DIFFERENT TECHNIQUES

1. **BOUNDARY VALUE ANALYSES**

* It is a technique used to validate the input data with test box in AUT. [application under test]
* It is black box testing. [tester does not have access the source code of software]
* A boundary value for a valid partition is a valid boundary value.
* A boundary value for an invalid partition is an invalid boundary value.
* For each variable we check-
  + Minimum value.
  + Just above the minimum.
  + Nominal Value.
  + Just below Max value.
  + Max value.
* **Example**

Consider a system that accepts PHONE NUMBER from 5 to 10 numbers.

| **Boundary Value Analysis(phone number 5 to 10)** | | |
| --- | --- | --- |
| Invalid  (min-1) | Valid  (min, min + 1, nominal, max – 1, max) | Invalid  (max + 1) |
| 4 | 5,6,7,8,9,10 | 11 |

**Valid Test cases:**Valid test cases for the above can be any value entered greater than 5 and less than 10.

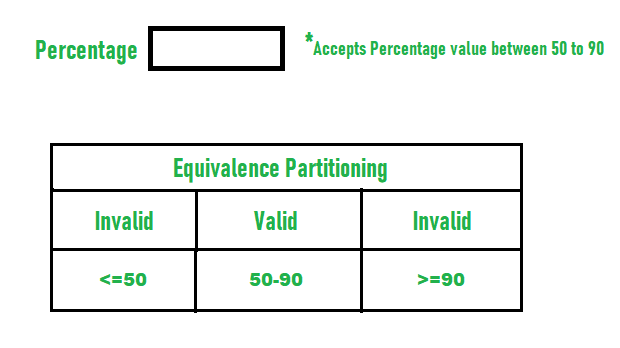
* Enter the value- 5
* Enter the value- 6.
* Enter the value- 7
* Enter the value- 8
* Enter the value- 9

**Invalid Test cases:**When any value less than 18 and greater than 56 is entered.

* Enter the value- 11
* Enter the value- 4

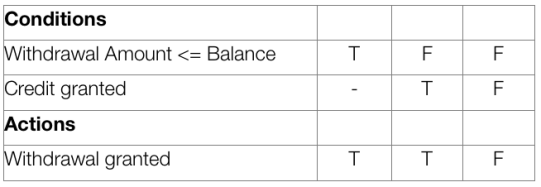
1. **EQUIVALENCE PARTITIONING**

* It is a technique used to validate and partition test data to reduce the test cases in AUT.
* It is black box testing.
* **Example**
* College admission process.
* There is a college that gives admissions to students based upon their percentage.
* Consider percentage field that will accept percentage only between 50 to 90 %, more and even less than not be accepted, and application will redirect user to an error page.
* If percentage entered by user is less than 50 %or more than 90 %, that equivalence partitioning method will show an invalid percentage.
* If percentage entered is between 50 to 90 %, then equivalence partitioning method will show valid percentage.



**.3. DECISION TABLE**

* It is a technique used to go with conditions and rules check the functionality.
* **Example:**

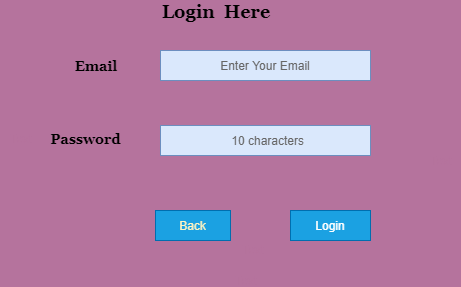


* [Write test cases](https://reqtest.com/testing-blog/learn-how-to-write-effective-test-cases/) based on the table.
  1. Test case for R1: balance = 200, requested withdrawal = 200. Expected result: withdrawal granted.
  2. Test case for R2: balance = 100, requested withdrawal = 200, credit granted. Expected result: withdrawal granted.
  3. Test case for R3: balance = 100, requested withdrawal = 200, no credit. Expected Result: withdrawal denied.

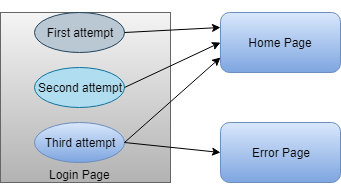
**4.STATE TRANSITION**

* It is a technique used to go with input conditions in AUT.

**Example**



* There is a login function of an application which provides a maximum three number of attempts, and after exceeding three attempts, it will be directed to an error page.



* S1 denotes first login attempt.
* When the first attempt is invalid,
* The user will be directed to the second attempt (state S2).
* If the second attempt is also invalid, then the user will be directed to the third attempt (state S3).
* Now if the third and last attempt is invalid, then the user will be directed to the error page (state S5).

|  |  |  |  |
| --- | --- | --- | --- |
| **STATE** | **LOGIN** | **VALIDATION** | **REDIRECTED** |
| S1 | First Attempt | Invalid | S2 |
| S2 | Second Attempt | Invalid | S3 |
| S3 | Third Attempt | Invalid | S5 |
| S4 | Home Page |  |  |
| S5 | Error Page |  |  |

* Otherwise if the third attempt is valid, then it will be directed to the homepage (state S4).

|  |
| --- |
|  |
| First Attempt | Invalid | S2 |
| S2 | Second Attempt | Invalid | S3 |
| S3 | Third Attempt | Valid | S4 |
| S4 | Home Page |  |  |
| S5 | Error Page |  |  |
|  |  |  |  |

**5. ERROR GUESSING**

\* It is a technique based on tester or dev experience.

\* **Example**

 we have one bank account, and we have to deposit some money over there, but the amount will be accepted on a particular range of **which is 5000-7000**.

So here, we will provide the different input's value until it covers the maximum test coverage based on the error guessing technique, and see whether it is accepted or give the error message:

|  |  |
| --- | --- |
| **value** | **Description** |
| 6000 | Accept |
| 5555 | Accept |
| 4000 | Error message |
| 8000 | Error message |
| blank | Error message |

**Condition: if amount >5000 and amount<7000 amount**

And, if we enter 5000 → error message (not accepted based on the condition)

7000→ error message (not accepted based on the condition)