CHAPTER 5
REQUIREMENT ANALYSIS

1) What is called Requirement:- A requirement is a feature / functionality of software that customer / client either wants, needs or commands.

There are 2 types of requirements:-

- **Functional Requirement**:- A functional requirement is a requirement that describes what the software does. It's kind of the "entry point" from which software engineers start designing a piece of software.
  We express functional requirements in terms of:
  - Data storage.
  - Any processes that transform data.
  - Any outputs that it can produce.

- **Non-Functional Requirement**:- A non-functional requirement defines about software's Look&Feel, how much fast its response(Performance) and any limitations that the software may have.
  We express non-functional requirements in terms of:
  - **Performance:** for example, the number of transactions that the software should do in a day.
  - **Security and access**: it should comply with the law.
  - **Graphical User Interfaces** and **Scalability**
  - **Technical constraint:** run on an existing network.
  - **Project constraint:** the software should be ready within a set period.
  - **Organisational constraint:** the software should be teachable to new staff in a short amount of time.
  - **Usability** and **reliability** issues

    etc etc...

2) What's the goal of determining a requirement?

We can ask these four questions:

- What do we want the system to do? In terms of storing data, any processes, and the behaviour.
- Who are the users of the system?
- What are the needs of the users?
- What does the system need to do to achieve those needs?

3) What is Requirement Analysis:-

- The main goal of requirements analysis is to create a *document that describes the software system to be built*. Requirements analysis produces the (Software Requirement specification / System Requirement Specification) document. Here Senior BA and Technical Writer together is going to analyse the requirements mentioned in the BRS document and going to identify which is the valid and invalid requirement. They determine whether the requirement is valid or invalid with the help of some factors called as **Characteristics/ Good factors of requirement**. Later Senior BA and Technical Writer together creates the SRS document in which they mention the valid requirement in more detailed form with technical concepts in it.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.
- The requirements received from client are written in natural language. It is the responsibility of Senior Business analyst & Technical Writer together to document the requirements in technical language so that they can be comprehended and useful by the software development team.

4) What are the benefits of good requirements management?:-

This article from IBM explains it gracefully:

- Lower cost of development across the lifecycle.
- Fewer defects.
- Minimized risk for safety-critical products.
- Faster delivery.
- Reusability.
- Traceability.
- Requirements being tied to test cases.
- Global configuration management.

5) Good Factors / Characteristics of Requirements:-

- ✓ Correct
- ✓ Unambiguous
- ✓ Complete
- ✓ Verifiable/testable
- ✓ Consistent (Baselined)
- ✓ Understandable
- ✓ Modifiable (following C.M. process for next ver.)
- ✓ Traceable
- ✓ Achievable

6) Different Categories of Requirements:-

- **Explicit Requirements:-** These are the requirements explicitly stated / mentioned by the customer.
- **Implicit Requirements.:-** These are the requirements which are not explicitly stated by the customer, but implied / derived from the requirements stated by customer.

CHAPTER 6
USE CASE

What is Use case document:- Use case is a document which describes the step by step interaction between the User and a System. It is created by BA. It is mainly written for Functional Requirement to describe the procedures for doing Business process for a client. It is quiet easy to READ because it is written in language of the user. It can be visually represented as use case diagrams. It is also useful for creating acceptance test cases in coordination with customer. Use case are not created when:-

- System has few users
- System dominated by non-functional requirements.

There are 7 attributes in Use Case document:-

1. User id
2. Goal
3. Actors
4. Pre-condition
5. Path
6. Steps
7. Post-Condition

Path:- (Also called as flow)

Normal path:-(Success path)

- It is also called as Basic Path. The basic **flow** is the preferred sequence of actions in a **use case**, which delivers the desired result to a customer. This the normal flow through which the user(Actor) follows to check the functionality.
- The main flow of events describes a single path through the system. It specifies the interactions between the actor and the system for an ideal condition. It represents the most common way that the use case plays out successfully and contains the most common sequence of user-system interactions.

Alternate path:-(Success path)

- The **alternate flow** is a series of actions in non-random, repeatable order, different from the basic flow, which **still delivers the desired result** to a customer. Imagine that you want to find a friend on a social network.
- An alternate flow is a series of actions other than the basic flow that results in a user completing his or her goal. Often ,**It is considered to be an optional flow.** It means that the user has chosen to take an alternative path through the system.

Exceptional path:- (Failure path)

- In this exceptional path, the user knows the error (known error/expected error) and system knows how to handle it.
- An exceptional flow is any action that will cause the Actor to not to complete or achieve the desired result. Exception flows represent an undesirable path to the user. However, even though the exception flow has occurred the system should still react in a way that recovers the flow and provides some useful information to the user.

Error path:-(Failure path)

- In this Error path, the user doesn’t knows the error (Unknown error/unexpected error) and system doesn’t knows how to handle it.
- In Error path, it flow is any action that will cause the Actor to not to complete or achieve the desired result. Error flows represent an unexpected and undesirable path to the user. Here, when the error path is occurred the system doesn’t know to react to that situation.

CHAPTER 7
TEST DESIGN TECHNIQUES

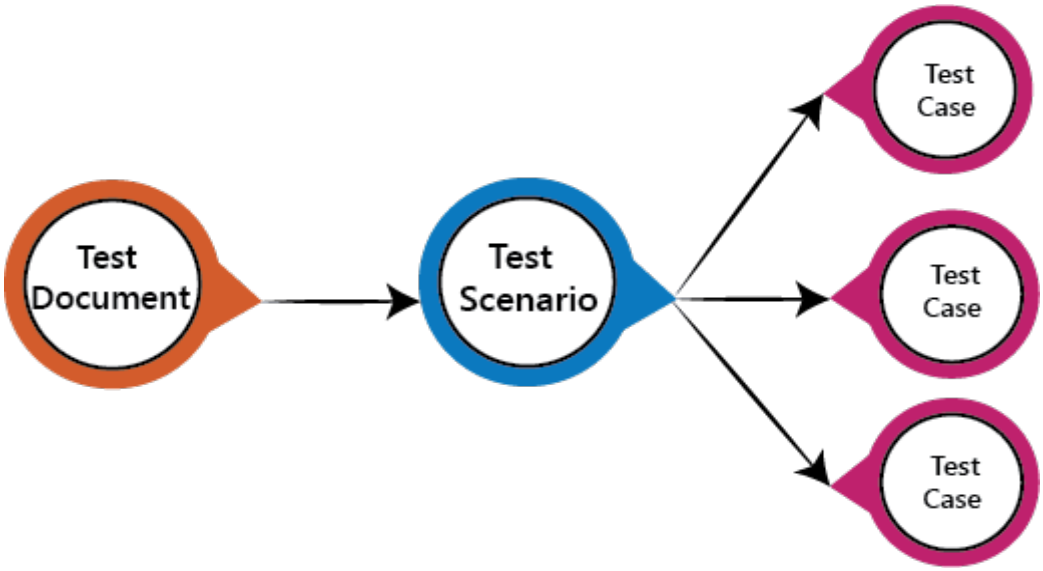
TEST SCENARIO:-

- The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.
- In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.
- In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.
- **Note:** *The test scenarios can never be used for the text execution process because it does not consist of navigation steps and input. These are the high-level documents that talks about all the possible combination or multiple ways or combinations of using the application and the primary purpose of the test scenarios are to understand the overall flow of the application.*

- **How to write Test Scenarios:- As a tester, follow the following steps to create Test Scenarios-**
  - ✓ Read the requirement document such as BRS (Business Requirement Specification)-if available, SRS (System Requirement Specification) and FRS (Functional Requirement Specification) of the software which is under the test.
  - ✓ Determine all technical aspects and objectives for each requirement.
  - ✓ Find all the possible ways by which the user can operate the software.
  - ✓ Ascertain all the possible scenario (Test Conditions)due to which system can be misused and also detect the users who can be hackers.
  - ✓ After reading the requirement document and completion of the scheduled analysis make a list of various test scenarios / Possibility / Multiple ways (Test Conditions) to verify each function of the software.
  - ✓ Once you listed all the possible test scenarios (Test Conditions), create a traceability matrix to find out whether each and every requirement has a corresponding test scenario or not.

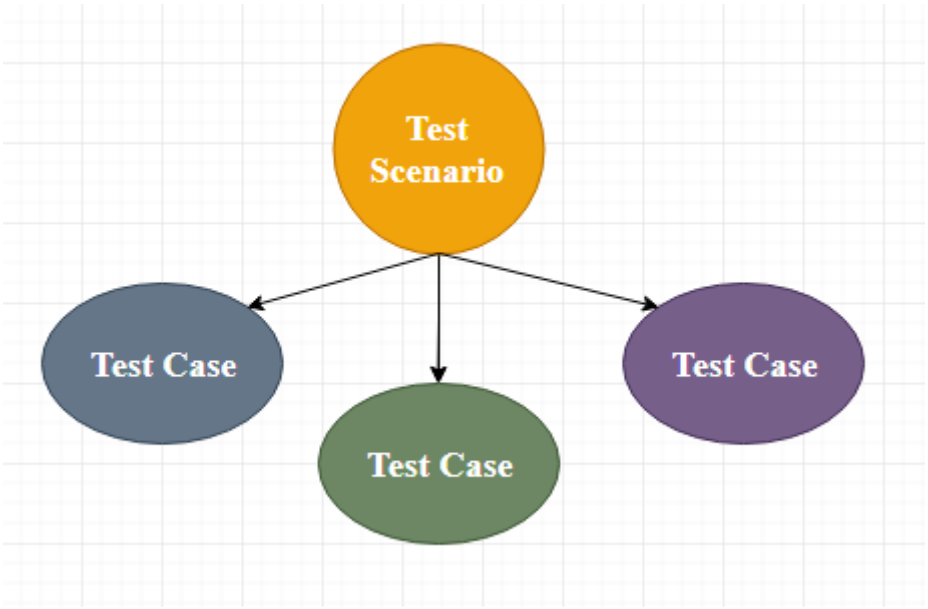
- ✓ Supervisor of the project reviews all scenarios. Later, they are evaluated by other stakeholders of the project.

- **Features of Test Scenario:-**
  - ✓ The test scenario is a liner statement that guides testers for the testing sequence.
  - ✓ Test scenario reduces the complexity and repetition of the product.
  - ✓ Test scenario means talking and thinking about tests in detail but write them in liner statements.
  - ✓ It is a thread of operations.
  - ✓ Test scenario becomes more important when the tester does not have enough time to write test cases, and team members agree with a detailed liner scenario.
  - ✓ The test scenario is a time saver activity.
  - ✓ It provides easy maintenance because the addition and modification of test scenarios are easy and independent.
- **Note:** Some rules have to be followed when we were writing test scenarios
  - ✓ Always list down the most commonly used feature (Functionality) and module by the users.
  - ✓ We always start the scenarios by picking module by module so that a proper sequence is followed as well as we don't miss out on any module level.
  - ✓ Generally, scenarios are module level for Functionality point of view.
  - ✓ Delete scenarios should always be the last option else, and we will waste lots of time in creating the data once again.
  - ✓ It should be written in a simple language.
  - ✓ Every scenario should be written in one line or a maximum of two lines and not in the paragraphs.
  - ✓ Every scenario should consist of Dos and checks/test.



TEST CASE

- The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.



- It is an in-details document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process.
- Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.
- Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.
- To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing. Then we should have the test case template to maintain the uniformity, or every test engineer follows the same approach to prepare the test document.
- Generally, we will write the test case whenever the developer is busy in writing the code.
- Once the test cases are created from the requirements, it is the job of the testers to execute those test cases. The testers read all the details in the test case, perform the test steps, and then based on the expected and actual result, mark the test case as **Pass** or **Fail**.

- **When do we write a test case?**

  We will write the test case when we get the following:
  - ✓ When the customer gives the business needs then, the developer starts developing and says that they need 3.5 months to build this product.
  - ✓ And In the meantime, the testing team will **start with writing the test cases**.
  - ✓ Once it is done, it will send it to the Test Lead for the review process.
  - ✓ And when the developers finish developing the product, it is handed over to the testing team.
- **Note:** When writing the test case, the actual result should never be written as the product is still being in the development process. That's why the actual result should be written only after the execution of the test cases.

- **TEST CASE TEMPLATE:-**
  - ✓ **Test case ID:-** A unique identifier of the test case. It is a mandatory field that uniquely identifies a test case e.g. TC_01. It is often generated automatically if creating test cases using tools. If it is assigned manually, it is advisable to make it meaningful to understand the purpose of a test case clearly.
  - ✓ **Test case Objective:-** Detailed description of the test case.
    This field defines the purpose of the test case e.g. To test/verify/check that the user can login with a valid username and valid password.
  - ✓ **Pre-condition:-** These are the necessary conditions that need to be satisfied by every testers before starting the test execution process. A set of prerequisites that must be followed before executing the test steps.
    For example – while testing the functionality of the application for login, then we can have the pre-requisite as "User must be registered in to the application".
  - ✓ **Step ID:-** An Identifier for each steps
  - ✓ **Step Description:-** Detailed steps for performing the test case.
    This is the most important field of a test case. The tester should aim to have clear and unambiguous steps in the test step description field so that some other person can follow the test steps during test execution.
  - ✓ **Input data:-** These are the values or the input we need to be created which will be used as an input for execution. For example – while testing the login functionality, the test data / Input data field can have the actual value of the username and password to be used during test execution.
  - ✓ **Expected Result:-** The expected result in order to pass the test.
    Based on the test steps followed and the test data used, we come up with the expected result e.g. the user should successfully login and navigated to home page.

- ✓ **Actual Result:-**The actual result after executing the test steps.
  This field is filled during test execution only. In this field, we write the actual result observed during the test case execution.
- ✓ **Status:- Pass/Fail** status of the test execution.
  Based on the expected result and the actual result, the test case is marked as passed or Failed.
  Apart from Pass/Fail, we can have other values also like – **Deferred/Hold** (when the test case is marked to be executed later, for some reason) and **Blocked** (when the test case execution is blocked due to some other issue in the application).
- ✓ **Remark:-** This is an optional field where the tester may define the defect in details which they found during their execution for that particular test case. Here they may also inform about the date, time of the test case execution and the person the who executed the test case

-
-
-
- hno
-