# Theory Questions

=======================

# 1.What is Data-Driven Testing ?

Data-Driven Testing is a software testing methodology which will allow testers to

efficiently execute multiple test cases using a single test script.

In this testing approach -

* Test data can be stored in a table or spreadsheet with various input values in the format of .csv,.xls,.xml.
* Test data file also includes other details like expected results which further helps to compare the test execution results automatically and the output results that is pass/fail status can be recorded in the file
* Uses Single Test Script for all the data set with different input values, can also use parameters to pass the values from test data file

With this approach, Testers can test multiple data scenario thus reducing the execution time and provides accurate test results.

**Example:**

If the user wants to login to a system with multiple input fields with 100 different data sets.

**Approach 1)** Create 100 scripts one for each dataset and runs each test separately one by one.

**Approach 2)** Manually change the value in the test script and run it several times.

**Approach 3)** Import the data from the excel sheet. Fetch test data from excel rows one by one and execute the script.

# 2.Explain the Steps for Bug Cycle ?

The Bug Life Cycle (also known as the Defect Life Cycle) in software testing involves a series of states that a defect or bug goes through from its initial stage to final stage. Below are the different stages.

1. **New**:

If the code is broken or any functionality got disturbed, a new defect is logged and status will be assigned as **New.**

1. **Assigned**:

The bug is posted by the tester and the lead approves the bug and assigns it to the developer team.

1. **Open**:

The developer starts analysing and works on fixing the defect.

1. **Fixed**:

After making necessary code changes, the developer verifies the fix and the bug status is updated to **Fixed**.

1. **Pending Retest**:

Once the developer fixes the bug and the status will be updated to –**Pending Retest** .

1. **Retest**:

Once the responsible tester starts working on the revised code then the status changes to **Re-test**.

1. **Verified**:

If no bug is detected during retesting, the bug is considered **fixed** and the status becomes **Verified**.

1. **Reopen**:

If the bug persists even after the fix, the tester changes the status to **“Reopened”** and the bug goes through the life cycle again.

1. **Closed**:

If the bug no longer exists, the status is set to **Closed**.

1. **Duplicate**:

If the defect is repeated or corresponds to the same concept, it is marked as **Duplicate**.

1. **Rejected**:

If the developer believes the defect is not genuine, it is labelled as **Rejected**.

1. **Deferred**:

Bugs of lower priority, expected to be fixed in the next release, are marked as **Deferred**.

1. **Not a Bug**:

If the defect does not affect application functionality, it is categorized as **Not a bug**.

# 3.Mention the different types of Software Testing ?

Software testing is the process of checking the Quality, functionality, and performance of a software product before launching to Users.

There are mainly 2 types of testing -Functional and Non Functional Testing.

## **Functional Testing**

Functional testing focuses on verifying that individual components or units of software work correctly. Here are the main types within functional testing:

1. **Unit Testing**:

Conducted at the **lowest level**, unit testing checks individual units (such as methods, functions, procedures, or objects) to ensure their correctness.

Example:

Imagine a simple calculator application. A unit test could verify if users can

enter two numbers and get the correct sum for addition functionality.

* **White Box Testing**:

This technique exposes the internal structure or code of an application to the tester.It helps find design loopholes or faults in business logic.

Examples include **statement coverage** and **decision coverage/branch coverage.**

* **Gorilla Testing**:

Thoroughly tests a specific module of an application. Useful for assessing the robustness of your application. Focus

thoroughly and test it with positive and negative scenarios.

1. **Integration Testing**:

Involves logically grouping two or more modules and testing them as a whole. Ensures that integrated components work seamlessly together.

1. **System Testing**:

Validates the entire system’s functionality.

Tests the system as a whole, including interactions between different modules.

1. **Acceptance Testing**:

Performed to verify if the system meets user requirements

Includes **customer-based testing** to ensure alignment with user expectations.

## **Non-Functional Testing**

Non-functional testing assesses aspects beyond functionality. Here are some key types:

1. **Security Testing**:

Evaluates the system’s security features and identifies vulnerabilities and ensures data protection.

1. **Performance Testing**:

Measures system performance under various conditions (e.g., load, stress, scalability) and also determines response times, resource usage, and bottlenecks.

1. **Usability Testing**:

Assesses the user-friendliness of the software and focuses on user experience and ease of interaction.

1. **Compatibility Testing**:

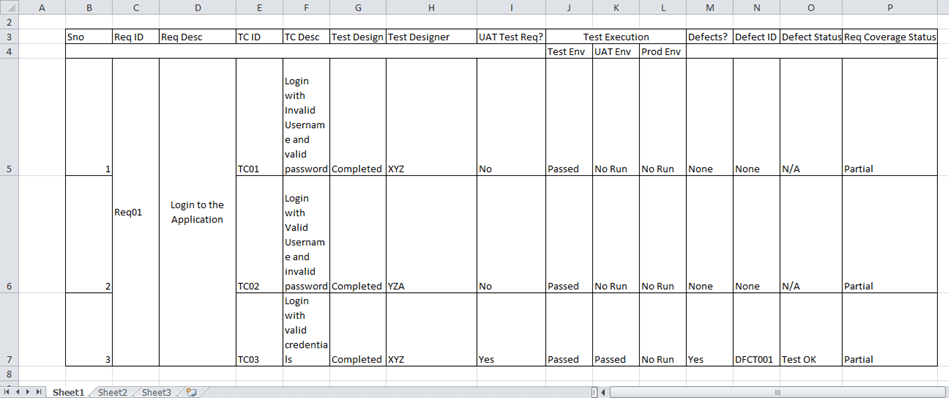
Checks if the software works across different platforms, browsers, and devices.

# 4. What is Traceability matrix and what information does it contains?

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.It is used to track the requirements and to check the current project requirements are met.

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the [Software development life cycle](https://www.guru99.com/software-development-life-cycle-tutorial.html). The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

Below is the sample requirement traceability matrix in [software testing](https://www.guru99.com/software-testing.html) project.



# What are the automation challenges that testing team faces while testing?

Below are the few challenges faced by testing team during the time of testing

* Sometimes it will be difficult to find the right testing framework or tool that suits the project requirements and budget.
* Effective communication and collaboration in the team to ensure consistent and quality testing.
* Identifying the test automation strategy and prioritizing the test cases that need to be automated
* High initial investment in terms of time, money, and resources.
* Inadequate testing infrastructure and environment
* Problems with data reliance and data quality
* Finding the right skills and expertise for automation testing
* Scalability, UI, and API challenges
* Maintenance and updating of test scripts and tools