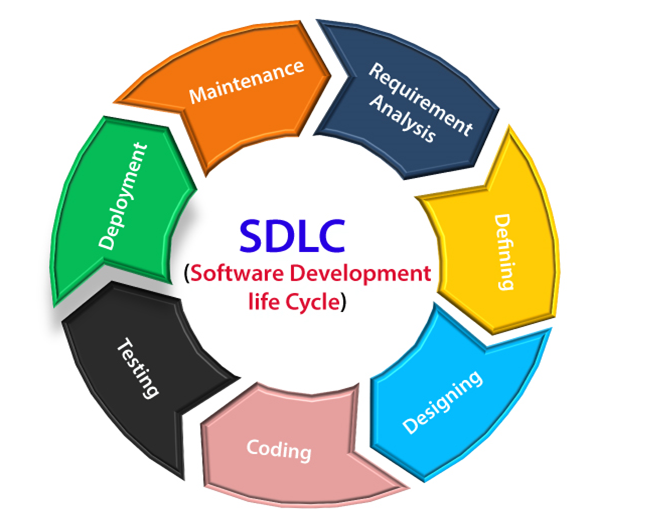
**Software Development Life Cycle – SDLC**

****

It is a step-by-step process to develop a new software.

**Stages in SDLC**

1) Requirement Collection

2) Feasibility Study

3) Design

4) Coding

5) Testing

6) Installation

7) Maintenance

1) Requirement **collection: B**usiness analyst from the company collects the requirement in business Language and converts that language into software language and explains it to the entire team (testing engineer, architect, developers, PM etc….). this process is called Requirement collection.

Requirement collection is done by BA (business analyst) or PA (product analyst). BA acts as a bridge Between company and customer.

**Who can become BA**

a) Domain expert

b) Senior developer / senior test engineer.

**2) Feasibility Study stage:**

It is done by a team which consists of PM, architect, BA, finance team and HR team. Project manager will interact or communicate with BA, HR team, or architect and gather all The Info and project manager will decide whether to take up the project or not.

3) Design: Once feasibility study is completed, we go to the design stage.Design is done by an architect / technical architect /senior architect.

There are two types of design:

1) High level design is also called external design.

2) Low level design is also called internal design.

4) **Coding:** Once design is completed, then we go for the coding phase or stage.It is done by looking into customer requirements and LLD. It is done by Senior developer, Junior developer and freshers by looking into the LLD and the requirements.

5) **Testing**: Testers test the software which is given by developers and test software to find bugs according to customer requirements.

6) **Installation**: Once Testing is done, installation will be done by IT Engineers. IT Engineers from the company go to the customer's place, install the software and provide a user guide to customers.

7) **Maintenance period**: This is nothing but an agreement period between company and the customer where they sign in for a particular period of time, in between this time, if there is any issue, we are responsible to fix the issues.

**There are 7 models in SDLC:**

1) Waterfall model

2) Spiral model

3) Prototype model

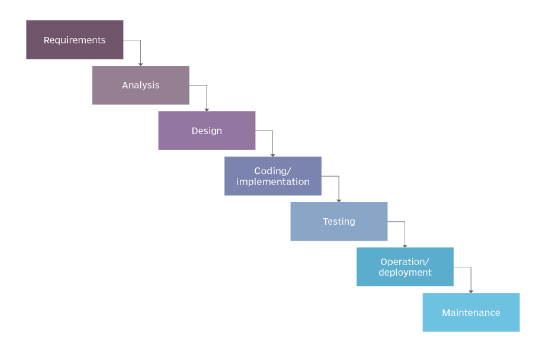
4) V & V model

5) Derived model (not covered)

6) Hybrid model (not covered)

7) Agile model

**WATERFALL MODEL**

****

It is also called a Traditional model or sequential model.

It follows the same stages of SDLC, where it starts with Requirement collection done by BA. BA collects complete requirements in Business language and explains to the team in software language. Later on we go with a feasibility Study where the Project Manager takes opinions from HR, Finance team, Architect, BA and other team members and decides whether we can take up this project or not.

Once Project is started, initially Architects need to design software in terms of HLD(High level design) and LLD(Low level design).

Developers by looking into LLD and requirements starts writing the code. Once done it will be moved to the Testing team, where Testers will find bugs according to customer requirements.

It is important to note that Requirements and Design is not tested in this model. Because of which if there is any defect in the requirements such as missing requirement, wrong requirement, and conflict requirements, it will flow to the next stages.

In Waterfall model, once Requirement collection and Feasibility study, we cannot go back and make any changes i,e,. backtracking is not possible because of which any defects found will flow till testing phase. To fix this issue, it consumes a lot of time and reworking. Hence this model is not preferred.

**Advantages**:

1) Product will be robust

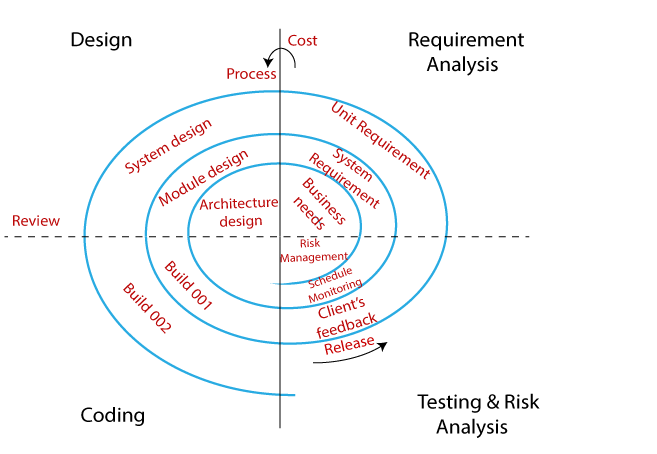
2) Simple model to follow

**Disadvantages**:

1) Requirement collection and design is not tested

2) Backtracking is not possible

**SPIRAL MODEL**

****

It is also called the Iterative model or Incremental model.

This model is called Incremental model because the customer keeps on adding the module after every cycle. It is called an Iterative model because the steps keep on repeating.

This model is divided into 4 stages:

1) Requirement and Feasibility Study

2) Design

3) Coding

4) Testing

**Module** **A** - Let’s say customer gives module A, in stage 1, we do Requirement collection and feasibility study, later on it is moved to Design stage, here Architect will do HLD and LLD and will be moved to coding stage, where Developer will look into the Requirement and LLD and starts writing the code. Later once the module is moved Testing, testers will test if the module is stable or not. If the module is stable, it will be released to the customer.

**Module B** - Let’s say customer gives module A, in stage 1, we do Requirement collection and feasibility study, later on it is moved to Design stage, here Architect will do HLD and LLD and will be moved to coding stage, where Developer will look into the Requirement and LLD and starts writing the code. Later the module is moved to Testing, here

1) Testers need to test for module B

2) Testers need to test for module A

3) Later, data flow between module A and module B needs to be tested

Once the modules are stable, we can release it to the customer.

**Module C** -- Let’s say customer gives module C, in stage 1, we do Requirement collection and feasibility study, later on it is moved to Design stage, here Architect will do HLD and LLD and will be moved to coding stage, where Developer will look into the Requirement and LLD and starts writing the code. Later once the module is moved Testing, here

1) Testers need to test for module C

2) Testers need to test for module B

3) Testers need to test for module A

4) Later test the dataflow between module C and B

5) Test the dataflow between C and A

Finally, once the modules are stable, it will be released to the customer.

**Advantages**:

1) Initial investment is less

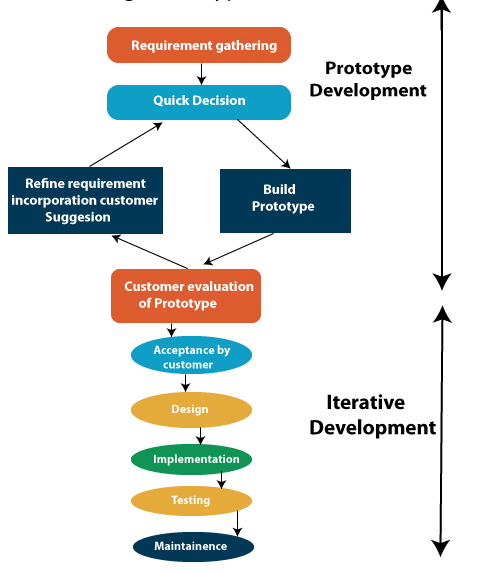
2) Integration between the modules is tested

3) Requirements changes are allowed after every cycle

**Disadvantages**:

1) Requirements changes are allowed only after the end of the cycle, not in between the cycle.

2) Every stage looks like a waterfall model.

**PROTOTYPE MODEL**

Prototype model is also called a dummy model.

**Definition**:Prototype model is a dummy model prepared by the web designers/ developers/content Developers where they convert ‘Text format’ into ‘image format’ by using tools like photoshop, paint, Picasa or web HTML.

**Explanation for prototype model:**

1) Let’s say, Customer will give requirement (60% to 70%)

2) BA will do requirement collection (60% to 70%)

3) Team will do a feasibility study.

4) Web designers, content developers, they will design & develop prototype model and give them to test engineers.

5) TE will test prototype while testing if they find any bug (some of the text format might not been converted into image format) they will communicate to developer and developer will fix the bug in prototype and send again to TE and TE will do prototype Testing, if there are no bugs, then prototype will be sent for customer review.

6) If the customer is happy (satisfied) with the prototype then the customer will approve to proceed further.

7) Architects will do actual design (HLD & LLD), developers will do actual coding /find Testing. If there are no bugs and software quality is good, then software will be released to the customer, so that customer can use software and run the business.

**Advantages**:

1) Customers will get to know how the software looks in the early stage itself.

2) Requirement changes are allowed.

3) We can set high expectations for the customer.

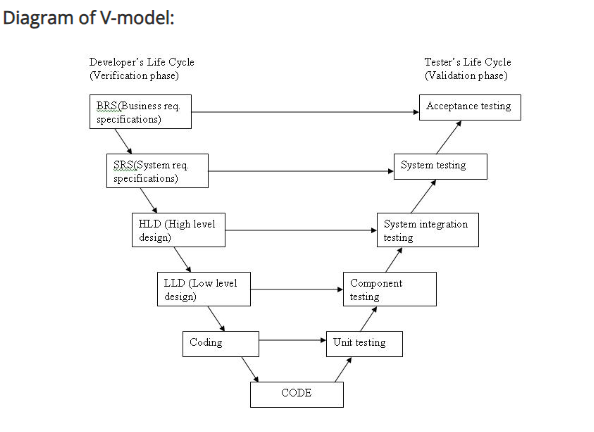
**Drawbacks/ disadvantages:**

1) Total time taken is more.

2) Total investment is high

3) There will be a delay in releasing software to the customer.

**V & V model(Verification and Validation model)**

****

In this model, Developers and Test Engineers work parallely, hence it is called V & V model.

Explanation for the model:

Customer will give the requirement in the form of ‘CRS’(it is also called BRS as in the above diagram) that will be in Business language. BA will go to the customer place and collect the CRS and Explain to ‘PM’ in the business language.

PM will take up the ‘CRS’ and send it to both developer and test engineers. Test engineer review CRS and start writing Acceptance test plan and test cases, parallelly developer will be

converting

‘CRS’ to SRS, while reviewing CRS, if test engineer find any bug(conflicting requirement, missing requirement, wrong requirement) they will communicate to developer and developer will communicate bug to customer and customer will update ‘CRS’ and send updated ‘CRS’ to both test engineer and developers. Tester will review updated CRS and also update corresponding Acceptance test plan and Acceptance test cases, parallelly developer will update defect in ‘SRS’ and continue converting ‘CRS’ to ‘SRS’, once both developers and test engineers complete the work, ‘SRS’ will be ready.

‘PM’ will take up ‘SRS’ and send it both to developers and test engineers. Test engineers will review ‘SRS’ against ‘CRS’ write System test plan and test cases. Parallelly developers will be converting ‘SRS’ to ‘HLD’ once both developers and Tester complete the work, ‘HLD’ will be ready. Like this way flow continues till the coding once after coding is completed, Developers will do WBT (testing each and every time of the code). While doing WBT, if developers find the defect in the code, developers will fix the defect and do WBT. Once WBT is completed, developer will give software to tester and Tester will start with validation activities, where in first tester will first do Functional testing by executing Functional test cases, while doing functional testing, if test engineer finds any bug in the software, then test engineer will inform to developer. Developer will fix the defect and do WBT and give new software to the tester and Tester will uninstall the old software, install new software & test the defect first and continue doing functional testing, after functional testing is completed. Testers will do Integration and System Testing respectively. Later on, customer/end-users will do Acceptance testing by executing test cases once Acceptance testing is completed, software quality is good, then software will be released to the customer, where the customer will use software and run the business.

**Advantages**:

1) Total time taken is less since testing starts from the early stage itself.

2) All the stages are tested (CRS, SRS, HLD, LLD, Code).

3) Since testing is done in every stage, the downward flow of defects is less because of this less Reworking and less time consuming.

4) Requirement changes can be done in any stage.

**Disadvantages**:

**1)** Initial investment is high.

**2)** Documentation is more (in every stage, we review documents and write test plan and test cases).

**Static and Dynamic Testing**

**Static Testing** – Testing the project related documents in the form of Review, walkthrough and Inspection is called Static Testing.

**Dynamic Testing** – Testing the actual software by Unit Testing, Integration Testing, System Testing, User Acceptance Testing.

**Review** – To check if the document is correct and complete.

Review can be done by single person

Reviews can be

a) Requirement Review

b) Design Review

c) Code Review

d) Test Plan Review

e) Test case Review

**WalkThrough** – This needs to be done by multiple people.

1

a) It is a Informal review

b) Author reads the document and discuss with the peers

c) It is not planned and can be done anytime

d) No time limits

**Inspection** :

1) It is a formal review type

2) 3-8 people will be part of this meeting. Readers (incharge of the document), writers(on who writes the questions), moderator(organizer) will be involved

3) Inspection is a proper schedule and will be informed to developer and Tester according to the schedule

**Levels of Testing**

**1)** **Unit Testing**

**2)** **Integration Testing**

**3)** **System Testing**

**4)** **User Acceptance Testing**

**Unit Testing** means testing small piece of a code/component

**Integration Testing** means testing the dataflow between the modules

**System testing** means testing overall functionality of the application

**User Acceptance testing** means customer testing to check if the software is ready for production or not

**TYPES OF TESTING**

**Functional Testing**

Testing the functionality of an application according to customer requirements. Here, we check if all modules/functionalities are working fine or not.

Functionality is nothing but checking behavior of the application.

1) Here we check the **functionality of an application**, for example by entering characters into the text field, we check whether the text field accepts required characters according to the customer requirements or not.

2) We check by clicking on modules, whether their **functionalities are working fine or not,** are we navigating to the right page/ screen or not.

3) **Links** – It is applicable for web applications only. We need to check where exactly links are placed, check if links are navigating to the proper page or not. Links can be internal links, external links and broken links.

4) **Cookies and sessions** – It is done for web applications only. Browsers save the information we entered in the form of cookies. Basically, cookies are temporary files created by the browser while the user is browsing. We need to test whether the browser is saving cookies or not.

**Importance of Functional Testing:** Functional testing focuses on verifying that a software application's features and functionalities work as intended and meet the specified requirements. It is an integral part of the software development life cycle, contributing to the overall success of the software product.

**Reasons why Functional Testing is so important.**

* Functional Testing verifies the functionality of core features.
* Functional Testing ensures that the software is developed according to customer requirements.
* Functional Testing helps to identify defects related to software functionality in the early stage itself.
* By conducting this Functional Testing we can ensure that software is user friendly and contributes to the overall user experience.
* This includes user interaction scenarios, by conducting Functional Testing, we can ensure that the users can interact with the software as intended.

**This can further be categorized into 2 Types: Positive Functional Testing and negative Functional Testing.**

* **Positive Testing** - Testing the functionality of the application by entering only valid data or data which is in accordance with customer requirements is called Positive Testing.
* **Negative Testing** - Testing the functionality of the software by entering only invalid data or data which is not in accordance with the customer requirements is called Negative Testing.

**Component Testing**

Component testing, also known as module or unit testing, is the process of testing individual components of a software application in isolation to ensure that each part functions correctly. This type of testing is typically performed by developers but testers can also be involved to ensure thorough validation.

### **Advantages of Component Testing for Testers**

* **Early Defect Detection:** Identifies defects early, reducing the overall cost and effort of fixing issues.
* **Improved Code Quality:** Enhances the quality and reliability of the code.
* **Simplified Maintenance:** Makes maintaining and updating code easier and more reliable.

Let's write some scenarios:

Assume that we need to test the password text field. We have a requirement that it should accept only alphabets and numbers, special characters are not allowed. Write Functional scenarios.

**Positive scenarios:**

1. To verify that the password text field accepts a combination of alphabets and numbers.
2. To verify that the password text field accepts only alphabets.
3. To verify that password text field accepts only numbers

**Negative scenarios:**

1. To verify that the password textfield does not accept a combination of alphabets and special characters.
2. To verify that the password textfield does not accept a combination of numbers and special characters.
3. To verify that when the password text field is blank, check whether the error message is displayed or not.

(Above example needs to be written with specific alphabets and numbers as test data in the test case template and needs to be tested accordingly.)

**Graphic user interface Testing (GUI Testing)**

What is GUI Testing?

Graphical User Interface (GUI) testing is crucial in software testing because the GUI is the primary point of interaction between users and the software application. The GUI represents the visual and interactive aspects of the software, including buttons, menus, forms, and other elements that users interact with.

Why GUI Testing?

Testing the GUI is important for several reasons:

* GUI Testing ensures better user experience as it makes application easy to understand and visually appealing
* This testing ensures user interface functionality is working properly
* It ensures navigation within the application and workflow is according to the user expectations
* It ensures visual elements such color, font, layouts remains consistent throughout the application
* It involves validating error messages that they are clear and informative to the users.

Note: We can test size, position, width, height of the elements, error messages are displayed or not, alignment of the images, spell checks, testing if there is any disabled fields, test size of the images, color of the hyperlink, check for checkbox, radio button, dropdown, links etc.

**Scenarios on GUI Testing:**

1. To verify that in the signup page of gmail, “Google logo” is displayed.
2. To verify that in the signup page of gmail, “forgot password” option is available.
3. To verify that in the signup page of gmail, in the username textfield, when we enter some data, check that all the letters entered are in the same size.

**Integration Testing**

Testing the data flow between two or more modules is called as Integration Testing.

**Importance of Integration Testing**: Integration testing is a critical phase in the software testing process that focuses on verifying the interactions and interfaces between different components or systems within a software application.

**Integration testing is important because of following reasons:**

* It ensures proper data flow - Integration testing validates the proper flow of data between different modules and
* ensures that data is transferred accurately. Data consistency and accuracy are crucial
* for the correct functioning of the entire system. Integration testing helps uncover data-related issues that may arise when different components exchange information.
* Validating business workflows - Users interact with software applications through specific workflows. Integration testing ensures that these workflows, which often involve multiple components, are seamless and error-free.
* It helps to detect Interface issues - Integration testing helps identify issues related to the interaction between different modules, components, or services.
* It ensures System Stability - By verifying the integration of components, integration testing helps ensure that the system operates cohesively and reliably, meeting performance and stability requirements.

**There are 2 types of Integration Testing:**

a) Incremental Integration Testing

b) Non-incremental Integration Testing

a) **Incremental Integration Testing –** Incrementally adding the modules and testing the data flow between the modules is called Incremental Integration Testing.

It is of 2 types: **Top Down approach and Bottom-up approach**

1) **Top Down approach** – Incrementally adding the modules and testing the data flow between the modules and ensuring that the modules we are adding are the child of the previous module is called the Top Down approach. Here data will flow from top to bottom.

2) **Bottom-up approach** – Incrementally adding the modules and testing the data flow between the modules and ensuring that the modules we are adding are the parent of the previous module is called the Bottom-up approach. Here data will flow from bottom to top.

b) **Non-Incremental Integration Testing** – Here we randomly test the data flow between all other modules. We go for this testing, when we don’t know which parent and child module or when requirements are complex to understand, we can do Non-Incremental Integration Testing.

**Positive Integration Testing scenario on Gmail:**

* Login as a user, click on compose. Enter some information, now click on cancel. Now click on the draft option, check that the information written so far should be displayed.

**Negative Integration Testing scenario on Gmail:**

Login as user, click on compose. Enter some information and click on cancel. Click on Trash and check if the entered details are displayed.

**System Testing**

**It** is an end to end testing conducted by the Test engineers in a testing environment which is similar to a production environment.

**What is End to end testing?**

Navigating through all the features and checking whether the end feature is working as expected or not is called end to end testing.

End-to-end testing is a comprehensive software testing approach that evaluates the entire software application from start to finish, including all integrated components and external dependencies. The primary goal of end-to-end testing is to ensure that the entire system functions as expected and meets the specified business requirements. This type of testing simulates real-world scenarios to validate the software's behavior in a production-like environment.

In this testing, we test all the end-to-end features and check whether it is according to the customer requirements. It is very essential to do end to end testing because customers in real time do verify end to end features of an application and if it is working fine or not.

**System testing is a critical phase in the software testing process that evaluates the entire software system as a whole. It is conducted after integration testing and before acceptance testing, focusing on verifying that the integrated system functions according to the specified requirements.**

**Importance of System testing:**

* System testing verifies the entire system including all integrated components and functions as intended.
* System testing identifies all System level defects, that may surface only when complete system functionalities are assembled.
* It verifies all individual components when integrated collectively fulfill the intended business and functional requirements.
* System testing ensures end to end business scenarios are met for better user experience.

**Positive System testing on Flipkart:**

* Log in to flipkart, click on mobiles feature, click on samsung, select one of the samsung mobile. Now click on buy now, select the payment option as phonepe, complete the payment. Click on the orders and check the product details are displayed.

**Negative System testing on Flipkart:**

* Login to flipkart, search for fastrack watches, select one of the watches. Click on buy now, click on continue, select the payment option as debit card. Generate one time password. Enter the wrong OTP and check if the payment is done.

**USER ACCEPTANCE TESTING**

It is an end to end testing done by the customers where they use software for real time business for some particular period of time and check whether software can handle all the real time business scenarios and situations. This is called User Acceptance Testing.

This Testing can also be called as Final Acceptance Testing or Red Box Testing because it is done in the last stage of testing. This Testing will be done by IT Engineers, agents or any technical team present in the customer’s place by applying real time business scenarios. Test Engineers based on customer’s request can conduct Acceptance Testing by looking into business scenarios.

**Why does the customer do Acceptance Testing?**

* To get confidence on the software.
* To ensure that the software meets the business requirements.
* To make sure that software company is not developing wrong features.
* To make sure software is bug free and is capable of running the business.

**Acceptance Testing is of 2 types: Alpha Testing and Beta Testing.**

**Alpha Testing** also called Internal user Acceptance is the testing done by the Internal team of Testing organization to check if software meets all the business requirements.

**Beta Testing** also called External user Acceptance testing is the testing done by the External teams or Client teams at customer organization to check if software meets all the business requirements.

**Change Request(CR)** - When a customer is not satisfied with the requirements they have given. They might reach out to the company and ask that they want to change the requirement. This request is called as Change Request.

**Request for Enhancement(RFE)** - Once software is released to the customer and if the customer feels this particular feature needs to be enhanced and developed in different ways. Then customer will again reach out to the company to enhance the feature/change the feature. This requirement is called as Request for Enhancement.

**Smoke & Sanity Testing**

**Smoke Testing** also called as Positive Testing, Build verification Testing,

Now, let’s assume, we have in 1 Sprint which has duration of 10 days and Release duration is 30 days and we need to test 3 modules A, B and C. Now, Testing team has planned in such a way that they have divided 3 modules into duration of 4,3, 3 days to complete the task. Now, let's say we completed testing for module A, now on 5th day we just started testing for module B, now there is a blocker defect and because of this defect we are unable to proceed further i,e,. user is blocked. Now, TE will inform the developer to fix it ASAP else the release will get postponed. Developer will say he needs at least 1 day to fix this issue. Once developers fix this issue, we need to start testing again from functional testing. If we go with this flow, it will take 35 days to release the build to the customer instead of 30 days. Customer might not accept this issue as it will affect his business. Hence to avoid these kinds of issues we should do Smoke testing in the beginning stage itself.

**So, remember Smoke Testing is the first testing done by Test Engineers. Followed by Functional Testing, Integration Testing, System Testing, Regression Testing etc.**

**What is Smoke Testing?**

It is High level testing conducted before doing rigorous testing or other types of testing to ensure that there is no blocker or critical defects that affects the customer business workflow.

**Q)** **Why Smoke Testing is necessary?**

* Because it helps in early detection of Blocker and critical bugs
* It verifies that the latest build or release is stable enough for further testing. If the build passes the smoke test, it indicates that the basic components are intact and functioning as expected.
* It ensures basic functionalities of the software is working fine
* It saves time and resources as we detect the blocker and critical bugs in the early stage itself.

**Q) When to do Smoke Testing?**

* Whenever new build is released
* Before conducting rigorous testing i,e,. Functional, Integration, Regression, End to end testing etc.

**What is Sanity Testing?**

It is also a kind of Smoke testing but it is conducted usually when we find a bug, then once retesting is done, we do sanity testing, meaning we check the application in depth and verify all dataflows if it is working correctly or not. This is called Sanity Testing.

It is basically a focused and narrow verification process performed on a specific functionality or a small set of functionalities within a software application. The primary purpose of sanity testing is to quickly check whether a particular aspect of the software is working correctly after changes, updates, or bug fixes.

**Q)** **Why is Sanity Testing necessary?**

* It helps to quickly validate specific/key functionalities of a software after some changes are done
* It helps to save time and resource as we focus on critical functionalities
* It acts like a build verification where it ensures if the current build is stable enough for further testing or not.

**Adhoc Testing**

Testing the application randomly without looking into the requirements is called Adhoc Testing. It is also called Out of Box testing or Negative Testing.

**Why Adhoc Testing?**

Customers might use the software randomly and might find defects, to avoid that we do Adhoc Testing. To find more defects, we should do Adhoc Testing.

Ad-hoc testing is an informal and unstructured testing approach where testers spontaneously execute test cases without predefined test plans or documentation. This type of testing is often exploratory and aims to discover defects that might not be found through more formalized testing methods.

Since it is not a planned Testing, it allows Testers to:

* Think creatively, include real world scenarios according to end user perspective.
* Find bugs in unexpected scenarios that were not part of regular testing approaches.
* Adapt to agile methodology, since it is dynamic and acts in accordance to changing nature of software.

**Task:**

Conduct Adhoc Testing for Bank application, assuming Bank manager needs to block user account number.

**Requirement** : Account number text field can accept valid 10 digit account number.

1. To check that when the user enters account number in reverse order, check if it is not accepting.

2. To check that when a user enters Branch pincode, check if it is not accepting.

3. To check that when a user enters IFSC code, check if it is not accepting.

4. To check that when a user enters a mobile number, check if it is not accepting.

5. To check that when a user enters a 9 digit account, checking if it is not accepting.

6. To check that when a user enters a valid username, check if it is not accepting.

7. To check that when user enters valid email ID, check if it is not accepting

8. To check that when the user does not enter any details/Blank, check if it is not accepting.

9. To check that when a user enters another bank IFSC code, Check if it is not accepting.

10. To check that when a user enters another bank account number, check if it is working or not.

11. To check that when a user enters the account number of the same bank but it is already blocked, check if it is working or not.

12. To check that when the user enters the Pan card number, check if it is not working.

**Regression Testing**

Testing the unchanged/old feature of an application to make sure that changes like adding a feature, modifying a feature, deleting a feature or fixing a defect is not introducing any defects in the changes or old feature is called Regression Testing.

**Regression Testing** is also called as **Release Candidate Testing**

Note:

1. First always test for newly added feature or any other changes made

2. Then test old module(nothing but regression testing)

**Types of Regression Testing**

1. Unit Regression Testing

2. Regional Regression Testing

3. Full Regression Testing

**Unit Regression Testing** – Testing only the changes or modifications done by the developer is called Unit Regression Testing.

Example : First name, Last name, In sign up country dropdown, FAQ section

**Regional Regression Testing** – Testing the changes and impacted areas of an application is called Regional Regression Testing.

Example : 1) Gmail - Assume an attachment file is being added in the compose module of gmail application, this will impact other modules such as Inbox, sent items etc.

2) Facebook – Assume features of FB – sign up, login, photo, like, share, logout. Now, if you add an additional feature upload gift in the photo module, this will have an impact on others such as share. So, we need to do Regional Regression Testing in this case.

**Q) How do we know which all areas got impacted?**

A) By doing an Impact Analysis meeting – Here we interact with Sr TE, Developers, BA, customers who have very good product knowledge, gather the information about the impacted areas and consolidate and document the impacted areas, this process is called an “Impact Analysis meeting”.

**Full Regression Testing** – Testing the changes and all the remaining features of an application is called Full Regression Testing.

Example: Let’s say there is an issue in one of the cell of MS Excel, since the cell is the root of the product here, we will have to make changes to the entire application page.

**Q) When we go for Full Regression Testing?**

* When changes are more, do not spend time doing impact analysis meeting, test the software by doing Full Regression Testing.
* When changes are made to the Root of the Product, then we need to test the entire product by doing Full Regression Testing.

**Q) Drawbacks of Regression Testing?**

* Time taken is more
* Manpower is more
* No consistency in Testing

To overcome the drawbacks of Regression Testing, companies go for Automation.

Regression Testing acts like a bridge between manual and Automation testing, when there are more number of Regression test cases, then we convert those manual test cases into automation script by using tools like Selenium/QTP.

**Q) What is Retesting?**

A) When TE finds a bug, he asks the Developer to fix the bug. The developer fixes the bug and gives it back to us, again as TE we need to retest if the bug is fixed or not. This is called Retesting. It is mandatory testing and high priority when compared to Regression Testing.

**Importance of Regression Testing:**

* Regression testing helps to detect bugs introduced by new code changes.
* Regression Testing ensures that existing features or functionalities work as expected.
* It ensures that bug fixes or modifications made does not impact other functionalities
* By conducting Regression Testing we can ensure that overall integrity of the software is achieved.
* In an Agile environment, where changes are frequent, regression testing helps team to adapt to continuous modifications.

**Exploratory Testing**

Exploring the application, understanding how each and every feature works and test the application as per your understanding is called Exploratory Testing.

**Q) When to do Exploratory Testing?**

* When there is no requirement
* There is a requirement, but it is complex to understand
* There is requirement, but TE does not have time to understand the requirement

**Q) Why Exploratory Testing when there is no requirement?**

A) If we do exploratory Testing, we can find most of the critical defects and blocker defects and ensure that customer business workflow is smooth though some minor defects are missed out. Hence it is advised to do Exploratory testing when there is no requirement.

**Q) Drawbacks of doing Exploratory Testing?**

* Sometimes we might misunderstand features as defects, sometimes defects as features.
* If a feature is missing, we will not get to know it is really missing.
* It is a more time consuming process because we need to spend more time in understanding the application.

**Note: We can overcome the drawbacks of Exploratory Testing by:**

* Having good interaction with Senior TE, Developer, BA, Customer etc who has very good product/domain knowledge.
* By comparing the application with a similar kind of application which is available in the market.

**Q) How to do Exploratory Testing?**

* By understanding the requirements, We can enter all possible inputs to each and every component.
* By exploring the application, checking the data flow between the modules.
* By understanding how each and every feature works and doing end to end scenario testing by doing Exploratory testing.

**Importance of Exploratory Testing:**

* Exploratory Testing mimics real world user scenarios
* Exploratory Testing is not scripted which allows the user to explore the application freely and test the software
* This testing help to identify bugs in the unplanned areas of the software
* Exploratory Testing validates user interactions and interfaces
* In agile environments where changes are frequent, exploratory testing supports continuous testing and validation, enabling teams to respond quickly to evolving requirements.
* It allows Testers to think creatively and test the software.
* Exploratory Testing brings continuous learning to the Testing team as Testers can understand the application behavior, its complexities and repeatedly adapt to their testing approach based on new information and discovery.

**Compatibility Testing**

Testing the functionality of an application in different hardware and software configurations or platforms or environments is called Compatibility Testing.

**Q) Why is compatibility Testing important?**

* If TE tests the software in one platform and release to market, customers can use the software in different platforms and software might not work as expected in other platforms, because of this customer market value goes down, to avoid this Compatibility Testing is a must.
* To ensure that all features are consistently working in all platforms, compatibility testing is necessary.

**Q) How to do Compatibility Testing?**

A) First we need to identify the base platform, I,e,, based on market research we need to analyze which browser, OS, MAC, Android versions most users are using. Which platform is widely used by our customers, that is our base platform. First we need to do all types of testing in our base platforms, the perform all testing in other platforms.

Note: Developers while writing the code can either write it in platform specific code(which is specific to particular platform) or can write it as Generic code(Test in all platforms).

Type of defects we see usually while doing compatibility Testing:

1) Scattered content defect

2) Alignment issues

3) Object overlapping

4) Look and feel of the application

**Non Functional Testing**

Once functionality of a feature is stable, then we do non-functional Testing.

**Non Functional Testing** mainly focuses on customer expectations not requirements.

In E Commerce applications, the features such as login, order, cart these are customer requirements. While doing Functional testing, we are mainly focussing on requirements and checking whether functionality is working correctly or not.

Now lets say Functionality of application is working fine, but what if the page is taking too long to respond, performance is low, it is not secured. Now these are the customer expectations we are speaking about. We are checking if the application is good, it is secured, it is working on different platforms, it is safe from unauthorized users.

Example: If more users use the application at the same time, we should ensure the application does not go slow.

Non-Functional Testing done by a separate team.

· **Performance Testing**

Performance Testing deals with checking the speed of an application. It is done for Web applications, but not applicable for Desktop applications.

**There are 3 types of Performance Testing:**

1) **Load Testing** – Increasing the load(multiple users) of the application slowly and checking the speed of the application. We use tools(jmeter, loadrunner) which will create virtual users to perform these tasks.

2) **Stress Testing** – Here we suddenly increase or decrease the load of the application and check the speed, stability of the application.

3) **Volume Testing** – Here we check how much data or volume an application can handle, we can check through volume testing.

1. Soak Testing(Endurance Testing) - We will add the user continuously for a particular period of time and check the stability of an application.

**Importance of Performance Testing:**

* **Performance** Testing contributes to user satisfaction. Users expect applications to be fast and reliable. Through this testing we can achieve customer satisfaction.
* Performance testing verifies the stability of the application under different levels of load. Applications should remain stable and reliable even when subjected to heavy user traffic or data processing loads.
* Performance testing validates the speed and efficiency of critical functionalities. Applications with slow or inefficient performance can deter users and impact the competitiveness of the software.
* Performance testing helps identify potential issues that could lead to system downtime. Downtime can have severe consequences, including loss of revenue, damage to reputation, and inconvenience to users.
* Performance testing reduces the risk of performance-related failures in production.

**Web Security Testing**

Through web security testing, we check how secure our application is.

Here we mainly focus on authorization and authentication.

* **Authentication** - Here, we check if users are valid or not.
* **Authorization** – Here, we check if user is authorized to access particular department or not.(Basically we check permissions given to valid users)

**Importance of Web Security Testing:**

* Web security testing helps safeguard sensitive user data, such as personal information, login credentials, and financial details.
* Web security testing reduces the risk of financial losses associated with data breaches, fraud, and other security incidents.
* Unauthorized access can lead to various malicious activities, including data theft, manipulation, or disruption of services.By conducting Web Testing we can prevent these issues.
* Web security testing helps protect against common web application attacks, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
* Web security testing ensures the secure transmission of data between users and the web application through encryption.
* Web security testing assesses the security of third-party integrations to ensure they do not introduce vulnerabilities

**Recovery Testing** – To check how soon our application recovers from the crashes and check if data is restored or not. For example: While placing an order in an E-com application, let's say you placed an order in a cart and suddenly the system shut down. When you login again, you should be able to see the same data in the cart whether it is available or not. Similarly, while composing a mail, if the system gets shut down. After user logins, he should be able to see this information in the drafts section.

In simple words, check if the system changes from abnormal state to normal state or not.

**Compatibility Testing** – Checking if applications work on different platforms or not.

Example – (Checking for version compatibility) Let’s say a developer develops an application with version number 1.5 and installed it in the system. For the same software when it is upgraded I,e,. version number is changed to 1.6, the software should be compatible to upgrade its version without any issues. (refer notes shared earlier)

**Installation Testing** – Here, we need to test if the installation is happening properly or not. Check if screens are clear to understand or not. Also we need to uninstall the software and check if all files belonging to that software are removed or not.

**Sanitation or Garbage Testing** – If we have any additional features in the software which is not required according to customer requirement, we can raise it as a bug and ask the developer to remove these additional features.

**Important Differences between Functional and Non-Functional Testing:**

**Functional Testing**

1) It is done before Non-Functional Testing

2) It validates/checks functionality of the software

3) Functionality describes what software does

4) Works on customer requirements

**Non-Functional Testing**

1) It is done after Functional Testing

2) It validates/checks non- functionality of the software

3) Functionality describes how software works

4) Works on customer expectations

**Challenges in Software Testing:**

1)  **Complexity of Software**: As software systems become more complex, testing becomes more challenging. Interactions between different components, integration issues, and the need for testing across various platforms and devices add to the complexity.

2) **Changing Technology**: Rapid advancements in technology and the adoption of new tools and frameworks can make it challenging for testing teams to keep up. Learning and implementing new testing tools can be time-consuming.

3) **Regression Testing**: As software evolves, changes to one part of the code can unintentionally introduce defects in other areas. Regression testing is essential to ensure that new features or bug fixes do not negatively impact existing functionalities. However, running comprehensive regression tests can be time-consuming.

4)  **Incomplete Requirements:** Testing without clear and complete requirements can lead to ambiguity in test cases. Incomplete or changing requirements make it difficult to create comprehensive test cases, leading to potential issues being overlooked.

5) **Time Constraints**: Often, software development projects are subject to tight deadlines. Testing is sometimes compressed, leaving limited time for thorough testing. This can result in inadequate test coverage and potentially missed defects.

6)  **Communication Issues**: Effective communication between development and testing teams is essential. Miscommunication or a lack of collaboration can result in misunderstandings about requirements, leading to ineffective testing.

**Principles of Software Testing**

1. We should not do Exhaustive Testing
2. We should do early Testing
3. Testing should be done to identify the bugs in the software
4. We should focus on “Pesticide paradox”
5. We should focus on “Defect clustering”
6. Testing is context dependant(depending on the type of application and customer requirements, testing should be done)
7. Absence of errors in the software does not mean that software is free from defects.

**Pesticide Paradox:** If the same test cases are run for more number of executions, then the test case will not have capability to catch new bugs, hence we need to re-update our test cases. This concept is called the Pesticide Paradox.

**Defect clustering**: Non-uniform distribution of defects across the features or one feature having more number of bugs, another feature having very less number of defects can be termed as Defect clustering.

**Difference between Defect, Bug, Error and Failure**

1) **Defect** – When developers are doing Unit testing to check whether functionality is perfect or not in their local set up, now they compare actual result with the expected result, there is any difference between the actual result and expected result, this leads to a defect. Basically this occurred during the development phase.

2) **Bug** – When Tester is testing the application, if they find variation between actual result and expected result, this will be considered as Bug. Also defects accepted by developers can be considered as bugs. Bug is also an informal name given to the defect.

3) **Error** – Mistakes done in the program while writing the code, because of which the developer is not able to write and compile the code is called an Error.

4) **Failure** – Once software is developed, it is tested and verified by our testers. Finally end users are facing issues in the production, it is termed as failure.

Defect - Any deviation from customer requirements is called as Defect. Or

If Expected Result and Actual Result does not match while testing the software, we cam consider it as a Defect.

**Severity and Priority**

1) **Severity** – Impact of Bug on customer business workflow is called Severity.

If the impact is high, it can be considered as high severity.

If the impact is low, it can be considered as low severity.

2) **Priority** – Importance given to fix the bug is called Priority.

**Types of Severity:**

1) Blocker

2) Critical

3) Major

4) Minor

1) **Blocker** – Lets say, when the Tester is testing the application it is not allowing the user to proceed further, it may say internal server error or might take the user to a blank page, the functionality itself is blocked. This is considered a Blocker bug.

2) **Critical** – If the main functionality of the application is not working or not according to customer requirement specification, it can be considered a Critical bug. In other words, if Tester is sure that this bug affects customer business flow, then it can be termed as Critical bug. Note that it is not blocking the user, but customer business work flow is affected largely because of this bug.

3) **Major** – If the Tester is not sure that this bug which is found is affecting the customer business workflow or not, it can be considered as a Major bug. In other words, if minor functionalities of an application is affected, it can be considered a Major bug.

4) **Minor** – If the Tester is sure that this bug is definitely not affecting the customer business workflow, it can be considered a minor bug. These are usually with respect to the look and feel of the application, alignment issues, spell issues etc.

**Types of Priority**

1) P0 – High Priority, the developer has to fix the bug immediately.

2) P1 – Medium Priority, developers can fix the bug in upcoming builds.

3) P2 – Low Priority, developers can fix the bug in the coming releases.

Blocker and Critical bugs can be considered as High severity.

Examples:

**High Severity High Priority** – While logging into the login page, a blank page is displayed. It is a blocker bug and Priority is P0 because this has a large impact on customer business workflow.

**High Severity Low Priority** – When a user clicks on the Terms and conditions page, a blank page is displayed. It is High Severity low priority because this does not affect customer business workflow.

**Low Severity High Priority** – When logging in to the login page, let’s say login spelling is wrong (logone), now this low severity, but it will impact the user as it is displayed in the first page itself.

**Low Severity Low Priority** – Lets say there is a spelling mistake in the inside contact information page with respect to email Id, phone etc or any color formation issue or alignment issues. This can be considered as a Low severity and low priority.

**Test Plan**

Test Plan is a document which derives all future testing activities. Test manager will review and approve the Test plan.

**Test plan consists of 15 attributes:**

1) **Objective**: It speaks about the aim of writing the test plan. We can mention which model is used and what process is used.

2) **Scope**: Here we mention features which need to be tested and features which need not be tested.

3) **Test methodology**: Based on type of application, we decide what type of testing needs to be done.

a) Web based application

b) Client Server application

c) Stand alone application

4) **Test approach**: It explains how we test the application. Several approaches are:

a) Writing Test cases

b) Writing Test scenarios

c) Writing Test cases and Test scenarios

d) Writing the flowchart

5) **Assumption**: While writing Test plan, team will assume few aspects:

a) Assumption from resource point of view

b) Assumption from technology point of view

c) Assumption from development team point of view

d) Assumption from Knowledge Transfer point of view

e) Assumption from supporting documents point of view

6) **Risk**: When assumption fails, we will be in Risk, to overcome it, we have a Backup plan.

7) **Backup plan/Mitigation Plan**: If assumption fails, it will lead to 100% risk, by having a Backup plan we can reduce risk to 20%. Example: Lets say 3 TE’s are working on a project, due to some reasons, 1 TE quits job suddenly. Now the team is at risk as we are supposed to finish the project in a given deadline. To overcome this risk, we will initially assign a secondary TE, so that they can act as a Backup and ensure software is tested in a given timeline before it is released to the customer.

8) **Roles & Responsibilities of TE**:

a) Understand customer requirement

b) Write Test scenarios & Test cases

c) Conduct Brainstorming meeting for better test case coverage & get test cases reviewed

d) Always do optimized testing

e) Conduct Smoke, Functional, Integration, System, adhoc testing etc

f) Perform Test case execution

g) Log the defect to the defect tracking tool

h) Give Knowledge Transfer to other team members

9) **Scheduling**: In this section, we mention start and end dates for each and every testing activity.

10) **Defect Tracking**: As soon as TE finds any defect, he will login to the Defect tracking tool and log the defect, generate a defect report, and a unique id will be created. TE will communicate this defect to the developer and the developer will fix the defect and give back to TE. TE will retest to check if the defect is really fixed or not. If a defect is fixed, TE will close the defect. This process is called Defect Tracking.

11) **Test environment**: It is an environment configured for testing, where TE will test the application by executing the test cases. Test environment consists of hardware, software, database, OS, CPU etc details.

12) **Entry & Exit criteria**: These are the set of conditions that should be met in order to start and end the project. It is also the key attribute of Test plan.

**Functional Testing:**

Entry criteria:

1) Coding should be completed

2) WBT should be done

3) Software should be installed

4) Smoke Testing should be done

5) Functional Test scenarios & Test cases should be ready

6) Resource should be present

Exit Criteria:

1) Based on number of Test cases executed, test case pass% should be 85%

2) Based on number of defects found:

a) There should be no blocker defects

b) Critical defects should not be more than 20

c) Major defects should not be more than 50

d) Minor defects should not be more than 80

**Integration Testing:**

Entry criteria:

1) It should match exit criteria of Functional Testing

2) Functional Test scenarios & Test cases should be ready

3) Resource should be present

Exit Criteria:

1) Based on number of Test cases executed, test case pass% should be 95%

2) Based on number of defects found:

a) There should be no blocker defects

b) There should be no critical defects

c) Major defects should not be more than 20

d) Minor defects should not be more than 40

**System Testing:**

Entry criteria:

1) It should match exit criteria of Integration Testing

2) Integration Test scenarios & Test cases should be ready

3) Resource should be present

Exit Criteria:

1) Based on number of Test cases executed, test case pass% should be 99%

2) Based on number of defects found:

a) There should be no blocker defects

b) There should be no critical defects

c) Major defects should not be more than 10

d) Minor defects should not be more than 20

13) **Test Automation**: Here we mention which automation tool we use on a project, which features to be tested and which need not be tested, which automation framework we use.

14) **Deliverables**: These are the outcomes of the Testing team. This section contains what we are going to give to customer by the end of the project. The document includes – Test plan document, Test scenario & Test case document, Traceability matrix document, Test execution report document, Defect report document, Release notes, Graphs & metrics

**Q) What do you mean by Release notes?**

A) These are the set of documents given to the customer along with software which is signed off by the Test manager.

Release notes consist of – List of Pending and open defects, List of defects fixed in current release found in previous release, platform on which software is tested, platform on which software is not tested, list of features added, deleted, modified, procedure to install the software, version of software.

15) **Templates**: Here, we mention all the empty templates which will be used in future by TE’s.

Templates include – Test case template, Test case review template, Traceability matrix template, Test case execution template, Defect report template, Test plan template.

**Test Strategy**

Test strategy is a high level plan consisting of principles that guide the overall software testing process.

It provides a structured approach to the entire QA team, guiding them towards achieving testing objectives in the most efficient way.

Q**) Why Test Strategy**?

1. To make sure that all purposes are covered entirely and understood by all stakeholders

2. To support quality assurance with respect to planning of resources, language, test, roles and responsibilities etc.

u **Test Strategy document** - It is a well described document in software testing which clearly defines the exact software testing approach and testing objectives of the software application.

u Components of Test Strategy

u Scope & Overview

u Testing methodology

u Testing environment

u Testing tools

u Release

u Risk analysis

u Review and approvals

**Test case Design Techniques**

**Types of Test case Design Techniques:**

1) Error Guessing

2) Equivalence Class Partitioning

3) Boundary Value Analysis

4) Decision Table Technique

5) State Transition Diagram

1) **Error Guessing** – Here, TE will guess the error and derive more scenarios.

Example: Assume there is an “Amount” text field and the requirement says that it accepts + integer only.

Now we will have to enter only invalid inputs such as -10, abc, 10@ etc and try to guess more errors in this case.

2) **Equivalence Class Partitioning** - Here, when the input is in range of values, let us say, there is an “Amount” text field and the requirement says that it can accept numbers between 100 to 500. In this case what we can do is instead of entering all numbers, we can divide this range into equal classes I,e,. -100 to 0, 0 to 100, 100 to 200, 200 to 300, 300 to 400, 400 to 500 & 500 to 600. After this try entering any 1 value from each of this class. Example if you enter 150 for the class 100 to 200, need not enter other values in range of 100 to 200, if it is accepting, then test case is pass, if it is not accepting, then test case is fail. Likewise we can only test only 2 scenarios/values(which includes 5 positive & 2 negative scenarios) and ensure that our test case coverage is achieved.

3) **Boundary Value Analysis** – Lets say, here you need values between the range A to B. We need tests for A,A+ & A- similarly B, B+ & B- . So here we will have 4 positive scenarios and 2 negative scenarios. Since the boundaries are covered, our test case coverage is good and no need to test for other values.

4) **Decision Table Technique** – In this Technique, we check for multiple conditions, combinations and Rule criterias.

Formula = 2^no of conditions= total no of rules or scenarios

Example1:

Requirement- Customer wants to order from Swiggy,

1) First time customers get 50% discount

2) If the coupon code is used, they get a 25% discount.

In this case there are 2 conditions, 2^2 = 4 scenarios we can derive.

Example 2 for Login page:

5) **State Transition Diagram** – This technique is used to check for different screens of a software. It is basically a pictorial representation of the scenarios.

Example: Let's say a person has to withdraw cash from an ATM machine, we can derive 4 scenarios for this.

1) When he enters the correct pin for the first time, he withdraws the cash.

2) When he enters the wrong pin for the first time and the correct pin for the second time, he can withdraw the cash.

3) When he enters the wrong pin for the first and second time and enters the correct pin for the third time, he can withdraw the cash.

4) When he enters the wrong pin for all attempts I,e,. first, second & third attempt, the card gets blocked and he cannot withdraw the cash.

These scenarios customers can write in a pictorial way and present how scenarios we can derive here.

**Traceability matrix**

It is also called as Requirement Traceability matrix(RTM) or Cross reference matrix (CRM)

It is a document through which we are ensuring that each and every requirement has minimum test cases.

**Advantages of RTM:**

1) Ensures complete requirements are documented.

2) It helps in analyzing the root cause of any defect.

3) Applications can be developed according to the requirements.

**Types of RTM:**

1) Forward Traceability Matrix

2) Backward Traceability Matrix

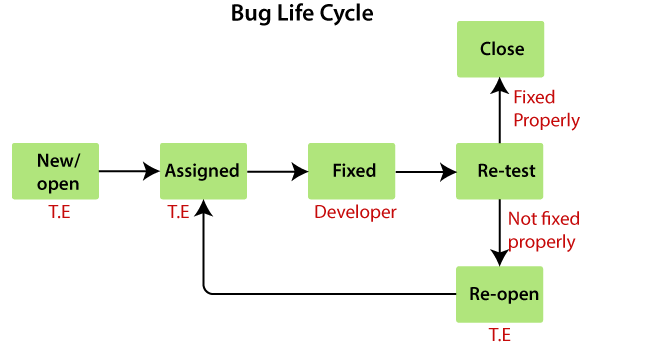
3) Bi-directional Traceability Matrix

1) **Forward Traceability Matrix** – It is used to map requirements to the test cases. This is done before Test case execution. Here we are ensuring that the product developments are going in the right direction.

2) **Backward Traceability Matrix** – It is used to map Test cases to the requirement. This is done after Test case execution. Here we are ensuring that we are not going against/developing products against customer requirements.

3) **Bi-directional Traceability Matrix** – It is a combination of both Forward Traceability Matrix and Backward Traceability Matrix.

**BUG LIFE CYCLE**

****

The bug life cycle, also known as the defect life cycle, describes the stages that a software bug goes through from the time it is identified until it is resolved. The bug life cycle typically consists of the following stages:

**New**:

This is the initial stage when the bug is first identified and reported by a tester or user.

The bug is in the "New" state until it is reviewed and confirmed by quality assurance (QA) team member.

**Open**:

Once the bug is confirmed, it is assigned to a developer for further investigation and fixing.

The bug is in the "Open" state during the development phase.

**Assigned**:

The bug is assigned to a specific developer or a development team responsible for fixing it.

This stage indicates that the developer is actively working on the bug.

**Fixed**:

After the developer has successfully addressed the bug, the status is changed to "Fixed."

The bug is considered resolved in the developer's environment.

**Retest**:

In this stage, the QA team tests the fixed bug to ensure that it has been resolved successfully.

If the bug is verified, it moves to the next stage.

**Closed**:

The bug is marked as "Closed" when it has been fixed, retested, and verified.

This signifies that the bug has been resolved and the issue is considered closed.

**Reopened**:

If the bug is found to persist or reoccurs after being closed, it is reopened.

The bug goes back to the "Open" or "Pending Retest" stage depending on the circumstances.

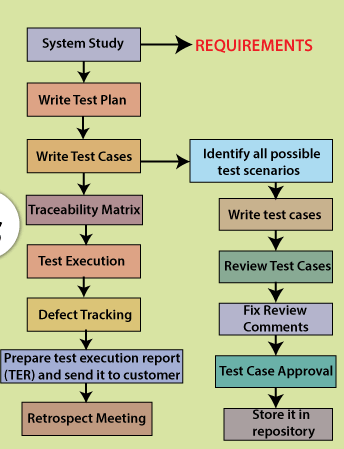
**Duplicate**:

If it's discovered that the reported issue is a duplicate of another bug, it may be marked as a duplicate.

The development team then focuses on fixing the original bug, and the duplicate is closed.

These stages may vary slightly depending on the specific bug tracking or issue management system used by a development team. The bug life cycle is crucial for effective communication between developers and testers and helps in tracking the status of bugs throughout the software development process.

**STLC- Software Testing Life Cycle**

****

STLC stands for Software Testing Life Cycle, and it represents the series of activities performed during the testing process of a software application or system.

**System Study**: Customer give requirements in the form of CRS. BA will convert CRS to SRS and BA will explain how each and every feature works for the entire team. BA will also explain customer business scenarios to teammates. Every Test Engineer will understand requirements, while understanding requirements, if they find any issues, they will talk to BA and clarify the doubts. This complete process of understanding the requirements is called System study.

**Test Plan**: It is a document which derives all the future testing activities. Test plan document consists of information regarding how TE are required to test in future, which features are to be tested by which Test Engineers, what types of testing to be done on project in future, roles and responsibilities of every Engineer working on project, which Test cases management tool to use, which defect tracking tool to use in future will be maintained in Test plan document.

**Test case**: Once the Test plan is completed, every TE will start writing Test cases from their assigned modules, where the TE will follow the procedure to write the Test cases. First do System study for assigned modules, later identify all the possible scenarios, document it and conduct brainstorming meetings to have good coverage, later once scenarios are ready, TE will write Test cases by applying Test case design Technique in Test case Template and get it reviewed by reviewers. If there is any problem in Test cases, it will be fixed by TE and later it will be approved by Test Lead. Finally Test case will be Test case repository.

Prepare Traceability matrix – It is a document which ensures that each and every requirement has got at least 1 Test case. There are 3 types of Traceability matrix. They are:

1) **Forward Traceability Matrix** – It is used to map requirements to the test cases. This is done before Test case execution. Here we are ensuring that the product developments are going in the right direction.

2) **Backward Traceability Matrix** – It is used to map Test cases to the requirement. This is done after Test case execution. Here we are ensuring that we are not going against/developing products against customer requirements.

3) **Bi-directional Traceability Matrix** – It is a combination of both Forward Traceability Matrix and Backward Traceability Matrix.

**Advantages of RTM:**

1) Ensures complete requirements are documented.

2) It helps in analyzing the root cause of any defect.

3) Applications can be developed according to the requirement.

**Test case execution**: In this stage, developers will give the build to Test Engineers and Test Engineers will start to test the build by executing the Test cases against the application where in TE will do Smoke testing, Functional Testing, Integration Testing, System Testing etc. Since Test Engineers are testing the application by looking into Test cases, it is called Test case execution. Here, we follow the procedure to execute the test cases.

**Defect Tracking**: While executing the test cases, if the Test engineer finds any defects, he will login to Defect tracking tool and prepare defect report and communicate to developer with unique defect id. Developer will fix the defect and TE will retest the defect and if the defect is really fixed. He will close the bug or TE will re-open the bug. This is called Defect Tracking.

**Test case execution report**: TL’s will prepare Test case execution report/test case summary report for every build/feature/test cycle/ end of every Release. This report consists of how Test cases are written, executed, not executed, pass or fail. This report will be sent to Management team, Development team, Testing team, customers

**Release Retrospect meeting or Project closure meeting**: Once Testing is completed, at the end of the Project, Test manager will do this meeting called as Release Retrospect meeting where in he will invite all the Test engineers who has worked on project and discuss about the mistakes and achievements done by Test engineers, while working on project.

**MIND MAPPING**

****

Mind mapping is a valuable technique in software testing for several reasons:

**Visualization of Information**: Mind maps provide a visual representation of complex information, making it easier to understand and remember. In software testing, where there are various test cases, requirements, and interconnections, a visual overview can be immensely helpful.

**Organization of Test Scenarios**: Mind maps help organize and structure test scenarios hierarchically. Testers can break down testing into different levels (e.g., unit testing, integration testing, system testing) and map related test cases under each category.

**Creative Problem Solving:** Mind maps encourage creative thinking and brainstorming. Testers can use mind maps to explore different test scenarios, edge cases, and potential issues that might not be immediately apparent in a linear document.

**Efficient Test Planning**: Test planning involves considering various factors such as test objectives, scope, resources, and schedules. A mind map provides a clear, concise overview of the test plan, making it easier for team members to understand and contribute to the planning process.

**Traceability**: Mind maps can be used to establish traceability between test cases and requirements. This visual linkage helps ensure that every requirement has associated test cases and provides a clear view of test coverage.

**Communication and Collaboration**: Mind maps are effective tools for communicating testing strategies and progress to team members, stakeholders, and even non-technical audiences. They promote collaboration and provide a shared understanding of the testing process.

**Test Case Design**: When designing test cases, mind maps help testers systematically consider various input combinations, scenarios, and potential outcomes. This aids in creating comprehensive test coverage and reduces the likelihood of overlooking critical test scenarios.

**Flexibility and Iteration**: Software testing often involves iterative processes as the software evolves. Mind maps are flexible and can be easily updated or modified as testing progresses or as new information becomes available.

**Test Execution Planning**: Mind maps can assist in planning the execution of test cases. Testers can use the map to determine the optimal sequence of test execution, helping to identify any dependencies or prerequisites.

**Documentation**: While mind maps are not a replacement for detailed documentation, they provide a concise and structured overview that can complement more extensive documentation. This is particularly useful for conveying key testing concepts and strategies.

**In summary, mind mapping is a versatile and powerful tool in software testing. It enhances understanding, aids in planning and execution, supports creative problem-solving, and facilitates communication and collaboration among team members. By leveraging mind maps, testing teams can streamline their processes and improve the effectiveness and efficiency of their testing efforts.**

**AGILE**

It is an Iterative and Incremental approach where requirements keep on changing, as a company we should be flexible to take up all the requirements, develop the changes, test the changes and give quality software to customers within a short span of time.

Through Agile, we can provide customer satisfaction through “quick delivery of working piece of the software”.

**Advantages**:

1) Customers can change the requirement at any stage of development.

2) Release will be short.

3) Developers, Test Engineers and BA will be having meeting regularly. It will be helpful to improve the process.

4) Teams will be self-organized.

5) This methodology is fast and flexible.

**Disadvantages**:

1) We are not sure when the customer will stop changing the requirements.

2) We cannot do effort estimation at the beginning of the project.

3) More pressure in the process.

**Few Types/Flavours of Agile:**

1) Scrum model - It is a framework which implements Agile process

2) Extreme Programming

3) Feature Driven Development

4) Crystal clear

5) Lean and Kanban

**Scrum model** – Scrum model is a standard procedure/framework which helps team to work together and develop new software.

**Agile Terminologies:**

1) **Epic** – Collection of user stories.

2) **User stories** – These are nothing but functionalities or features

3) **Sprint** – Actual time taken to develop and test one or more user stories.

4) **Story point** – It is a rough estimation given to develop and test individual user stories. It is calculated in terms of the Fibonacci series.

0, 1, 1, 2, 3, 5, 8, 13, 21

0,1, 1, 2, 3, 5, 8, 13

5) **Burndown chart** – This chart is used to calculate the total amount of work remaining at the beginning of every sprint.

6) **Burnup chart** – This chart is used to calculate the total amount of work completed at the end of every sprint.

7) **Capacity** – Total amount of available hours of every individual engineer for the given sprint is called Capacity.

8) **Velocity** – Total amount of work completed by every individual in the given sprint is called Velocity.

**Q) Who is a Scrum master?**

A) Scrum master is a person who is responsible for delivery of the software in the planned period of time. He will track each and every activity of team members working on the project. Scrum master mainly works on Project management tools like Jira where he creates Epic for requirements, create user stories for the epic and create tasks for the user stories and assign tasks to every individual engineer.

**Scrum ceremonies/meetings in Scrum model**

1) **Product Backlog** – This contains a list of all user stories or tasks to develop a software. Product Owner is responsible for backlog of product, based on Product owner feedback, the user stories will be divided to work efficiently in further stages.

2) **Sprint Planning** – It is a meeting conducted by Scrum master on the first day of the Sprint(some do conduct on the last day of the Sprint). In this ceremony, BA will explain how each and every user story should be worked and explains business workflow. In this meeting, user stories will be prioritized according to customer requirements and work will be assigned to every team member respectively.

3) **Sprint Review** – This is a demo/report given to Product owner/Stakeholders, how the user stories have been worked so far. The intention of conducting this review to get feedback on the product worked so far. This review is helpful in analyzing the progress of the product.

4) **Sprint Retrospect** – This meeting is usually conducted at the end of the Sprint, where Scrum team members discuss what went well, what did not go well and what are the action plans. The main purpose of conducting this meeting is to get feedback on the process we are working on and implement the ideas to make it better. This meet can also be called a Sprint closure meeting.

5) **Scrum meeting** – This is also called a Daily stand-up meeting, conducted everyday for a short duration, ranging from 15-20 mins.

6) **Bug Triage Meeting** – This meet is conducted to know the status of pending and open defects and we will prioritize bugs according to customer business workflow and fix it accordingly. Sr Testers are responsible for this meeting as they have better ideas on bug priorities. As Testers, we are supposed to give a Live report once a bug triage meeting is done before production.

7) **Project closure meeting** – In this meeting we discuss the workflow done so far, how efficient we were in delivering quality products and what are the areas of improvement. This is usually conducted at the end of the Project.

**Q)** **What is a Sprint Backlog?**

A) In Jira, during Sprint planning, we decide what are the stories we are prioritizing during the start of the Sprint and those user stories will be stored in the Sprint backlog.

**Q) How do you rate story points to your user stories?**

A) We need to first breakdown user story into task and subtasks and convert those into scenarios, Test cases, identify time taken to execute Test cases, roughly estimate time period and rate story point using Fibonacci series.(Fibonacci series – 0,1,1,2,3,5,8.13,21…..)

**Q) What is the Test Cycle duration?**

A) Time period taken to test 1 COMPLETE build is called Test cycle duration.

**Few Important notes:**

**Short Release/Interim Release** – In production server, while using the software if customer feels that some changes needs to be done on the software, they will communicate to the company. Set of developers and Test Engineers will work on this, developers will write the code according to customer requirements and give the software to Testers, where Test engineers test the changes made and give it to the customer. Customers will do Acceptance testing and deploy the software into the production server. This entire process will be done in a short duration(usually within 15-30 days). These kinds of releases are called Short release or Interim release.

**Hot fix** or Incident Management- When software is deployed into the production server and customers face critical issues which affect their business workflow. They will communicate to the company about the Blocker or critical bugs and ask to fix it ASAP, now this issue will be created in Jira and assigned to Tester. Testers will first check if this issue is reproduced in their QA environment as well, if a bug is found. Then the Developer will work on it and fix the issue, and the Tester will retest if it is really fixed or not, later Tester will do Regression Testing on high level to ensure this bug is not affecting other stories. Later he will check the bug in the QA environment and release it to the customer. This entire process will be done within a few hours. In short Hotfix is a solution provided on priority on impacted areas and it will be done for only High priority High severity issues.

**Test scenarios**

Test scenarios are high level documentation of test cases or high level documentation of the functionalities which are to be tested.

Test scenarios - What to be tested

1. To verify login Functionality >
2. To verify login functionality by entering invalid username & valid password
3. To verify login functionality by entering valid username & invalid password
4. To verify login functionality by entering invalid username & invalid password
5. To verify login functionality by entering blank username & valid password
6. To verify login functionality by entering valid username & blank password
7. To verify search functionality
8. To verify add to cart functionality

**Test scenarios for a login page:**

1) To verify if a user will be able to login with a valid username and valid password (Positive functional testing)

2) To verify if a user will be able to login with invalid username and invalid password (Negative functional testing)

3) To verify if a user will be able to login with blank username and password both (Negative functional testing)

4) To verify forgot password functionality (Positive functional testing)

5) To verify messages for invalid login (Positive functional testing)

6) To verify time taken to login with valid credentials (Performance testing)

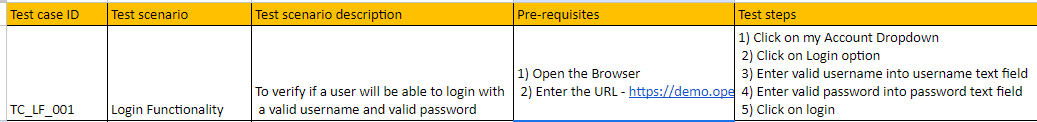
7) To verify if color, font, text, text fields of the login page is as per the standard (GUI testing)

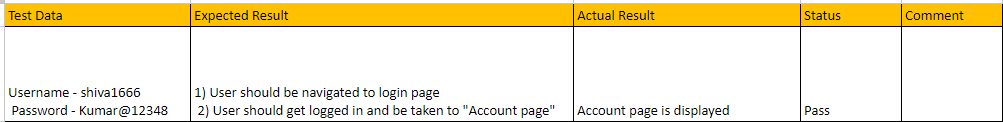
8) To verify if cancel button is available to erase the data (Usability testing)

**Test cases**

Test case is a detailed description of test scenarios used to test the application.

Let’s see how to derive the test case from test scenario for the 1st scenario mentioned above.





· Observe above template contains Test case ID, here TE needs to write Test name for every individual test cases.

· Test scenario is nothing but the feature functionality which we are testing

· Test scenario description is an documentation of the functionality we are testing

· Prerequisite are the set of criteria that must be available or satisfied before conducting testing activities

· Test steps are the action we perform step by step basis of Test scenario

· Test data are the set of data which will be a input for executing test cases

· Expected Result – These are the expected outcomes according to customer requirement

· Actual Result – Once execution is done, result we get will be specified as Actual result

· Status – If the Expected Result and Actual Result match, then status will be pass else it will be fail

· Comment – If status is fail, then we need to mention it in comments

**Why should we write Test cases**?

1. To have better test case coverage – Once a customer gives a requirement, the developer will be busy in writing the code, TE at this point of time should start writing test cases and keep it ready. So we will have better Test case coverage while testing the software.

2. To have better consistency in Testing – If we already have our test cases ready, we can test the given build by looking into it, even if developer makes any changes in future build and it affects old feature, we can look into old test cases and start testing the software, so we will have better consistency in Testing.

3. If any new TE joins the company, he can look into the test case document and start testing the application. By this we can save a lot of time on training and can utilize resources efficiently.

4. Test cases are the only proof to show to developers and customers that we have covered all possible scenarios and tested the software.

5. We will not miss any scenarios and defects if we write Test cases.

6. Testing can be done in an organized way, if test cases are written.

**Test case review process**

Test case review, is a systematic process of examining and evaluating test cases to ensure their quality, accuracy, and effectiveness. The goal of test case review is to identify and correct defects or deficiencies in the test cases before they are executed, ultimately contributing to the improvement of the overall testing process. This review process involves collaboration among team members, including testers, developers, and Test Leads.

**Key objectives of test case review include:**

**Ensuring Accuracy** – Reviewing Test cases ensures test cases are in alignment with the project requirements.

**Identifying Ambiguities**– Reviewing Test cases helps us to know if we have missed any test scenario, test cases or written wrong test cases or repeated test cases.

**Verification of Coverage** – Reviewing test cases ensures that test cases cover all the relevant functionalities within the software.

**Consistency Checks** – Reviewing test cases ensures consistency is maintained which makes test cases potentially capable of catching bugs.

**Feedback and Collaboration** – Reviewing test cases provides an opportunity for collaboration among the team members to share their insights and contribute to the improvement of Test cases.

**Process of conducting Test case Review:**

1) Prepare relevant test case documents

2) Get the test cases reviewed by Test Leads, other Testers or who is having good product knowledge

3) Once review is conducted – Check for the discrepancies, duplicate test cases or missing test cases

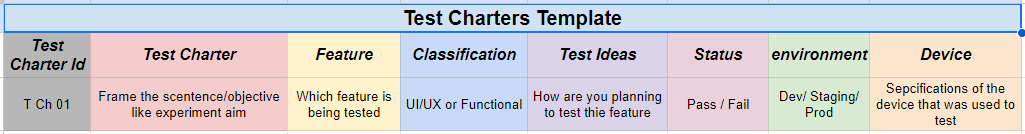
4) Ensure feedback is documented and correct the Test cases accordingly

5) Update the Test cases accordingly and ensure that the updated test cases align with the Project objectives.

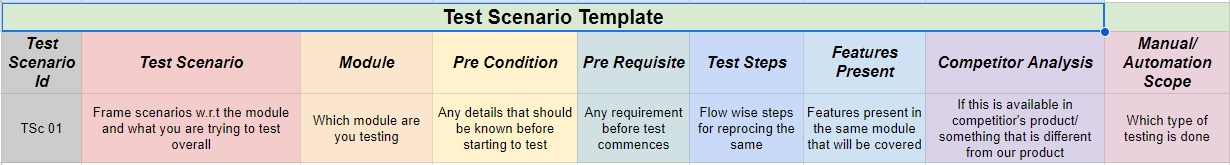
Hence, we can conclude that Test case reviews contribute to the overall quality of the testing process and the reliability of test results.

**TEMPLATES**

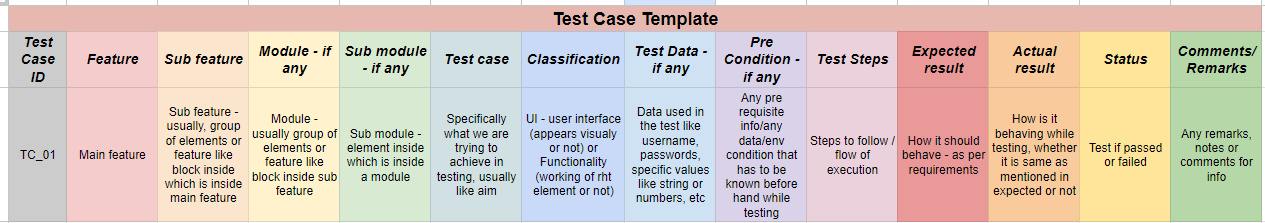
**Test charter template**

****

**Test scenario template**

****

**Test case template**

****