***Equivalence partition and boundary value analysis***

**Black box testing**

Tester is unaware of the internal workings (end user’s perspective)

***Equivalence partitioning***

Focus on eliminating sets of input data that result in the same system behaviour and produce identical outcomes during app testing

***Boundary Value Analysis***

Test boundaries of input values, complements equivalence partitioning.

***Decision table and state transition testing***

***Decision table***

Assist testers in understanding the effects of combinations of different inputs and other software states that must correctly implement the business roles

***State transition testing***

Aims to ensure that the system transitions between different states correctly and consistently

* All states coverage
* 0 switch coverage/ valid transitions coverage

***Use Case testing***

Helps software testers write better test cases, it is user-oriented and explains real-life user scenarios

* Actors
* Steps
* Description
* Extension

***Negative testing***

Software testing is often seen as making sure that applications work as they should, where certain actions lead to expected results.

Only considering positive scenarios wouldn’t be enough

Some applications and web pages have fields that are marked as mandatory. To test the behaviour of such fields, we can create tests that leave the required fields empty and analyse the applications response.

***Positive testing*** verifies that your application works as expected “Happy path testing”

**Smoke, sanity and regression testing**

**Smoke testing**

Is an approach that is usually carried out during the initial development stages of the software development lifecycle. Its purpose is to make sure that the core functionalities of the program are working fine without any issues

Use:

* Developers provide a fresh build to the Quality Assurance team
* New module is added to the existing functionality

**Sanity testing**

Performed to check whether the software product is working correctly. It does a quick evaluation of the quality, release to determine whether it is eligible for further rounds of testing or not.

Use:

1. When the build is received after many regressions or if there is a minor change in the code
2. The build is received after a bug fix
3. Before deployment on production

***Regression testing***

Process of verifying that the software still performs correctly after changes were introduced.

Once sanity testing of the changed functional is completed, all the impacted features of the application require complete testing

Areas to test during regression testing:

* Areas are exposed to a high number of users
* Areas that have frequent defects
* Core features
* High complex functionality