**Software testing classification:** software testing is broadly classified into two categories of how and when testing is done.

1. **Based on execution:**

**Static testing:**  this testing is done before the execution of the code; it involves reviewing the documents and articrafts to find potential defects early. The static techniques include reviews (informal, walkthrough, inspection)

**Dynamic testing:** it is done by executing the software to check the behavior during runtime

1. **Based on testers knowledge:**
2. **Whitebox testing:** in this testing the tester has full knowledge about the internal architecture, code and algorithms (testing some small sort of code such as if-else statements)
3. **Blackbox testing:** in this testing the tester has zero knowledge about the internal architecture, code (testing the login page without having the knowledge about the backend)
4. **Gray box testing:** in this testing the tester has limited knowledge about the s/w (API testing)

**3.Based on Levels of testing:**

**Unit testing:** in this testing the testing is done on individual component, it focuses on smaller unit of the s/w design. This is usually done by developers by providing a sample input and checking the output.

**Integration testing:** in this testing process the testing is performed by joining the modules into the group. The main purpose of the integration testing is to check whether the modules are communicating with each other properly. This testing is also done by the developer.

**System testing:** in system testing the testing is done on whole the software application. It is usually done by the tester.

**User acceptance testing:** a level of software testing in which testing is done for user acceptance.

**4.Based on objective:**

**Functional testing:** it is the type of testing in which the system is tested against the functional requirements and specifications

**Non functional testing:** it is the type of testing in which the system is tested against the non-functional requirements and specifications.

The functional testing is further divided into:

* Unit testing
* Integration testing
* System testing
* End to end testing
* Acceptance testing

The non-functional testing is divided into:

* Performance testing (load, stress, scalability, stability)
* Usability testing
* Compatibility testing

**Regression testing:** it is a type of testing that ensures the code changes doesn’t negatively affect the existing features of the s/w

**Sanity testing:** Sanitytestingensures that the specific functionality of the s/w works properly

**Smoke testing:** this ensures that the basic functionalities of the s/w works properly (search bar, landing homepage)

The above regression, sanity, smoke testing also comes under the types of testing “**based on objective”**

**Test case design techniques or black box design techniques:**

The test case design techniques are classified as:

* **Error guessing:** in this design the test engineer will think of all the negative possibilities of the system

**Ex:** assume that the test field requirement is to enter only the positive integers, then the test engineer will check for all the negative possibilities such as by entering alphanumeric, negative numbers, alphabets as the input

* **Equivalence** **partitioning**: this design is also called as equivalence **class** partitioning(**ECP**). In this design the input is divided into sub classes and from each class a random input is selected and verified for each class

**Ex**: assume you have a requirement to enter the values only between 1 to 500. In this case instead of entering the inputs manually from 1-500 since this is a time consuming process the 500 inputs and divided into groups called classes such as (-100 to 0 , 1-100 , 101-200….so on) and from this classes one input is selected and if that test case pass the whole class is present and there is no need to check the other value from the class. In this way the ECP saves the tester time

* **Boundary value analysis:** as the name suggests it the testing at the boundaries between the partitions.

**Ex :** assume you have two numbers namely **a** and **b ,** in this case the test cases for **a** would be (**a-1 , a , a+1**) and for **b** would be (**b-1 , b , b+1**)

* **Decision table technique:** it is technique in which the test cases are designed based on condition, combination and rule criteria

**Ex:** assume the following:

**New\_customer = 15%**

**Repeating\_customer = 10%**

**Coupon\_code = 20%**

**Therefore,** the no. of test cases = no. of rules = 2^ no. of conditions

**For above, = 2^3 = 8**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Conditions/Rules | Rule1 | Rule2 | Rule3 | Rule4 | Rule5 | Rule6 | Rule7 | Rule8 |
| **15%** | T | T | T | T | F | F | F | F |
| **10%** | T | T | F | F | T | T | F | F |
| **20%** | T | F | T | F | T | F | T | F |
| **Result** | x | x | 35% | 15% | 30% | 10% | x | x |

* **State transition technique:** it is technique in which the systems behavior depends on the current state. It tests the transitions between the different stages triggered by the various inputs. As the name says transitions from one stage to another

**Ex:** food delivery app and if user enters the login credentials wrong for more than three times then the user should be blocked and doesn’t allow him to order the food.