

# Software Testing Life Cycle and Test Management

---

# Outline

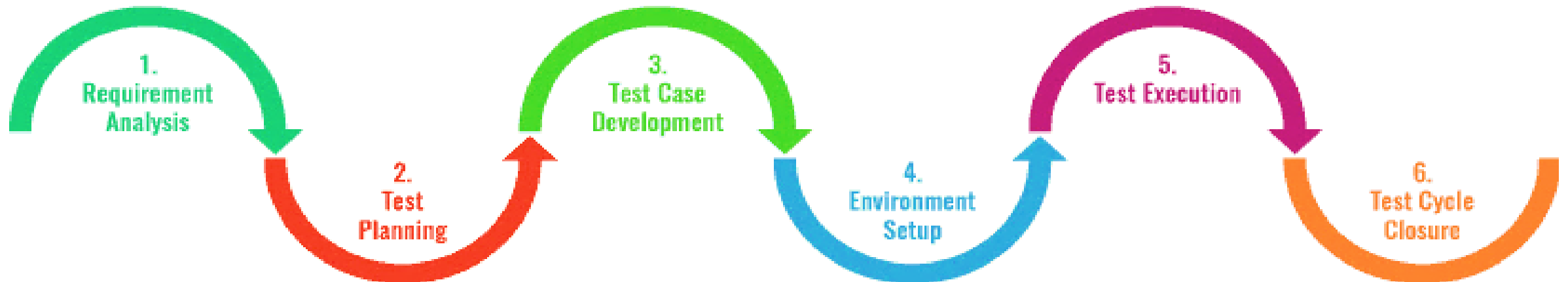
---

- Overview on STLC
  - Requirement Analysis
  - Test Planning
  - Test Design/Case Development
  - Setting up the Test Environment
  - Test Execution
  - Test Reporting/Cycle Closure
- Develop software test plan
- Explain software test management processes

# Software Testing Life Cycle (STLC)

---

- Software testing life cycle defines **the stages** in testing of software.
- It is executed in a systematic and planned manner.
- In STLC process, various activities are carried out to improve the quality of the product.
- STLC, in general, comprises of the following phases:



# 1. Requirement Analysis

---

- Before we go for designing any software, we need to analyze the requirements. Like
  - What is the **purpose** of the application?
  - **How many users** are going to use this?
  - What is the **traditional process** of the client and how our software is going to help our client?
- Here testing team gathers as much information as possible about the software which they are going to test.
- It's obvious that if the testing team has good knowledge about the software, then it can test it very well.

## 2. Test Planning

---

Test plan has to have sufficient information on:

1. How the testing will be done?
2. The test strategy to be followed?
3. What test methods will be followed?
4. When to stop testing? and
5. Its resource requirements:
  - hardware,
  - software,
  - money &
  - time

# 3. Test Design

---

- After reading and understanding the software requirement, the tester writes
  - the test scenarios - on the basis of requirements
  - the test cases and then
  - the test data (test data can be provided by the client also)
  - the test scripts
  - the requirements traceability matrix

# 4. Test Environment Setup

---

- Every software is developed by considering hardware and software requirement.
- Here we setup the testing environment (e. g. server/client/network, or install the setup if it is window's application etc.) with the goal of replicating the end-users' environment.
- This step decides on which Platform/OS the tester needs to perform testing of project.
  - Example: Use only Internet Explorer, Resolution must be like 136 x 768, or in many large applications they indicate the minimum hardware and software requirement like Photoshop, latest version of Matlab, Oracle, large games etc.
- In this phase, according to requirements, they configure required hardware and software.
- Tester may or may not involve in setting the configuration.

# 5. Test Execution

---

- The tester has to execute his written test cases against the software that he has written during the test case design phase.
- The tester **executes** the test cases, **checks** the response of the software and **verify** the response of software whether is it as expected or not.
- After executing each test cases, the tester verifies the output and if that output is not as expected then it means that the test case has **failed and a bug** has found then tester report that bug to team lead. Then bugs will be reported back to the development team for **correction and retesting** will be performed.



## 6. Test Reporting

---

- This is the last phase of software testing life cycle. It is conducted after testing and releasing of the software.
- During this phase, **all the testing team meet and discuss** about the software and the issues they faced during the testing.
- Also they **collect all the testing artifacts (documents)**. This meet up will help the testing team to improve the testing process
- A **bug report** of different bug description is forwarded to respective developer and the test manager & other seniors.

# STLC Phases, activities and deliverables

Phase	Activity	Deliverables
<b>Requirements Analysis</b>	You review the software requirements/design	‘Review Defect’ Reports
<b>Test Planning</b>	Once you have gathered a general idea of what needs to be tested, you ‘plan’ for the tests.	Test Plan Test Estimation Test Schedule
<b>Test Designing</b>	You design detail tests on the basis of detailed requirements/design of the software	Test Cases / Test Scripts /Test Data Requirements Traceability Matrix
<b>Test Environment Setup</b>	You setup the test environment (server/client/ network, etc.) with the goal of replicating the end-users’ environment.	Test Environment
<b>Test Execution</b>	You execute your Test Cases/ Scripts in the Test Environment to see whether they pass.	Test Results Defect Reports
<b>Test Reporting</b>	You prepare various reports for various stakeholders.	Test Results (Final) Test/ Defect Metrics Test Closure Report Who Worked Late & on Weekends

# Software Test Management

---

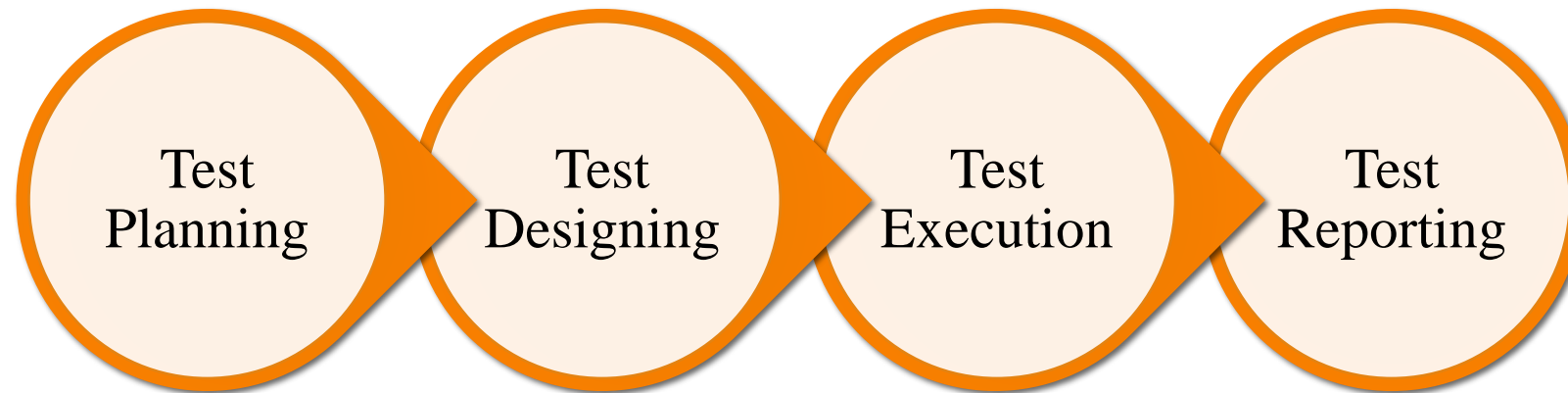
# Software Test Management

---

- Software test management is a process of planning, executing, monitoring and controlling software testing activities.
- It is the practice of **organizing and controlling the processes** and **artifacts** required for the testing effort throughout the project cycle.
- Software test management often has **multifunctional capabilities** such as **testware management, test scheduling, the logging of results, test tracking, incident management and test reporting.**
- It uses manual and automated tools.

# Software Test Management Processes

---



# Test Planning

---

## PROCESS 1

# 1. Software Test Planning

---

- Software test planning is the practice of documenting software testing requirements in an organized manner.
- The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

# ...Test planning

---

- A test plan is a detailed document that outlines the
  - ✓ test tasks, (what)
  - ✓ test strategy, (how)
  - ✓ test approach, (how)
  - ✓ testing objectives, (why)
  - ✓ resources (manpower, software, hardware) required for testing,
  - ✓ test schedule, (when)
  - ✓ test estimation and (how much)
  - ✓ test deliverables. (outputs)



# Importance of Software Test Plan

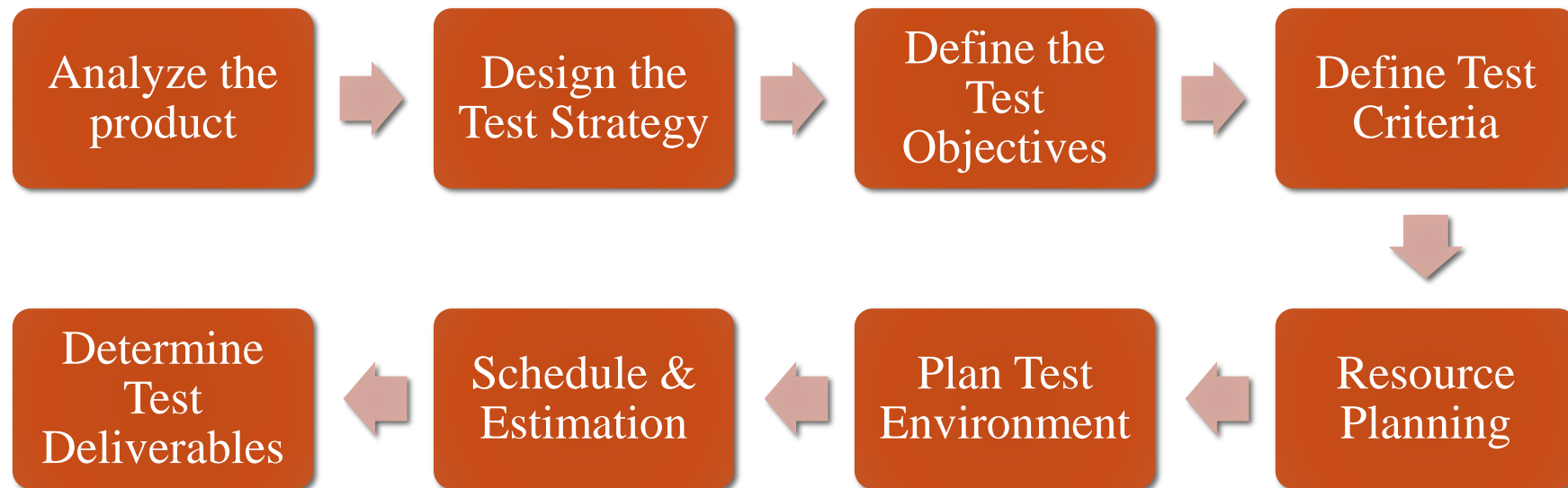
---

- 1) Test plan **guides** the testing phases to be followed.
- 2) Test plan helps us determine the **effort** needed to validate the quality of the application under test
- 3) Helps people **outside the test team** such as developers, business managers, customers **understand the details of testing**.
- 4) It can be **reviewed** by the management team and **re-used** for other projects.

# How to write a Software Test Plan

---

- Follow the eight steps below to create a test plan as per [IEEE 829 standard](#)



# I. Analyze the product

---

- You must learn a product **thoroughly** before testing it. How can you test a product **without** any information about it? The answer is **impossible**.
- You should research **clients** or **the end users** to know their **needs and expectations** from the application.
  - ✓ Who will use the software?
  - ✓ What is it used for?
  - ✓ How will it work?
  - ✓ What software/ hardware the product uses?

# ... Analyze the product

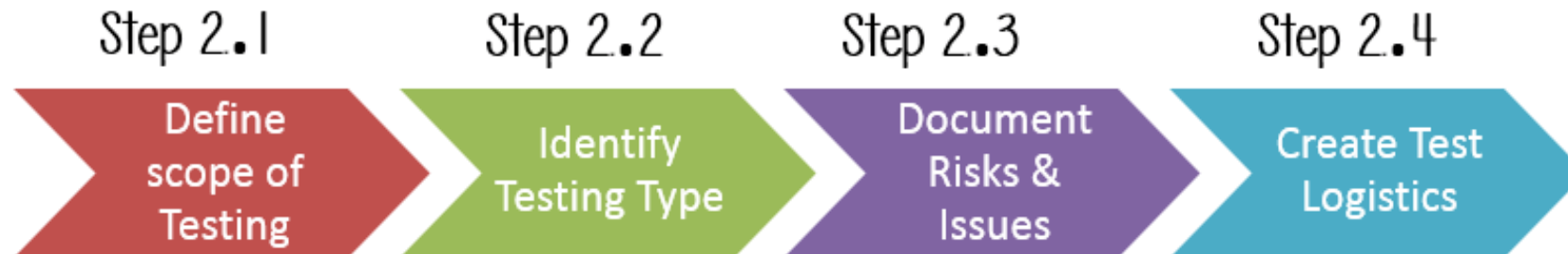
---

- For instance, the product under test is Guru99.
- You can use the following approach to analyze the site
- Now let's apply the above information to a real product:
  - **Analyze** the banking website <http://demo.guru99.com/V4>.
  - You should take a **look around** this website and also **review** [product documentation](#).
  - Review of product documentation helps you understand all the features of the website as well as how to use it. If you are unclear on any items, you might **interview** customers, developers, designers to get more information.

## II. Develop Test Strategy

---

- Test strategy is a **critical step** in making a test plan.
- A test strategy is a high-level document which is usually developed by test manager.
- This document defines:
  - The project's **testing objectives** and **the means** to achieve them
  - Determines testing **effort** and **costs**
- You should follow the steps below



## 2.1. Define Scope of Testing

---

- Before the start of any testing activity, scope of the testing should be known. You must think hard about it.
- The components of the system to be tested (hardware, software, middleware, etc.) are defined as "**in scope**"
- The components of the system that will not be tested also need to be clearly defined as being "**out of scope**."
- Defining the scope of your testing project is very important for all stakeholders. A **precise scope** helps you
  - Give everyone **confidence & accurate information** of the testing you are doing
  - All project members will have **clear understanding** about what is tested and what is not

# How you determine scope of

---

- To determine scope, you must have
  - ✓ Precise customer requirement
  - ✓ Project budget
  - ✓ Product specification
  - ✓ Skills and talent of your test team
- For instance, if a customer wants you to conduct stress and load testing, but the project budget does not permit to do so. In such a case what will you do?

# ...How do you determine scope of your project?

---

- Well, in such case you need to convince the customer that **stress and load testing** is extra work and will consume significant resources. Give him data supporting your facts. Tell him if we conduct these tests, the budget will increase by XYZ amount. If the customer disagrees, you have to modify your “in scope” and “out of scope” testing
- As the software requirement specs, only focus on testing the **functions** and external interface (**in scope** testing)
- Non-functional testing such as **stress and load testing** currently will not be tested. (**out of scope**)



## 2.2. Identify Testing Type

---

- A **testing type** is a standard test procedure that gives an expected test outcome.
- Each testing type is formulated to identify a specific type of product bugs. But, all testing types are aimed at achieving one common goal “**early detection of all the defects before releasing the product to the customer**”
- Your team **cannot have** enough efforts to handle all kind of testing.
- As test manager, you must set **priority** of the testing types
  - Which testing types should be **focused**?
  - Which testing types should be **ignored** for saving cost?

## 2.3. Document Risk & Issues

- Risk is future's **uncertain event** with **probability of occurrence** and a **potential for loss**.
- When the risk actually happens, it becomes the **'issue'**.
- In the test plan, you will document those risks.

Risk	Mitigation
Team members lack the required skill for website testing.	Plan training courses to skill up your members
The project schedule is too tight; it's hard to complete this project on time	Set test priority for each of the test activity.
Test manager has poor management skill	Plan leadership training for the manager
Lack of cooperation negatively affects your employees' productivity	Encourage each team member in his task, and inspire them to greater efforts.
Wrong budget estimate and cost overruns	Establish the scope before beginning work, pay a lot of attention to project planning and constantly track and measure the progress

## 2.4. Create Test Logistics

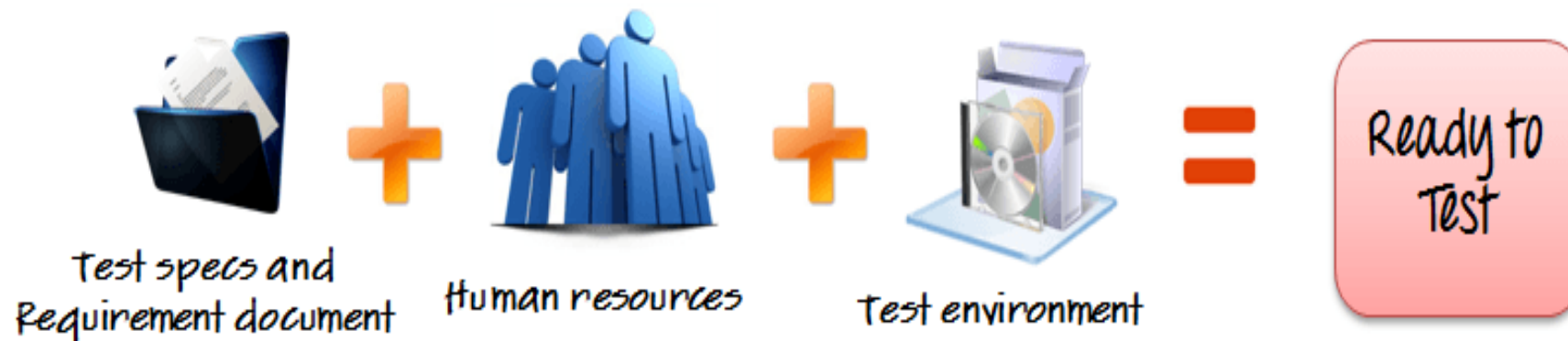
---

- In test logistics, the test manager should answer the following questions:
  - **Who will test?**
  - **When will the test occur?**
- You may not know exact names of the tester who will test, but the **type of tester** can be defined.
- To select the right member for specified task, you have to consider if his skill is qualified for the task or not, also estimate the project budget. Selecting wrong member for the task may cause the project to **fail** or **delay**.
- A person having the following skills is most ideal for performing software testing:
  - **Ability to understand customers point of view**
  - **Strong desire for quality**
  - **Attention to detail**
  - **Good cooperation**
- In your project, the member who will take in charge for the test execution is the **tester**. Based on the project budget, you can choose in-source or outsource member as the tester.

# ...When will the test occur?

---

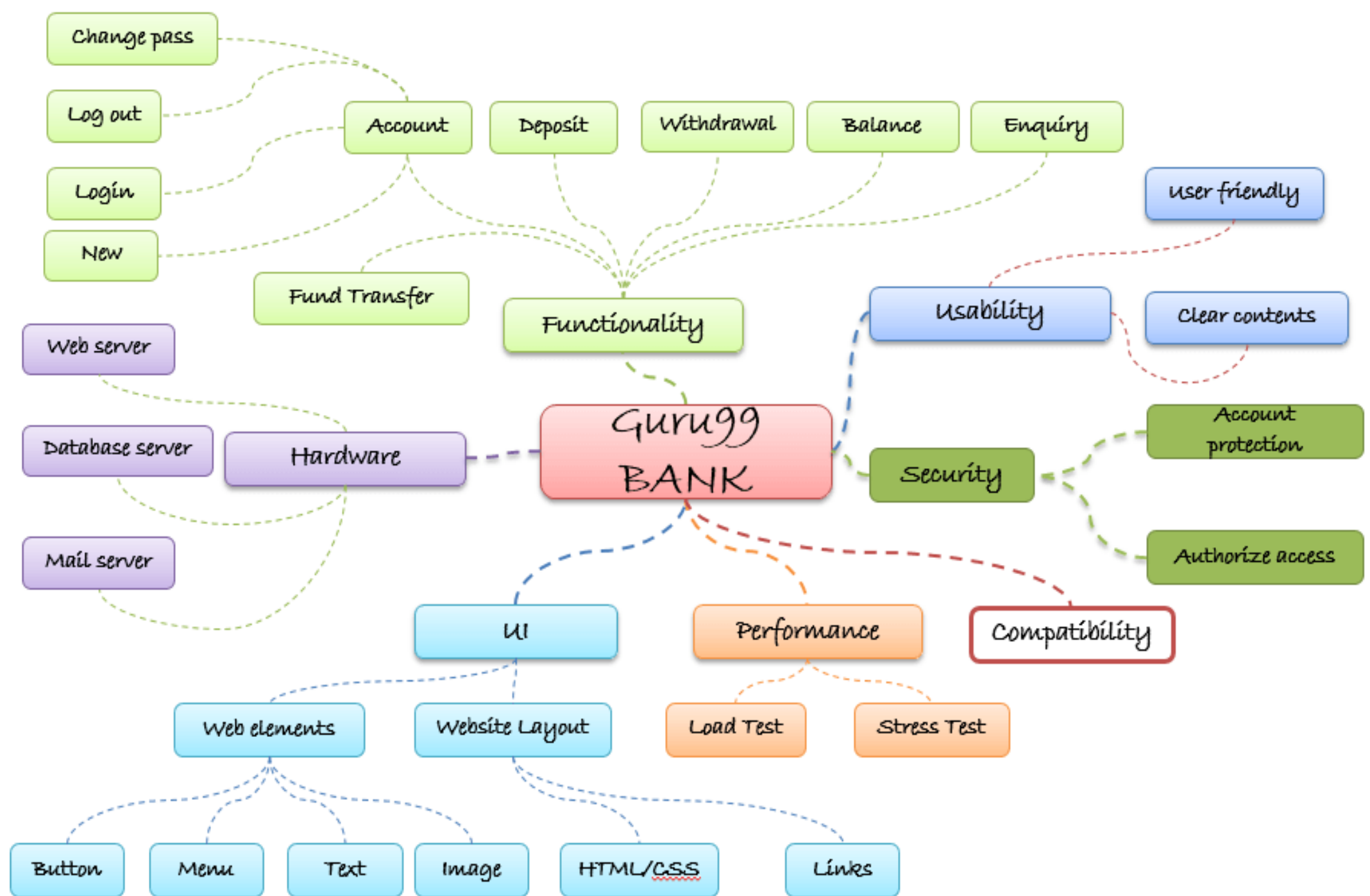
- Test activities must be matched with associated development activities.
- You will start to test when you have **all required items** shown in the following figure



# III. Define Test Objective

---

- Test objective is the overall goal and achievement of the test execution. The objective of the testing is
  - finding as many software defects as possible;
  - ensure that the software under test is **bug free** before release.
- To define the test objectives, you should do the following two steps:
  1. List all the software features (functionality and non-functional) which will be tested.
  2. Define the **target** or the **goal** of the test based on the above features



## ... Define Test Objectives

---

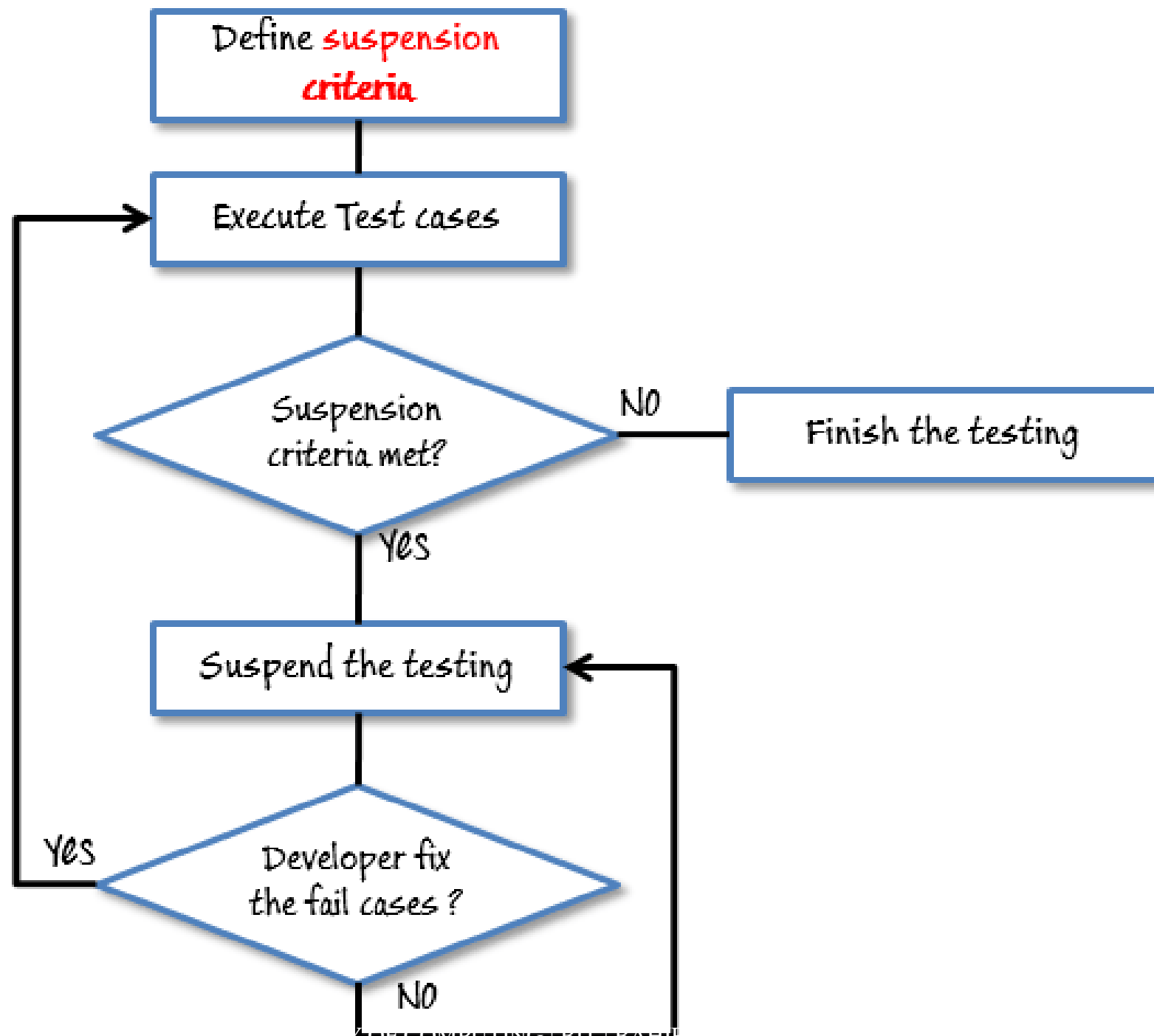
- This figure shows all the features which the guru99 website may have.
- Based on above the features, you can define the test objective of the project guru99 as follows
  - Check that whether website guru99 **functionality**(account, deposit...) is working as expected without any error or bugs in the real business environment
  - Check that the external interface of the website such as **UI** is working as expected and meet the customer needs
  - Verify the **usability** of the website. Are those functionalities convenient for user or not?

# IV. Define Test Criteria

---

- Test criteria is a standard or rule on which a test procedure or test judgment can be based. There're two types of test criteria:  
**(a) Suspension criteria**
- Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**.
- Example: if your team members report that **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.





# ... Define Test Criteria

---

## (b) Exit Criteria

- It specifies the criteria that denote a **successful** completion of a test phase.
- The exit criteria are the targeted results of the test and are necessary before proceeding the next phase of development.
- Example: **95%** of all critical test cases must pass.

# ...Define Test Criteria

---

- Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.
- **Run rate** - is the ratio between **number of test cases executed/total test cases** of test specification.
  - For example, the test specification has a total of 120 TCs, but the tester only executed 100 TCs. So, the run rate is  $100/120 = 0.83$  (83%)
- **Pass rate** - is the ratio between **number of test cases passed / test cases executed**.
  - For example, in the above example, 100 TCs executed and 80 TCs passed. So, the pass rate is  $80/100 = 0.8$  (80%)

## ... Define Test Criteria

---

- This data can be retrieved in **test metric** documents.
- **Run rate** is mandatory to be **100%** unless a clear reason is given.
- **Pass rate** is dependent on project scope, but achieving **high pass rate** shall be a goal.

# ..... Define Test Criteria

**Example:** Your team has already done the test executions. They reported the test result to you, and they want you to confirm the **Exit Criteria**.

In this case, the run rate is mandatory to be **100%**, but the test team only completed 90% of test cases. It means the run rate is not satisfied, so do NOT confirm the Exit Criteria.



# V. Resource Planning

---

- Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be **human, physical (equipment and materials) and financial** needed to complete the test.
- The test manager can make the correct **schedule & estimation** for the project based on the resources he has.
- This section represents the recommended resources for your project.

## Human Resource

This table shows various members in your test project team

No.	Member	Tasks
1.	Test Manager	Manage the whole software testing project Define project directions Acquire appropriate resources
2.	Tester	Identifying and describing appropriate test techniques/tools/automation architecture Verify and assess the Test Approach Execute the tests, Log results, Report the defects. Tester could be in-sourced or out-sourced members, base on the project budget For the task which required low skill, I recommend you choose outsourced members to save project cost.
3.	Developer in Test	Implement the test cases, test program, test suite etc.
4.	Test Administrator	Builds up and ensures Test Environment and assets are managed and maintained Support Tester to use the test environment for test execution
5.	QA members	Take in charge of quality assurance Check to confirm whether the testing process is meeting specified requirements

## Physical Resource

For testing, a web application, you should plan the resources shown in this table:

No.	Resources	Descriptions
1.	Server	Install the web application under test. This includes a separate web server, database server, and application server (if applicable)
2.	Test tool	The testing tool is to automate the testing, simulate the user operation, generate the test results. There are tons of test tools you can use for this project such as Selenium, QTP...etc.
3.	Network	You need a Network including LAN and Internet to simulate the real business and user environment
4.	Computer	The PC that users often use to connect the web server



# Financial Resources

---

- To estimate the financial requirements of a test, you need the following items:
  1. List of **activities** and their estimated direct and indirect costs must be calculated
  2. **Items** to be purchased and their estimated direct and indirect costs must be calculated
  3. **Human resource** to be hired
- There are three types of cost estimation techniques:
  - Preliminary cost estimation - accuracy (-25%, +75%)
  - Budgetary cost estimation - accuracy (-10%, +25%)
  - Definitive cost estimation - accuracy (-5%, +10%)

# VI. Plan Test Environment

---

- A testing environment is a setup of software and hardware on which the testing team is going to execute test cases.
- The test environment consists of **real business** and **user environment**, as well as **physical environments**, such as server, front end running environment.

# How to setup the test environment

---

- To finish this task, you need **a strong cooperation** between test team and development team
- You should ask the developer some questions to understand the web application under test **clearly**.
- Here're some recommended questions. Of course, you can ask the other questions if you need.
  1. *What is the maximum user connection which this website can handle at the same time?*
  2. *What are hardware/software requirements to install this website?*
  3. *Does the user's computer need any particular setting to browse the website?*

# VII. Schedule & Estimation

---

In the test estimation phase, suppose you break out the whole project into **small tasks** and add the estimation for each task as shown below

Task	Members	Estimate effort
Create the test specification	Test Designer	170 man-hour
Perform Test Execution	Tester, Test Administrator	80 man-hour
Test Report	Tester	10 man-hour
Test Delivery		20 man-hour
Total		280 man-hour

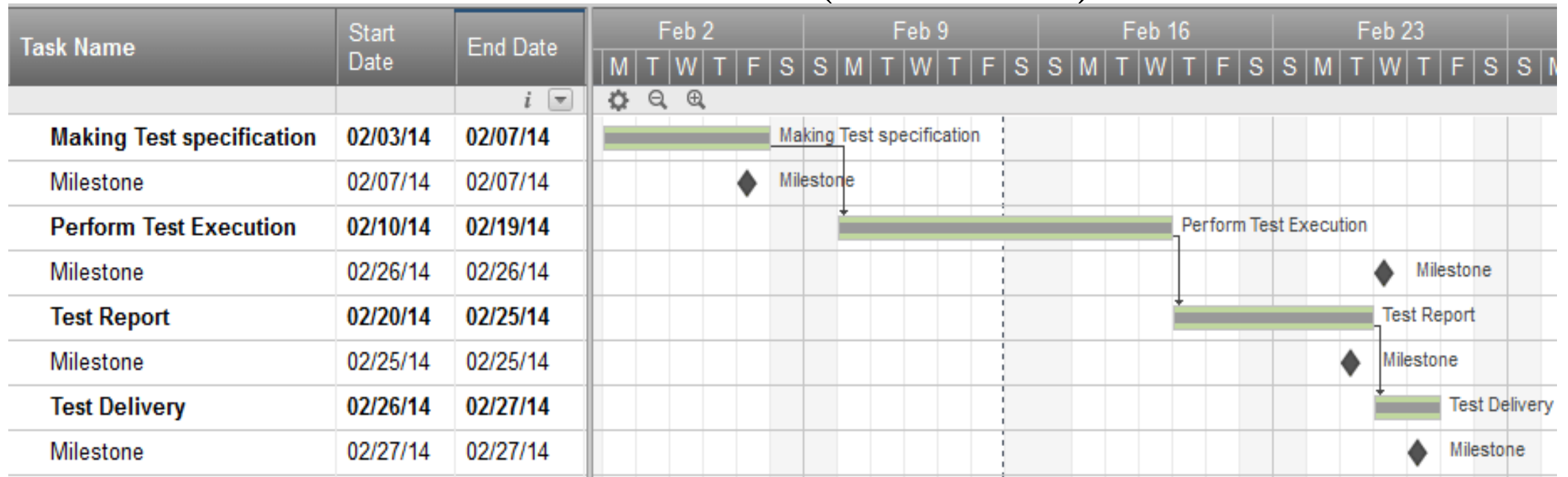
# ... Schedule & Estimation

---

- Then you create the **schedule** to complete these tasks.
- Making schedule is a common term in project management. By creating a solid schedule in the test planning, the test manager can use it as tool for monitoring the project progress, control the cost overruns.
- To create the project schedule, the test manager needs several types of input such as:
- **Employee and project deadline:** the working days, the project deadline, resource availability are the factors which affected to the schedule
- **Project estimation:** base on the estimation, the test manager knows how long it takes to complete the project (Expected time calculation). So, he can make the appropriate project schedule
- **Project risk:** understanding the risk helps test manager add enough extra time to the project schedule to deal with the risks. (Buffer)

# ... Schedule & Estimation

Suppose the boss wants to complete a project in one month, you already estimated the effort for each task in Test Estimation. You can create the schedule as shown below. (Gantt Chart)



# VIII. Determine Test Deliverables

---

- Test deliverables are list of all the documents, tools and other components that have to be developed and maintained in support of the testing effort.
- There are different test deliverables at every phase of the software testing process.
- They can be grouped into three.

- 1) Test deliverables provided **before** the testing phase.
  - Test plans document.
  - Test cases documents
  - Test design specifications.
- 2) Test deliverables are provided **during** the testing
  - Test scripts
  - Simulators.
  - Test data
  - Test traceability matrix
  - Error logs and execution logs.
- 3) Test deliverables are provided **after** the testing cycles is over.
  - Test results/reports
  - Defect report
  - Release notes



# Types of Test Plan

---

- One can have the following types of test plans:
  1. **Master Test Plan:** a single high-level test plan for a project/ product that unifies all other test plans.
  2. **Testing-Level-Specific Test Plans:** Plans for each level of testing.
    - Unit Test Plan
    - System Test Plan
    - Integration Test Plan
    - Acceptance Test Plan
  3. **Testing-Type-Specific Test Plans:** Plans for major types of testing like Performance Test Plan, Security Test Plan, etc.

# Outline of a test plan

---

Test plan format and content may vary depending upon different standards. The following format details the points usually covered in a test plan.

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Roles and Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

# Brief description

---

- **Test Plan Identifier:** Provides a unique identifier for the document. Every deliverable has a unique identification number which could be numeric or alphanumeric based on the company configuration management.
- **Introduction:** Brief introduction about the project and objective of the current release. Project could be platform configuration tool and objective could new mobile App interface or new feature / enhancement in existing product or defect fixes.
- **Test item:** Introduction and overview of Software Under Test.
- **Features to test:** In scope features. This could be newly added or updated features. Indirect features that have technical or functional dependency on newly added or updated features.

# Brief description

---

- **Features not to test:** Out of scope features. Excluded product features from current Test Plan. [Note: Provide reasoning for exclusion, like, non-impacted / less impacted / less priority features, as applicable.]
- **Approach:** Strategy to test the software. Includes types of tests and how to test. Functional, performance, security testing using combined [manual + automation], manual only, automation only approach.
- **Test deliverables:** All the deliverables from the testing e.g. approaches, test cases, reports etc.
- **Item pass/fail criteria:** Entry and Exit criteria for all items. E.g.
  - Test Case: All Steps passed
  - Feature: All test cases executed and no defects are detected.
- **Testing tasks:** All tasks / steps to execute for test planning and execution
- **Environmental needs:** Infrastructure required for application and testing.

# Brief description

---

- **Responsibilities:** Roles and responsibilities for various testing / supported activities.
- **Staffing and training needs:** Training / hiring needs to bridge the gap of available and expected skill.
- **Schedule:** Test estimation (Efforts) and high-level schedule. Schedule should be for key deliverables or important milestones. Ideally, all test deliverables included in the test plan should be scheduled. Detailed test schedule (at feature or defects or resource level) is prepared at appropriate time during test execution.  
Risks and Mitigation: Risk identification for applicable items, assumptions, and mitigation plan.
- **Approvals:** Approvals and sign of dates.

*Test plan is a guideline based on which test execution should be tracked. For successful testing and good product test delivery, it is important to update and make required changes in the plan as per changes in the any of the parameter which was basis of the test plan.*

# Test Design

---

## PROCESS 2

# Test Design

---

- In software engineering, test design is the process of creating and writing test artifacts for testing a software.
- Tasks include:
  - ✓ Identifying **test basis**
  - ✓ Developing **test scenarios**
  - ✓ Identifying and describing **test cases**
  - ✓ Developing **test suite**
  - ✓ Identifying and structuring **test scripts**
  - ✓ Reviewing and accessing test coverage using **traceability matrix**

# Test Basis

---

- Test basis is the source to create test scenarios and test cases.
- It can be
  - the application itself or
  - the requirement documents such as
    - SRS (Software Requirement Specification),
    - BRS (Business Requirement Specification), etc.

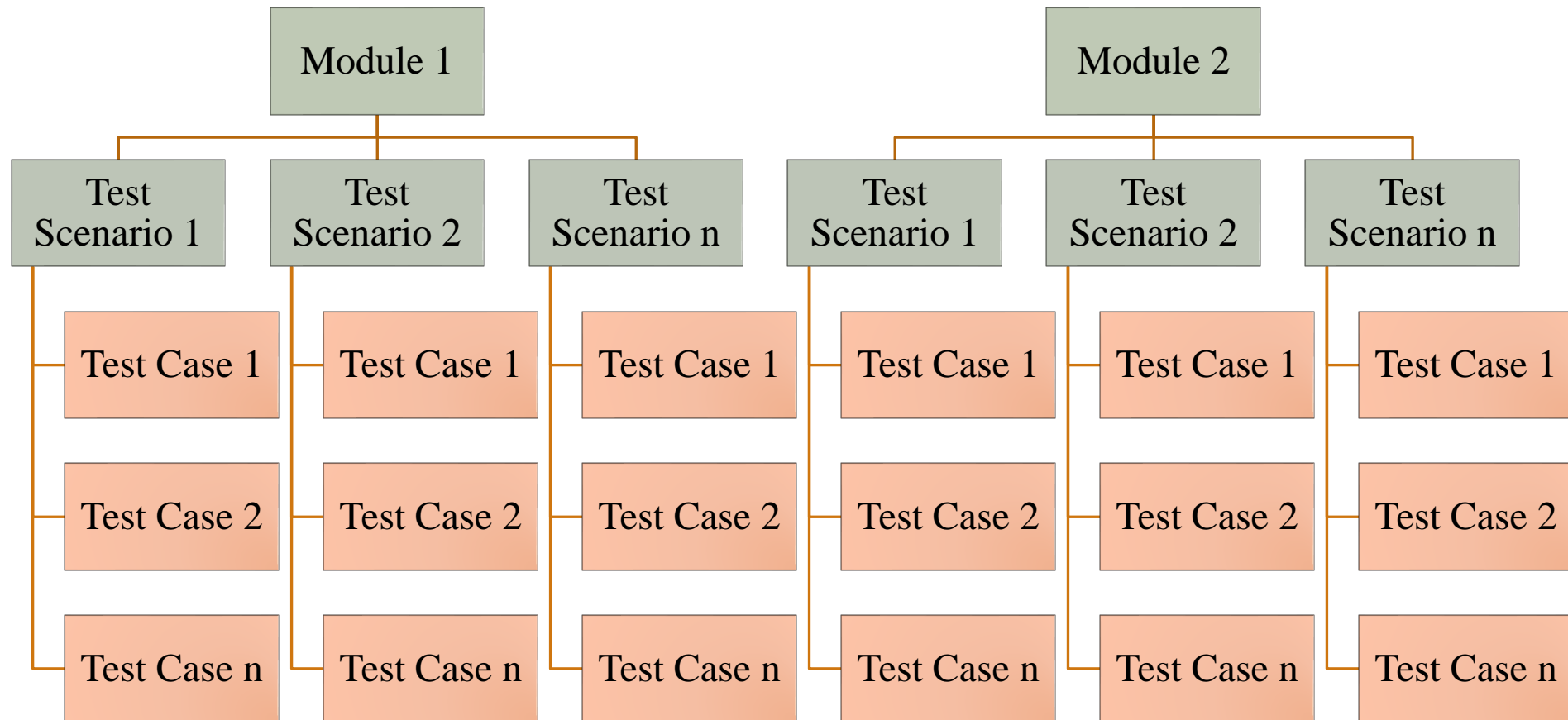


# Test Scenario

---

- A test scenario is **any functionality that can be tested**.
- It is also called **test condition** or **test possibility**.
- As a tester, you may put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test (AUT).
- **Test scenario** is **what to be tested** and a **test case** is **how to be tested**.

# ...Test Scenario



# ...Test Scenario

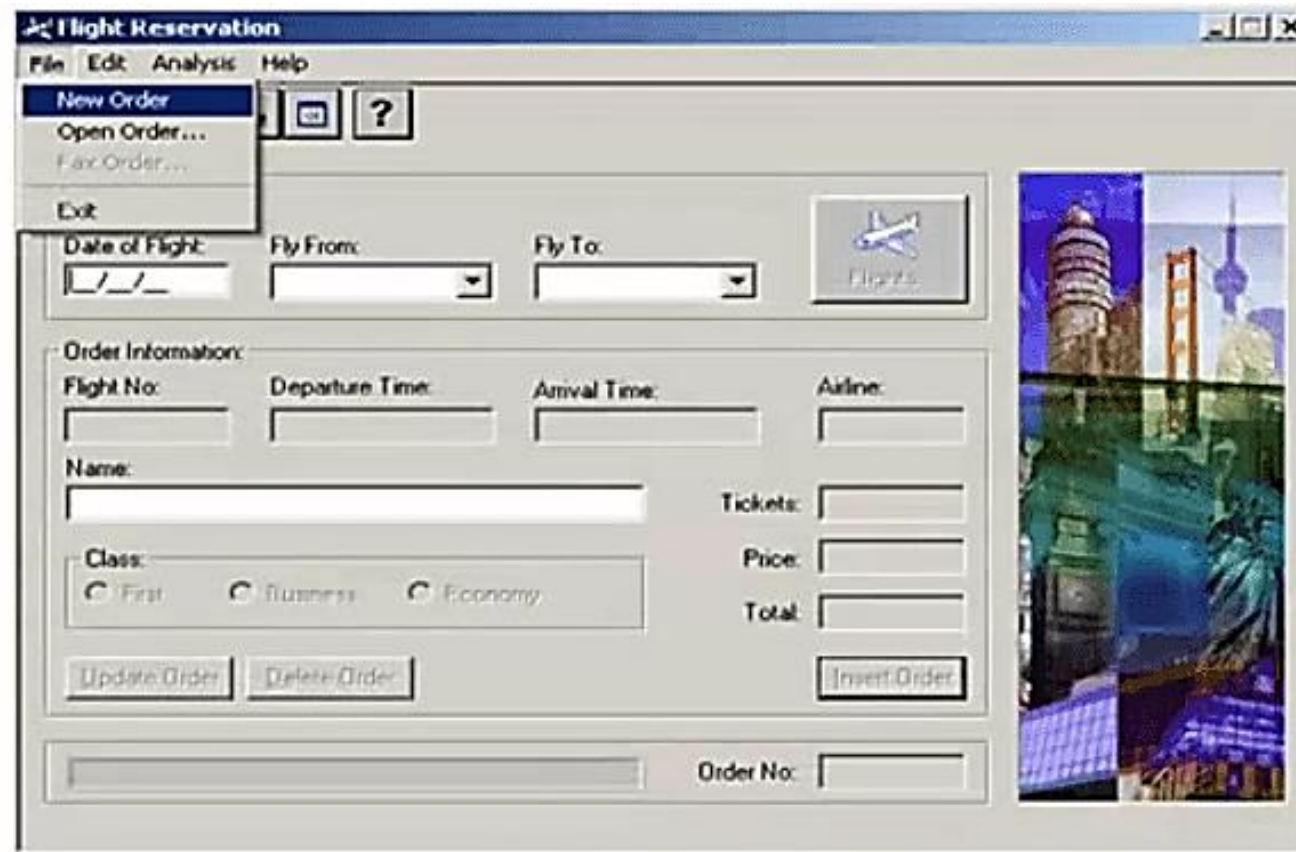
---

- **Example 1: Test Scenario for Flight Reservation**
  - For the Flight Reservation Application, a few test scenarios would be
  - **Test Scenario 1: Check the Login Functionality**



# ...Test Scenario

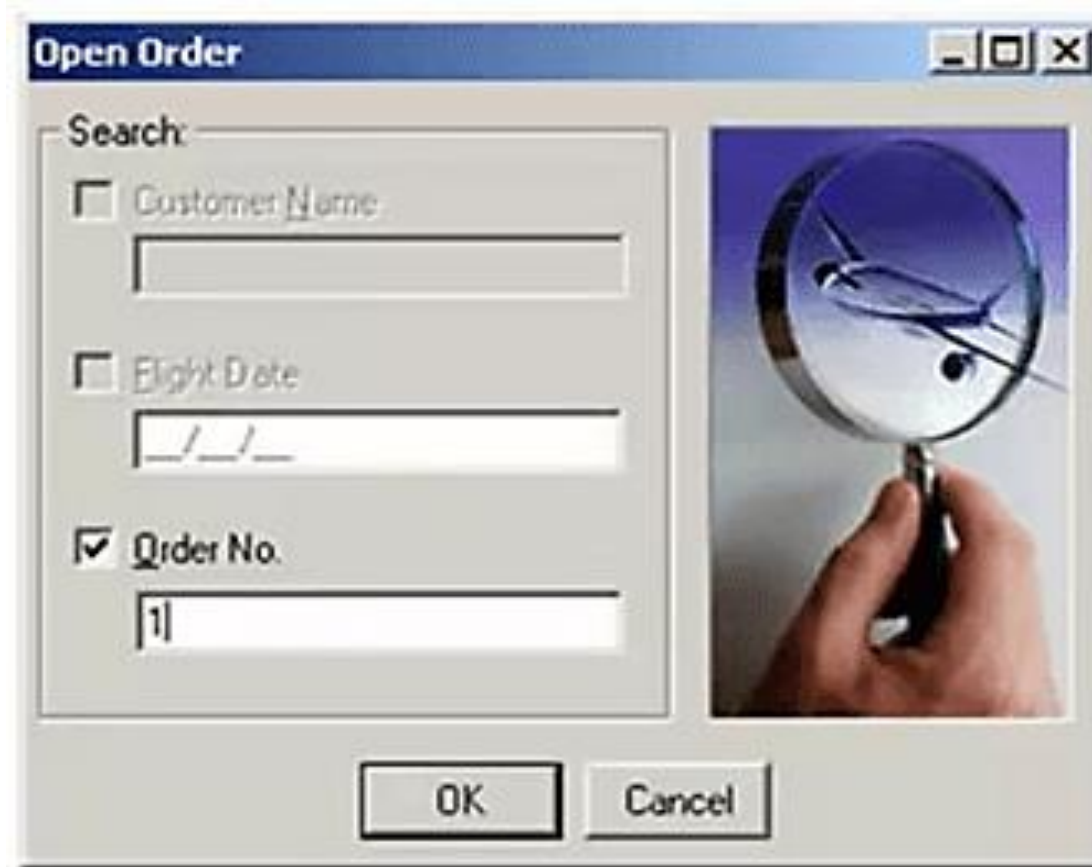
- **Test Scenario 2:** Check that a New Order can be created



# ...Test Scenario

---

- **Test Scenario 3:** Check that an existing Order can be opened



**Open Order**

Search:

☐ Customer Name

☐ Flight Date

☒ Order No.

1

OK Cancel

# ...Test Scenario

- **Test Scenario 4:** Check that a user, can FAX an order

The image shows a Windows-style dialog box titled "Fax Order No. 1". It contains several input fields for flight information. The "Name:" field is filled with "John Doe". The "Order:" field is "1", "Flight:" is "6232", and "Date:" is "10/11/09". The "From:" field is "Denver", "Departure:" is "10:24 AM", "To:" is "Los Angeles", and "Arrival:" is "01:24 PM". The "Class:" is "First", "# Tickets:" is "1", "Ticket Price:" is "312.00", and "Total:" is "312.00". There is a "Fax Number:" field with a placeholder "1.1.1." and an "Agent Signature:" area with a large empty box. A checkbox labeled "Send Signature with order" is at the bottom left. At the bottom are four buttons: "Preview Fax", "Send", "Cancel", and "Clear Signature".

Fax Order No. 1			
Name:	Order:	Flight:	Date:
John Doe	1	6232	10/11/09
From:	Departure:	To:	Arrival:
Denver	10:24 AM	Los Angeles	01:24 PM
Class:	# Tickets:	Ticket Price:	Total:
First	1	312.00	312.00
Fax Number:	Agent Signature:		
1.1.1.			
<input type="checkbox"/> Send Signature with order			
[Preview Fax] [Send] [Cancel] [Clear Signature]			

# ...Test Scenario

- **Test Scenario 5:** Check that the information displayed in the HELP section is correct





# ...Test Scenario

---

- **Test Scenario 6:** Check that the information displayed in About section, like version, programmer name, copy right information is correct





# ...Test Scenario

---

- Apart from these six scenarios, here is the list of all other scenarios
  - Update Order
  - Delete Order
  - Check Reports
  - Check Graphs and so on.
- We have already learned **exhaustive testing is not possible**.
- Suppose you have time only to execute 4 out of those 6 scenarios which two low priority scenarios of these six will you eliminate.
- I am sure most of you would have guessed scenarios 5 & 6 since they are not the core functionality of the application. This is nothing but **test prioritization**.

# Why to create test scenarios?

---

- Test scenarios are created for the following reasons:
  - To ensure complete test coverage
  - To get approval from various stakeholders like business analyst, developers, customers to ensure the application under test is thoroughly tested. It ensures that the software is working for the most common use cases.
  - To quickly determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
  - To determine the most important end-to-end transactions or the real use of the software applications.

# When **not** to create test scenario?

---

- Test scenarios may not be created when
  - ✓ The application under test is **complicated, unstable** and there is a **time crunch** in the project.
  - ✓ Projects that follow **agile methodology** like Scrum, Kanban may not create test scenarios.
  - ✓ Test scenario may not be created for **a new bug fix or regression** testing. In such cases, test scenarios must be already heavily documented in the previous test cycles.

# How to create test scenario?

---

- As a tester, we follow the following five steps to create test scenarios:
- **Step 1:** Read the Requirement Documents like **BRS, SRS** of the Application Under Test (AUT). You could also refer uses cases, books, manual, etc. of the application to be tested.
- **Step 2:** For each requirement, figure out possible **users actions** and **objectives**. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker's mindset.
- **Step 3:** After reading the requirements document and doing your due analysis, **list out different test scenarios** that verify each feature of the software.
- **Step 4:** Once you have listed all possible Test Scenarios, a **Traceability Matrix** is created **to verify** that each & every requirement has a corresponding Test Scenario
- **Step 5:** The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other stakeholders in the project.

# Test Case

---

- A test case is a set of actions executed to verify a particular feature or functionality of your software application.
- A test case is a document which consists of a set of conditions or actions which are performed on the software application in order to verify the expected functionality of the feature.
- Here we describe the end to end logical flow of a specific requirement with test data, prerequisites and expected results.
- Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

# ...Test case

---

- Now, consider the Test Scenario: Check Login functionality there are many possible cases like
  - Test Case 1: Check results by **entering** valid User Id & Password
  - Test Case 2: Check results by **entering** Invalid User ID & Password
  - Test Case 3: Check response **when User ID is Empty** & Login Button is pressed, and many more.

# Format of Standard Test Case

Below is format of a standard login Test case

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1. Go to site <a href="http://demo.guru99.com">http://demo.guru99.com</a> 1. Enter UserId 2. Enter Password 3. Click Submit	Userid = guru99 Password = pass99	User should Login into application	As Expected	Pass
TU02	Check Customer Login with invalid Data	1. Go to site <a href="http://demo.guru99.com">http://demo.guru99.com</a> 1. Enter UserId 2. Enter Password 3. Click Submit	Userid = guru99 Password = glass99	User should not Login into application	As Expected	Pass

Test cases shall include the following components

Test Case Field	Description
Test case ID:	•Each test case should be represented by a unique ID. To indicate test types follow some convention like "TC_UI_1" indicating "User Interface Test Case#1."
Test Priority:	<ul style="list-style-type: none"> <li>•It is useful while executing the test.</li> <li>• Low</li> <li>• Medium</li> <li>• High</li> </ul>
Name of the Module:	•Determine the name of the main module or sub-module being tested
Test Designed by:	•Tester's Name
Date of test designed:	•Date when test was designed
Test Executed by:	•Who executed the test- tester
Date of the Test Execution:	•Date when test needs to be executed
Name or Test Title:	•Title of the test case
Description/Summary of Test:	•Determine the summary or test purpose in brief
Pre-condition:	•Any requirement that needs to be done before execution of this test case. To execute this test case list all pre-conditions
Dependencies:	•Determine any dependencies on test requirements or other test cases
Test Steps:	•Mention all the test steps in detail and write in the order in which it requires to be executed. While writing test steps ensure that you provide as much detail as you can
Test Data:	•Use of test data as an input for the test case. Deliver different data sets with precise values to be used as an input
Expected Results:	•Mention the expected result including error or message that should appear on screen
Post-Condition:	•What would be the state of the system after running the test case?
Actual Result:	•After test execution, actual test result should be filled
Status (Fail/Pass):	•Mark this field as failed, if actual result is not as per the estimated result
Notes:	•If there are some special condition which is left in above field



# Characteristics of Good Test Case

---

## 1. Test Cases need to be simple and transparent

- Create test cases that are as simple as possible. They must be clear and concise as the author of test case may not execute them.
- Use **assertive language** like **go to home page**, **enter data**, **click on this** and so on. This makes the understanding of the test **steps easy** and test **execution faster**.

## 2. Create Test Case with End User in Mind

- Ultimate goal of any software project is to create test cases that meets customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end users' perspective.

# ...Test Case

---

## 3. Avoid test case repetition.

- Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column

## 4. Do not Assume

- Do not assume functionality and features of your software application while preparing test case. [Stick to the specification documents.](#)

# ...Test Case

---

## 5. Ensure 100% Coverage

- Make sure you write test cases to check all software requirements mentioned in the specification document. Use [Traceability Matrix](#) to ensure no functions/conditions is left untested.

## 6. Test Cases must be identifiable.

- Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.

## 7. Implement Testing Techniques

- It's not possible to check every possible condition in your software application. Testing techniques help you select a few test cases with the maximum possibility of finding a defect.
  - Example: BVA, EP, etc.

# ...Test Case

---

## 8. Self cleaning

- The test case you create must return the **Test Environment** to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

## 9. Repeatable and self-standing

- The test case should generate the same results every time no matter who tests it

## 10. Peer Review.

- After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.

# Test Case Management Tools

---

- Test case management tools are the automation tools that help to manage and maintain the Test Cases. Main features of a test case management tool are
  1. **For documenting Test Cases:** With tools you can speed up Test Case creation with use of templates
  2. **Execute the Test Case and Record the results:** Test Case can be executed through the tools and results obtained can be easily recorded.
  3. **Automate the Defect Tracking:** Failed tests are automatically linked to the bug tracker, which in turn can be assigned to the developers and can be tracked by email notifications.
  4. **Traceability:** Requirements, Test cases, Execution of Test cases are all interlinked through the tools, and each case can be traced to each other to check test coverage.
  5. **Protecting Test Cases:** Test cases should be reusable and should be protected from being lost or corrupted due to poor version control.

# Types of test cases

---

There are two types of test cases as mentioned below:

1. **Formal test cases:** Formal test cases are those test cases which are authored as per the test case format. It has all the information like preconditions, input data, output data, post conditions, etc. It has a **defined set of inputs which will provide the expected output.**
2. **Informal test cases:** Informal test cases are authored for such requirements where **the exact input and output are not known.** In order to test them the formal test cases are not authored but the activities will be done and the outcomes are reported once the tests are run.

# Test Suit

---

- In software engineering, a test suite is a collection of test cases that are intended to be used to test a software program, to show that it has some specified set of behaviors.
- Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status.
- A test suit is a collection of test cases that are grouped for test execution purposes.

# Test Script

---

- A test script is a **set of instructions** (written using a scripting/programming language) that is performed on a system under test to verify that the system performs as expected.
- Test scripts are used in **automated testing**.



# Test Data

---

- Test data is the documented data that is basically used to test the software program.
- Test data may be produced by the tester, or by a program or function that aids the tester.  
Test data may be recorded for re-use, or used once and then forgotten.
- Test data is divided into two categories.
  - **Positive test data** which is generally gives to system to generate the expected result and
  - **Negative test data** which is used to test the unhandled conditions, unexpected, exceptional or extreme input conditions.
- *If the test data is inadequately designed, then such test inputs will not cover all possible test scenarios, which impact the quality of the software application under test.*

# ...Test Data

---

- **Limitations**
  - It is not always possible to produce enough data for testing.
  - The amount of data to be tested is determined or limited by considerations such as time, cost and quality.
  - Time to produce, cost to produce and quality of the test data, and efficiency.

# Generating Test Data

---

- 1) **The best test data:** try to create the best data set that should **not be so long** and **identifies bugs** of all kinds of application functions but does **not exceed cost and time limitation** for preparing test data and running tests.
- 2) **Unlawful data set-up:** create wrong data set format. This invalid or dishonest format of data cannot be accepted by system and generates an error message. Check that, it **has to generate an error message**.
- 3) **Boundary condition data set:** data set holding out of range data. Recognize application boundary cases to organize data set that will cover lower as well as upper boundary conditions.

# ...Generating Test Data

---

- 4) **Correct data set:** create **correct data set** to ensure that an application is responding as per the requirement or not and to know that the data is correctly saved in a database/file or not.
- 5) **Incorrect data set:** create **incorrect data set** to confirm application behavior for negative values, alphanumeric string inputs.
- 6) **Create large data set for performance, load and stress testing, and regression testing:** large amount of data set is required for these kinds of testing. To do the performance testing for the DB application that fetches/updates data from/to DB table – the large data set required.

## ... Generating

---

- 7) **Blank or default data:** execute your test cases in blank or default data set environment to check that correct error messages are generated or not.
- 8) **Check the corrupted data:** fill a bug after correct troubleshooting. Before running test case on a particular data, you shall ensure that the data is not corrupted.

# Requirements Traceability Matrix (RTM)

---

- RTM captures all requirements proposed by the client or software development team and their traceability in a single document delivered at the conclusion of the life-cycle.
- It is used to **track the requirements** and to check the current project requirements are met.
- In other words, it is a document that maps and **traces user requirement with test cases**.
- The main purpose of RTM is to see that all test cases are covered so that no functionality should miss while doing software testing.

# Requirement Traceability Matrix

---

- RTM Parameters include
  - Requirement ID
  - Risks
  - Requirement Type and Description
  - Trace to design specification
  - Unit test cases
  - Integration test cases
  - System test cases
  - User acceptance test cases
  - Trace to test script

## Sample traceability matrix

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	REQ1 UC 2.3.1	REQ1 UC 2.3.2	REQ1 UC 2.3.3	REQ1 UC 2.4	REQ1 UC 3.1	REQ1 UC 3.2	REQ1 TECH 1.1	REQ1 TECH 1.2	REQ1 TECH 1.3
Test Cases	321	3	2	3	1	1	1	1	1	1	2	3	1	1	1
Tested Implicitly	77														
1.1.1	1	x													
1.1.2	2		x	x											
1.1.3	2	x											x		
1.1.4	1			x											
1.1.5	2	x												x	
1.1.6	1		x												
1.1.7	1			x											
1.2.1	2				x		x								
1.2.2	2					x		x							



# Types of RTM

---

## 1) Forward traceability

- It maps requirements to test cases.
- This matrix is used to check whether the project progresses in the desired direction and for the right product.
- It makes sure that each requirement is applied to the product and that each requirement is tested thoroughly.

# ...Types of RTM

---

## 2) Backward or reverse traceability

- It maps test cases to requirements.
- It is used to ensure whether the current product remains on the right track.
- The purpose behind this type of traceability is to verify that we are not expanding the scope of the project by adding code, design elements, test or other work that is not specified in the requirements.

# ...Types of RTM

---

## 3) Bi-directional traceability ( Forward+Backward)

- This traceability matrix ensures that all requirements are covered by test cases.
- It analyzes the impact of a change in requirements affected by the defect in a work product and vice versa.

# Advantage of RTM

---

- It confirms 100% test coverage
- It highlights any requirements missing or document inconsistencies
- It shows the overall defects or execution status with a focus on business requirements
- It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases

# Test Execution

---

## PROCESS 3

# Test Execution

---

- Once the preparation of test case development and test environment setup is completed, then test execution phase can be kicked off.
- Test execution is the process of **executing the code** and **comparing the expected** and **actual results**.
- In this phase, testing team start **executing test cases** based on prepared test planning & prepared test cases in the prior step.

# Test execution

---

- Once the test case is passed then same can be marked as passed.
- If any test case is failed then corresponding defect can be reported to the developer team via bug tracking system & bug can be linked for corresponding test case for further analysis.
- Ideally, every failed test case should be associated with at least single bug.

# ...Test Execution

---

- Once the bug fixed by development team then **same test case can be executed** based on your test planning.
- If any of the test cases are blocked due to any defect then such test cases can be marked as **Blocked**, so we can get the report based on how many test cases **passed, failed, blocked** or **not run** etc.
- Once the defects are fixed, same Failed or Blocked test cases can be executed again to **re-test the functionality**.



# ...Test Execution

Entry Criteria	Activity	Exit Criteria	Deliverables
<ul style="list-style-type: none"> <li>• Baselined RTM, Test Plan, Test case/scripts are available</li> <li>• Test environment is ready</li> <li>• Test data set up is done</li> </ul>	<ul style="list-style-type: none"> <li>• Execute tests as per plan</li> <li>• Document test results, and log defects for failed cases</li> <li>• Update test plans/test cases, if necessary</li> <li>• Map defects to test cases in RTM</li> <li>• Retest the defect fixes</li> <li>• Regression Testing of the application</li> <li>• Track the defects to closure</li> </ul>	<ul style="list-style-type: none"> <li>• All tests planned are executed</li> <li>• Defects logged and tracked to closure</li> </ul>	<ul style="list-style-type: none"> <li>• Completed RTM with execution status</li> <li>• Test cases updated with results</li> <li>• Defect reports</li> </ul>

# Test Reporting

---

## PROCESS 4

# Test Reporting and Cycle Closure

---

- Call out the testing team member **meeting** & evaluate cycle completion criteria based on **test coverage, quality, cost, time,** and **critical business objectives**.
- Discuss what all went good, which area needs to be improved & taking the lessons from current STLC as input to upcoming test cycles, which will help to improve bottlenecks in the STLC process.
- **Test cases & bug reports** will analyze to find out the **defect distribution by type** and **severity**.
- Once complete the test cycle then test **closure report** & **test metrics** will be prepared.

# ...Test Reporting and Cycle Closure

Entry Criteria	Activity	Exit Criteria	Deliverables
<ul style="list-style-type: none"> <li>• Testing has been completed</li> <li>• Test results are available</li> <li>• <b>Defect report</b></li> </ul>	<ul style="list-style-type: none"> <li>• Evaluate cycle completion criteria</li> <li>• Prepare test metrics</li> <li>• Document the learning out of the project</li> <li>• Prepare test closure report</li> <li>• Qualitative and quantitative reporting of quality of the work product to the customer.</li> <li>• Test result analysis to find out the defect distribution by type and severity</li> </ul>	<ul style="list-style-type: none"> <li>• Test closure report signed off by clients</li> </ul>	<ul style="list-style-type: none"> <li>• Test closure report</li> <li>• Test metrics</li> </ul>

# Recall

---

A mistake in coding is called **Error**,  
error found by tester is called **Defect**,  
defect accepted by development team then it is  
called **Bug**,  
build does not meet the requirements then it is **Failure**.

# Defect Reporting

---

- When a tester executes the test cases, he might come across the test result which is contradictory to expected result.
- This variation in the test result is referred as a **software defect**.
- While reporting the defect to the developer, your defect report should contain the following attributes:

- **Defect\_ID** - Unique identification number for the defect.
- **Defect Description** - Detailed description of the Defect including information about the module in which Defect was found.
- **Version** - Version of the application in which defect was found.
- **Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.
- **Date Raised** - Date when the defect is raised
- **Reference**- where in you Provide reference to the documents like . requirements, design, architecture or maybe even screenshots of the error to help understand the defect
- **Detected By** - Name/ID of the tester who raised the defect
- **Status** - Status of the defect , more on this later
- **Fixed by** - Name/ID of the developer who fixed it
- **Date Closed** - Date when the defect is closed
- **Severity** which describes the impact of the defect on the application
- **Priority** which is related to defect fixing urgency. Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

I thank you!