Testing Techniques

1] Equivalence Partitioning

- In this method, the input domain data is divided into different equivalence data classes. This method is typically used to reduce the total number of test cases to a finite set of testable test cases, still covering maximum requirements.
- In short, it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing.
- For example, if you are testing for an input box accepting numbers from 1 to 1000 then there is no use in writing thousands of test cases for all 1000 valid input numbers plus other test cases for invalid data.
- Using the Equivalence Partitioning method above, test cases can be divided into three sets of input data called classes. Each test case is a representative of the respective class.
- So in the above example, we can divide our test cases into three equivalence classes of some valid and invalid inputs.

Test cases for input boxes accepting numbers between 1 and 1000 using Equivalence Partitioning:

1) One input data class with all valid inputs. Pick a single value from the range of 1 to 1000 as a valid test case. If you select other values between 1 and 1000, then the result is going to be the same. So one test case for valid input data should be sufficient.

2) Input data class with all values below the lower limit. i.e., any value below 1, as an invalid input data test case.

#3) Input data with any value greater than 1000 to represent the third invalid input class.

- So using Equivalence Partitioning, you have categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.
- We have selected one representative from each input class to design our test cases. Test case values are selected in such a way that the largest number of attributes of equivalence class can be exercised.
- Equivalence Partitioning uses the fewest test cases to cover the maximum requirements.

2]Boundary Value Analysis

- It's widely recognized that the input values at the extreme ends of the input domain cause more errors in the system.
- More application errors occur at the boundaries of the input domain. 'Boundary Value Analysis' Testing technique is used to identify errors at boundaries rather than finding those that exist in the center of the input domain.
- Boundary Value Analysis is the next part of Equivalence Partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:

#1) Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000 in our case.

#2) Test data with values just below the extreme edges of input domains i.e. values 0 and 999.

#3) Test data with values just above the extreme edges of the input domain i.e. values 2 and 1001.

Boundary Value Analysis is often referred to as part of Stress and Negative Testing.

- Note: There is no hard-and-fast rule to test only one value from each equivalence class you created for input domains. You can select multiple valid and invalid values from each equivalence class according to your needs and previous judgments.
- For example, if you divide 1 to 1000 input values invalid data equivalence class, then you can select test case values like 1, 11, 100, 950, etc. Same case for other test cases having invalid data classes.
- This should be a very basic and simple example to understand the Boundary Value Analysis and Equivalence Partitioning concept.

3]Error Guessing

- Error Guessing is a software testing technique based on guessing the error which can prevail in the code. The technique is heavily based on the experience where the test analysts use their experience to guess the problematic part of the testing application. Hence, the test analysts must be skilled and experienced for better error guessing.

- The technique counts a list of possible errors or error-prone situations. Then tester writes a test case to expose those errors. To design test cases based on this software testing technique, the analyst can use the past experiences to identify the conditions.

* Guidelines for Error Guessing:

- The test should use the previous experience of testing similar applications
- Understanding of the system under test
- Knowledge of typical implementation errors
- Remember previously troubled areas
- Evaluate Historical data & Test results

4]Decision Table Based Testing.

- A decision table is also known as to Cause-Effect table. This software testing technique is used for functions which respond to a combination of inputs or events. For example, a submit button should be enabled if the user has entered all required fields.
- The first task is to identify functionalities where the output depends on a combination of inputs. If there are large input set of combinations, then divide it into smaller subsets which are helpful for managing a decision table.
- For every function, you need to create a table and list down all types of combinations of inputs and its respective outputs. This helps to identify a condition that is overlooked by the tester.

*** Following are steps to create a decision table:**

- Enlist the inputs in rows
- Enter all the rules in the column
- Fill the table with the different combination of inputs
- In the last row, note down the output against the input combination.
- Example: A submit button in a contact form is enabled only when all the inputs are entered by the end user.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Input								
Name	F	T	F	T	F	T	F	T
Email	F	F	T	T	F	F	T	T
Message	F	F	F	F	T	T	T	T
Output								
Submit	F	F	F	F	F	F	F	T

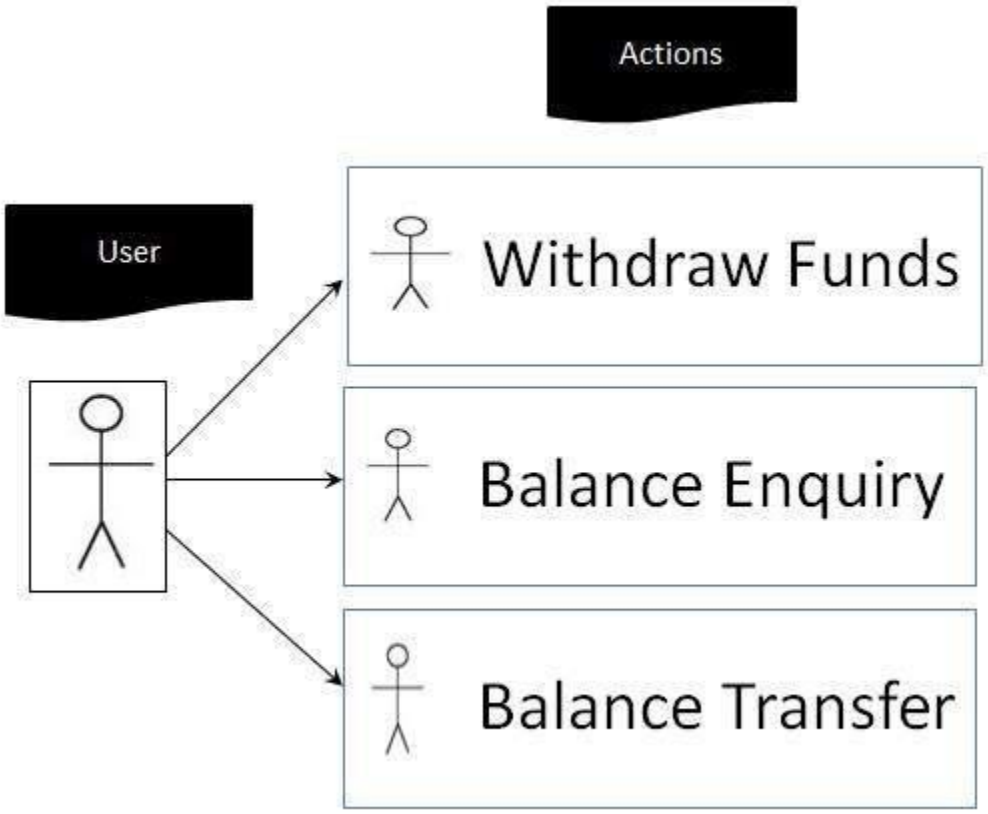
5]Use Case Testing

Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.

Characteristics of Use Case Testing:

- Use Cases capture the interactions between 'actors' and the 'system'.
- 'Actors' represents user and their interactions that each user takes part into.
- Test cases based on use cases and are referred as scenarios.
- Capability to identify gaps in the system which would not be found by testing individual components in isolation.
- Very effective in defining the scope of acceptance tests.

Example:The Below example clearly shows the interaction between users and possible actions.


Actions
User
Withdraw Funds
Balance Enquiry
Balance Transfer