**Test Driven Development**

* Test Driven Development is the process in which test cases are written before the code that validates those cases. It depends on repetition of a very short development cycle.
* Three activities are tightly interwoven: coding, testing (in the form of writing unit tests), and design (in the form of refactoring).
* Steps followed
* Add tests – Write a test case that describe the function.
* Run all the test cases and make sure/(it will fail) that the new test case fails.
* Write the code that passes the test case.
* Run the test cases.
* Refactor code – This is done to remove duplication of code.
* Repeat the above steps again and again.
* Advantages
* Unit test provides constant feedback about the functions.
* Quality of design increases which further helps in proper maintenance.
* Test driven development act as a safety net against the bugs.
* TDD ensures that your application actually meets requirements defined for it.
* TDD have very short development lifecycle.
* Disadvantages
* writing too many tests at once
* writing tests that are too large or coarse-grained
* writing overly trivial tests, for instance omitting assertions
* writing tests for trivial code, for instance, accessors

**Behavioural Driven Development (BDD)**

* BDD refers to an Agile software development process that is derived from the TDD (Test Driven Development) methodology.
* Behavioural Driven Development is a good approach in Automated Testing as it is more focused on the behaviour of the system rather than the implementation of the code.
* BDD is considered as a test to illustrate the behaviour of the system. It encourages the use of conversation and concrete examples in simple language for everyone involved like a customer, developer, tester, and stakeholder in the development to bring better clarity to the behaviour of the system.
* BDD cycle or steps
* Describe behaviour – This includes the flow and features of the product means the main vision.
* Define requirements – Modelled requirements with business rules for a shared understanding.
* Run and fail the tests – Develop and run the test cases.
* Apply code update – Refactor it according to the requirement.
* Run and pass the tests – Run the updated code and pass the test cases.
* Advantages
* Greater clarity on business goals and customer requirements.
* Reaches a larger customer set as it uses non-technical languages.
* Helps in avoiding unnecessary features and includes important features.
* Reduces effort for post-modification and post-deployment defects.
* Integration with Continuous Integration and Delivery processes can automate build, testing, and deployment of software, resulting in faster feedback loops and delivery of features.
* Detailed documentation of user stories, acceptance criteria, and step definitions can improve overall project documentation, facilitating maintenance and updates to the system.
* BDD prioritizes features based on their importance to the customer, ensuring that the development team is working on features that deliver real value to the business.
* Disadvantages of BDD
* BDD can be time-consuming to create and maintain extensive documentation, including user stories, acceptance criteria, and step definitions.
* Involving all stakeholders, especially non-technical stakeholders, can be a challenge.
* BDD tests heavily rely on the quality of initial requirements and user stories; incomplete or inaccurate requirements can lead to insufficient test coverage and bugs.
* BDD may not be suitable for short development cycles or projects with frequently changing requirements.

**Data Driven Testing**

* Data-driven testing is creation of test scripts where test data and/or output values are read from data files instead of using the same hard-coded values each time the test runs.
* Or an approach to the architecture of automated tests by creating test scripts and reading data from data files.
* testers can test how the application handles various inputs effectively. It can be any of the below data files.
* Datapools
* Excel files
* ADO objects
* CSV files
* ODBC sources
* Advantages
* Data-driven testing allows the reusability of code.
* It improves test coverage.
* In the case of regression testing, the test application allows the set of data values.
* Changes in the test script do not affect the test data as well as the test process.
* Different tools generate the test data automatically and a large volume of test data is taken to save time when necessary.
* Less maintenance is required as well as it provides flexibility in application maintenance.
* Several functions and actions can be reused in many test cases present in data-driven testing.
* It helps in reducing redundancy and unnecessary duplication of test scripts.
* Disadvantages
* One of the biggest drawbacks is that the quality of the test depended upon the automation team skills i.e., being implemented.
* Data-driven testing requires great knowledge and expertise in the scripting language.
* When the amount of data is more for validation it takes so much time to execute.
* In this type, maintenance plays a big issue in the code complexity and difficulty of understanding the logic.
* More documentation and high-level technical skills are required. Another thing is that the tester should learn the entire new scripting language.

**Gherkin**

* Gherkin is a business readable language which helps you to describe business behaviour without going into details of implementation.
* It is a domain specific language for defining tests in Cucumber format for specifications. It uses plain language to describe use cases and allows users to remove logic details from behaviour tests.
* The text in Gherkin language acts as documentation and skeleton of your automated tests. Gherkin format is based on TreeTop Grammar which exists in 37+ languages. Therefore, you can write your gherkin in 37+ spoken languages.
* Gherkin offers a common set of keywords in the plain English text, which can be used by members from different communities and can get the same output from the test scripts. Gherkin offers the following specific keywords to write the common test scripts in the feature file
* Feature
* Scenario
* Given
* When
* Then
* But
* And
* Background

**Automation framework**

* It is a set of guidelines like coding standards, test-data handling, object repository treatment and corresponding tools that are used for building test cases. It is designed to help engineering functions work more efficiently.
* These are just guidelines and not rules; they are not mandatory and you can still script without following the guidelines.
* Types of Test Automation Frameworks
* Below are the different types of Automated Testing Frameworks:
* Linear Scripting
* The Test Library Architecture Framework.
* The Data-Driven Testing Framework.
* The Keyword-Driven or Table-Driven Testing Framework.
* The Hybrid Test Automation Framework.

**Unit Testing**

* Unit Testing is a part of software testing where testing is performed on isolated small parts of the code. The objective of Unit Testing is to isolate small portions of application code that could be individually tested and verified.
* It is performed by the development Team in the SDLC process.
* Following are some of the Unit testing tools
* JUnit
* NUnit
* TestNG
* PHPUnit
* Mockito
* Cantata
* Karma
* Mocha
* TypeMock.
* Jasmine

**User Acceptance Testing**

* It is a phase of a web project life cycle where your ideal customers test your app or website in a production-like environment.
* Following are some of the user acceptance testing tools
* Marker.io
* UserBrain
* FullStory
* Amplitude
* Sentry
* Contentsquare
* Maze
* UserReport
* Qualaroo
* UserTesting

**API Testing**

* API is the Application Programming Interface. It acts as a middle layer between the User Interface and the Database. API communicates and exchanges data from one software system to another.
* API testing is a software testing practice that tests API for functionality, performance, and security. API testing helps to test the application logic.
* It consumes less time and is more effective compared to UI testing.
* Following are some of the API testing tools
* Testsigma
* Postman
* SoapUI
* REST-assured
* Assertible
* JMeter
* Karate DSL
* Apache JMeter.

**Software Testing Life Cycle (STLC)**

* The Software Testing Life Cycle (STLC) is a systematic approach to testing a software application to ensure that it meets the requirements and is free of defects.
* It is a process that follows a series of steps or phases, and each phase has specific objectives and deliverables.
* The STLC is used to ensure that the software is of high quality, reliable, and meets the needs of the end-users.
* The main goal of the STLC is to identify and document any defects or issues in the software application as early as possible in the development process. This allows for issues to be addressed and resolved before the software is released to the public.
* Phases of STLC

1. Test Strategy
2. Test Plan
3. Test Design
4. Test Environment – Environment/System setups.
5. Test Execution – Running the tests.
6. Defect Reporting – Finding, Logging, Fixing, and Resting Bugs.
7. Test Closure – Official closing of test and end of testing.

* Test Strategy
* It is an official documentation that is made by the testing team for the product during the start.
* This document discusses on how and with aspects testing is going to be implemented.
* Only one test strategy is made for the entire duration of the project, i.e., almost no alteration will be allowed.
* It has a template of ten steps

1. Objective – Sole purpose or aim of making this strategy.
2. Approach – What do we do regarding the objective.
3. Scope – In Scope and Out Scope.
4. Testing Methodologies – Type of test going to be implemented.
5. Tools and Technologies – Tools going to be used, both manual and automation related, management tools, external apps.
6. Platform – Testing environment, OS, Device details, App details.
7. Testing Team – Persons involved.
8. Test Deliverables – Test strategy, test plan, test documents, reports.
9. Summary – Brief summarization (7-8 sentences) about above points.
10. Approving Authority – PO/PM, Test lead, Head of Engineering, CTO, all senior people.

* Test Plan
* Documentation made for short term goals and smaller breakdowns of the strategy.
* Made by testing team, time to time.
* Elaborate on How and the action points to carry out the test strategy to achieve its goals.
* Can be multiple documents.
* Template or steps involved are same as Test strategy but always **with respect to duration.**
* Test teams includes the responsibilities, only has InScope, wrt duration.
* Test Design
* Deriving test cases and designing tests.
* Techniques

1. Black Box Testing – Done without knowing internal working.

* **Equivalence partitioning** – is also known as Equivalence Class Partitioning or ECP for short. It's a testing technique that divides the input data into different partitions or classes, where each partition is expected to exhibit similar behaviour. This helps in reducing the number of test cases while ensuring that representative test cases are selected from each partition to verify the software's functionality.
* **Boundary Value Analysis** – is a software testing technique used to design test cases around the boundaries of input values. It is particularly useful for identifying defects related to boundary conditions, as these are often where software tends to behave differently or encounter errors. BVA focuses on testing values that are on or near the edges of an input domain, as these are more likely to cause issues.
* **Decision Table** – A decision table is a structured representation of different combinations of conditions and corresponding actions or outcomes.

1. White Box Testing – Testing the internal and infrastructure of the codes.
2. Exploratory Testing – Simultaneous learning, test design and execution.

**BY**

**Hariharamanikandan K**