SIT Environment

- What is XYZ testing? Why we are doing these testing? Are we/you preparing any documents? (TCD, Defect, TCE, etc.), after these testing any documentation send /email send /inform to Test Lead, PM & BA, etc.?
- When developer will sent the build→ Inform to Tester throw Mail (JIRA) & attached in mail **Unit Testing** document (Step for testing feature+ Tables name)
- Sent the build/ Deployment process (Code → Push to GIT master Brach → Jenkins job → Dev to SIT environment)
- In SIT environment, Tester is working
- Tester will do TCD, Review (peer review), TCE, defect raised/inform to developer, demo etc.
- SIT environment different types of testing
  1. Sanity testing/Smoke testing
  2. System & functionality testing
  3. Re-testing & regression testing, etc.
- SIT Environments- URL - https://qa.Paytm.com

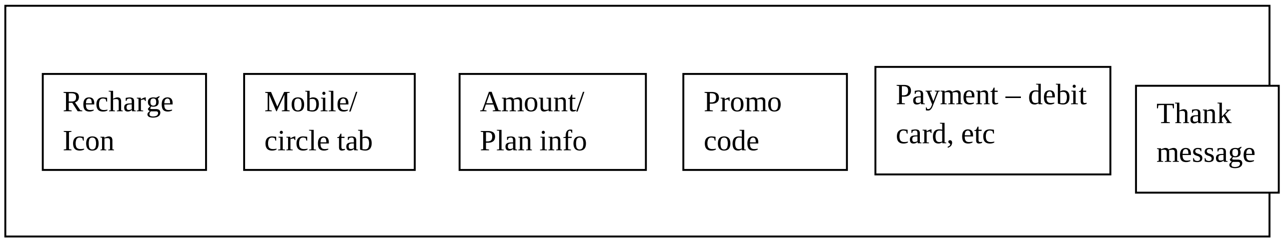
Sanity Testing

- Sanity testing- it is **first testing** in SIT environment
- In sanity testing, **tester** are involved
- When tester will performed **first types of testing in SIT environment**, these testing are called **level zero testing / zero level testing**
- Sanity testing also called **Tester acceptance testing/ Build verification testing**
- When tester/we will got **New build** from developer then tester will **check the build stability** i.e. **either build is stable for testing** these type of testing is called sanity testing
- Main **agenda** is to check **basic & core functionality** of the application/ **main flow** of application/ **happy flow** of application & application working flow
- In Sanity testing we will performed or Stability of application in sanity testing check/validate

1. Validating the Core functionality of application/ feature- ex. Paytm-Recharge-Recharge for every mobile **or** Basic & core functionality validation
2. Validating the GUI/ UI of application/ feature **or** GUI validation
3. Validating link present in application/ feature **or** Link Validation
4. Validating the tab/pages present in application/ feature **or** Tab Validation
5. Validating the Navigation **or** Page validation

1. Basic & Core Functionality validation

- We check **main flow** of application from start to end or **page to page** i.e. we check **happy flow**
- We check for **blocker/show stopper**, if we **found**, we raise that defect/we lock that defect & **assign to the developer**
- If we found defects, so according to the small, large, critical we note down those defect
- If we found blocker, we **reproduce it 2-3 time** before the raising the defect
- **Ex. Paytm-Recharge-Recharge for every mobile-** in this test, tester validate user can proceed for next stage
- Recharge Module-



3. Tab Validation

- We check functionality of tab, where we enter **characters, special characters, numbers, symbols** & we check whether this **text box is accepting it or not**
- Whenever we enter any value in tab by using on **screen keyboard or physical keyboard,** those characters, special characters, numbers, symbols should get entered in tab

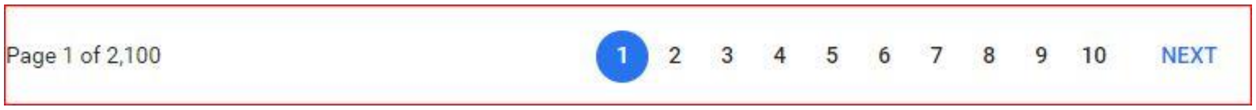
4. Link Validation

- In this validation, **sequence of interlink pages** are tested

- Ex. if I click on **flight**, then **flight information page should open**, so, developer should provide link of that page to the icon

5. Page validation

- Page validation means **navigation** validation
- We check, can we navigate from **one page to other page**
- We click on **next or back arrow**, so, pages should **navigate front & back**
- We check, **pagination** (for web based, whether it is navigating on that particular selected page or not)



6. Graphical User Interface/UI Validation

- This testing test the interface with which **user interact directly**
- In this test, **tester check**

1. Display of application
2. Logo & Images- Whether is it clear or blur
3. Alignment
4. Is UI as per wireframe functionality
5. Dropdown behavior
6. Resolution of logo etc.

- This **validation of visualization** is called GUI validation

- When we performed these sanity testing, if we **found blocker/show stopper** defects in the application/feature, then tester will **reject the build**
- When we performed these sanity testing, if we found **defects (buggy build)** then we simply **reject the build** (ex. we found more than 25 to 30 defects)
- After sanity testing we **decide** whether **build is stable or unstable**
- If we will **reject the build** then we will inform to the **developer throw mail (Outlook Mail)** (JIRA/HPALM)

- Then developer will **fix** the issue/defects & **sent** us a **New build**, then tester will perform **again sanity testing**
- **Ex.** Paytm - Recharge module- US- Browser plane information → Developer will coding/ preparing the **new build(V.9.0)** → Developer will sent for Testing → Tester will do **Sanity Testing (check build stability)** → If core functionality not working → **If we found a defect** → Tester direly **reject the build (V9.0)** → Inform to **developers by sent a mail &** Developer will fix the defect & **prepared a new build (V9.1)** → **New build (V9.1) will sent for testing** → Tester will do again **sanity testing New build (V9.1)**
- In sanity testing, we found issue/ defects → **Core functionality is not working, System hang out problem, Run time problem, Pop/link is not working, Environment problem, etc.**
- In sanity testing, Tester will required only **2 hr to 4 hr** for the Testing
- In sanity testing, Tester are **not writing the Test cases**

Note

- When we decide build is unstable then, we send mail to the development lead, test lead (CC of mail), developer, Product owner (CC of mail)
  & scrum master (CC of mail)

  Mail

  1. Sanity has been performed successfully & these are my observations
  2. During sanity, I have found this much defects
  3. I have raised that defect & I have assigned it to the developer also
     -Defect ID:-
     -Comment:-
     Expected behavior:-
     Actual got behavior:-
     -Screenshots
- On this report, actually decide, next testing has to continue or does not
- If we discard the build, then DL, TL & sanity tester & developer sit together & discuss the issues / defects / show stoppers/ blocker

Smoke Testing

- Smoke Testing, it is **advance version of sanity testing**
- In sanity testing, if we **found a defects** then tester will **reject build**
- In smoke testing- if developer will sent us a **new build** then test the build for testing. If we **found a defects** then we will **reject the build** but we will **provide the root cause of the defect.**
- Then developer will fix the issue/defects & sent us a new build, then tester will perform again smoke testing
- **In smokes testing** = **Sanity Testing + Troubleshooting/ root cause -** Tester
- **In smokes testing** = **Sanity Testing + Package validation –** Developer
- **Troubleshooting-** nothing but finds the exact root cause of defect
- **Package validation-** Package is collection of object
- In smoke testing **both Tester (Troubleshooting/ root cause)& developer are working (Package validation)**
- We (Tester) conduct session with the backend developer/developer, then we check package for which parameter is not passing there etc. we get all the response in console & we get root cause of defect.
- In smoke testing, also We will inform to **developers by sent a mail** (Outlook Mail) (JIRA/HPALM) **and with defects root cause**
- In smoke testing, **Tester can't write test cases** for smoke testing
- In smoke testing, Tester will required **only 2 hr 4 hr** for the Testing
- **In my project,** we are performing **Smoke testing**, whenever we get new build

❖ **Difference between Sanity & Smoke testing**

Sanity Testing	**Smoke Testing**
Validation- Basic & core functionality, tab, link, page & GUI	Validation- Basic & core functionality, tab, link, page & GUI and also find out the exact root cause /troubleshooting and package validation
Sanity is performed by tester	Smoke is performed by tester & package validation is done by developer
We do not write test cases & we do not execute test cases	We do not write test cases & we do not execute test cases
If we found defect in sanity we simply **reject the build**	If we found defect in smoke, we **reject the build** but we will **provide the root cause of the defect.**

Interview question-

1. What are your approaches, when you got the build?
2. What is difference between sanity testing & smoke testing?
3. What types of defect you have got in sanity testing/ Smoke testing?
4. Which testing you will perform in SIT Env./Testing, when you got the new builds?

   Answer- In My project, we will performed **Smoke testing or Sanity testing**
5. Are you writing test cases in smoke testing or sanity testing?

- **Answer-** No

6. Are you creating/log defects in smoke testing?
7. When developer is look into these defects, then what you will do? **OR** When you have rejected the build then what you will do?
8. Which testing, you will perform in your project when we get the new build? Why?

   - **Answer**- Smoke Testing or Sanity testing

     Why-

   1. Before going to system & functional testing, we don't get show stopper defect.

2. It will be **save the time**, for developer if we will inform that where is defect & their troubleshoot.

9. Which testing you have performed in your Origination

- Answer- In **my Origination we performed Smoke testing or Sanity Testing**
- In Smoke Testing, we are checking or validating stability of the build.
- In Smoke Testing, we will check Core functionality, Tab/Page, link validation, GUI/UI, navigation validation, etc
- In Smoke Testing we required 2 to 4 hr
- If we found defects in smoke testing, then we will reject the build and we will provide the root clause if defects.
- We will send us a mail to developer.

10. If we found defects in smoke Testing then what is your approaches?

- Answer- When we are randomly application **in smoke testing or sanity testing**.
- If we **found defects** then we will **reject the build**.
- Only critical defects are created in project management tool (JIRA/ HPALM) & inform to developer throw the Mail

11. Which testing will follow when you got new build?

a. Answer- In my organization, we are performing **Sanity Testing or Smoke testing**

- When we got **new build form developer** then we perform then **Sanity testing**
- In Sanity testing we are checking

1. Validating the **GUI/ UI of application**

2. Validating **core functionality (ex.** Recharge module- Mobile no. recharge)

3. Validating **the link**

4. Validating the **tab**

5. Validating the page/**navigation**

System & Functional Testing

- System & function testing, we will perform **after smoke testing or sanity testing** i.e. after **build stability**
- System & functional testing also called as **BBT (Black box testing)**
- In system & functional testing, tester tests **overall / entire functionality of the application** step by step from start to end
- According to **user stories**, **test scenarios** are prepared, **test cases** of test scenarios are **executed** in the system & functional testing
- The difference between System testing and Functional testing is that **functional testing is testing a single feature in a product works as specified**, while System testing is the **whole product/system** (functional & non-functional testing of the system)
- 4 types of System & function Testing

1. **Usability Testing -**	**(90 to 95%)**	**90 to 95 %**
2. **Functional Testing-**	**(90 to 95%)**	
3. Security Testing -	(0 to 5%)	
4. Performance Testing-	(0 to 5%) – Jmeter (Open source), Load runner (Licensed version)	

1. Functional Testing

- Validation **application/ build internal & external feature**
- Functional Testing 2 types
  1. Functionality Testing - Validation **application/ build internal feature**
  2. Non- Functionality Testing - Validation **application/ build external feature**

1. **Functionality Testing (BIEBSC)**
   - In this testing, we **check completeness & correctness** of the application / product as **functional point of view**
   - Functionality testing is a process to **check internal functionality** of the application depends / **based on the external functionality** / interface

- In this functionality testing, we **execute the test cases** step by step from start to end
- Validation **internal feature of build/ application**
- It includes:-

1. **B**ehavior coverage testing
2. **I**nput domain coverage testing
3. **E**rror handling coverage testing
4. **B**ackend coverage testing/ database coverage testing
5. **S**ervice base coverage testing
6. **C**alculation base coverage testing

1. Behavior coverage testing

- In this testing, tester test / validating the **behavioral of the object/ web elements**
- In Behavioral, we will validate **property of the object/ web elements**
- In this testing, we simply check whether, the **objects** are properly **working or not**

Object/ web elements	**Property**
Text box	Enable & Disable / Focus & Unfocused / accept user input
Radio Button	On & Off
Button	Enabled & Disabled / Click & Unclick
Check Box	Check & Uncheck
Link	Click & Unclick / Navigating or not

2. Input domain coverage testing

- Validating the **input which we are passing into objects or** Validation **what type of input we will pass in objects**
- Checking what is **Size or length of the object** & what is **types of object (i.e. data type) or** Validation **Data types of input** & **Size/ Length of input**
- **Input domain coverage testing maintain**

  A. BVA (Boundary value analysis) – input size or length
  In this, we check size or length of input value

  B. ECP (Equivalent class partition)- input data types
  In this, we check data types of input value

  C. Decision table testing techniques – Different input values combination sent – result

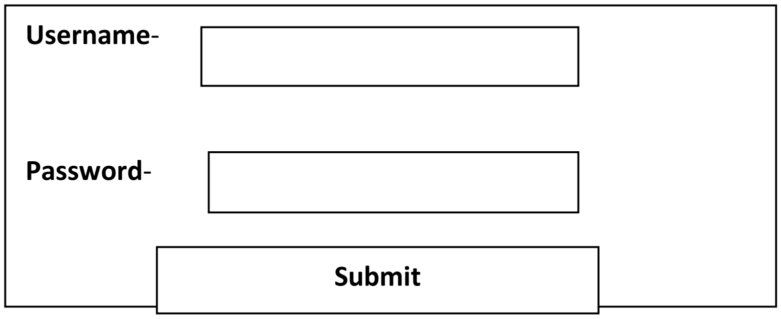
Where, we use different input combination

A. BVA (Boundary value analysis) –

- BVA we are **validating input Size or length into object / web element**
- Min & Max values of input on the objects
- Ex. US- Login Page –

  Username object accept – Mobile no. only

  Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no & 1 special charter



BVA	**Pass**	**Fail**
Username-	10 digit no.(Min & max)	11 digits, 9 digits no.(max+1, min-1)
Password-	4 digits no. (Min)	3 digits no (min-1)
	5 digits no. (Min)	3 digits no (min-1)
	6 digits no. (Max)	7 digits no. (Max+1)

B. ECP (Equivalent class partition)-

- Validating **data types of the input** which we will pass into object
- Ex.US- Login Page –

  Username object accept – Mobile no. only

  Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no & 1 special charter

ECP	**Pass**	**Fail**
Username-	Integer (0 to 9)	Charter, Fraction, Binary, etc.
Password-	String (A-Z, a-z, 0 to 9, Special charter)	Null/ Blank

C. **Decision table testing techniques –**

- Validation **different input combination values** sent into the object then what result
- Ex. US- Login Page –

    Username object accept – Mobile no. only

    Password- combination of 4 to 6 charter which contains 1 Capital letter, 1 small letter, 1 no & 1 special charter

Object	Rule/ Condition 1	Rule/ Condition 2	Rule/ Condition 3	Rule/ Condition 4
Username	Valid	In-Valid	Valid	Blank
Password	Valid	Valid	In-Valid	Blank
Submit	press	Press	press	press
Result/ O/P	Home Page	Error message	Error message	Error message

Examples

E.g. Textbox should accept only 4 to 6 characters (take text box from above-pw)

BVA		**ECP**	
Size	**Result**	**Valid**	**Invalid**
Min = 4	Pass	0-9	Space
Max =6	Pass	a-z	_
Min+1 = 5	Pass	A-Z	
Min -1 = 3	Fail	Special character	
Max+1 =7	Fail		
Max-1 = 5	Pass		

Textbox should accept only 4 to 6 characters - ?

BVA		ECP	
Size	**Result**	**Valid**	**Invalid**
Min = 4	Pass	a-z	0-9
Max =6	Pass	A-Z	Space
Min+1 = 5	Pass		Symbol
Min -1 = 3	Fail		-_
Max+1 =7	Fail		Special character
Max-1 = 5	Pass		

Mobile Number should accept 10 digits

BVA		ECP	
Size = 10	**Result**	**Valid**	**Invalid**
Min = 10	Pass	0-9	a- z
Max = 10	Pass		A-Z
Min+1 = 11	Fail		Special character
Min -1 = 9	Fail		Symbol
Max+1 = 11	Fail		Space
Max-1 = 9	Fail		-_

Password – Text Box – Should allowed

8 to 14 digits | 1CAP char | 1 Small Char | Special Symbol |No space and _|

BVA		ECP	
Size	**Result**	**Valid**	**Invalid**
Min = 8	Pass	A – Z	Space
Max =14	Pass	a- z	-_
Min+1 = 9	Pass	Special Char	
Min -1 = 7	Fail	0-9	
Max+1 =15	Fail	Symbol	

Max-1 = 13	Pass		

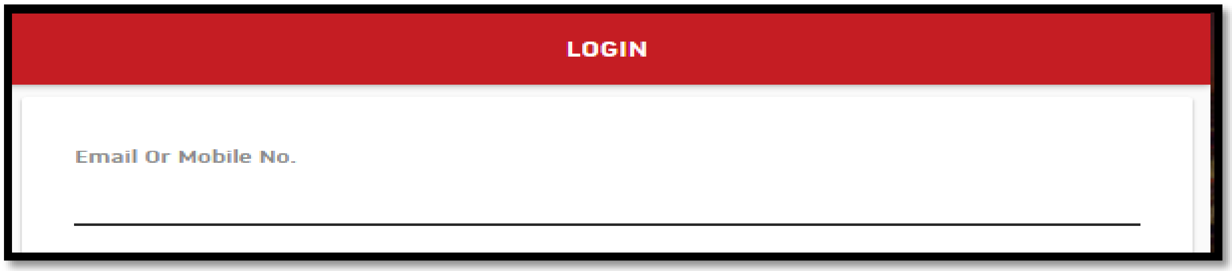
Check BVA & ECP for cycle stand having 100 cycles

BVA		ECP	
Size	**Result**	**Valid**	**Invalid**
Min =1	Pass	0-9	-_
Max =100	Pass		Special Character
Min+1 =2	Pass		Symbol
Min -1 =0	Fail		Space
Max+1 =101	Fail		A – Z
Max-1 =99	Pass		a- z

1. **BVA-**

- Ex. Dram 11 application Login page with mobile no. only

Size/Length	**BVA**
Min – 10 digits	Pass
Max – 10 digits	Pass
Min-1 = 9 digits	Fail
Max+1 = 11 digits	Fail

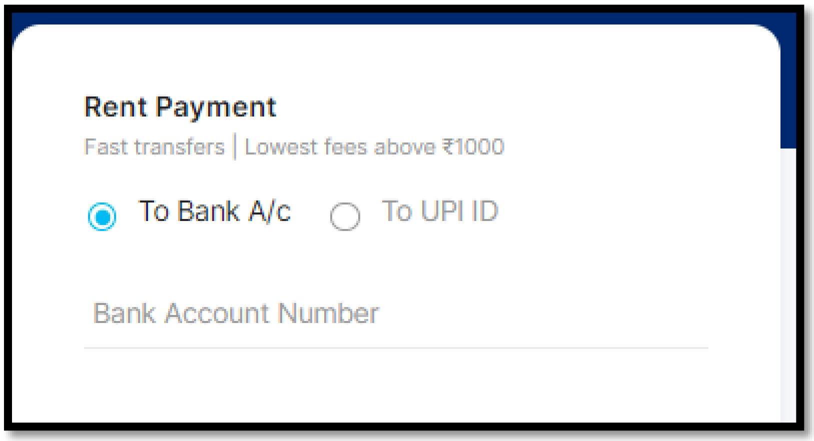


2. **ECP**

- Ex. Dream 11 application Login page

Data types	**ECP**
Integer	Pass (Mobile no.)
Charter	Pass (Email id)
String	Pass (Email id)

- **Ex.** For BVA & ECP


Rent Payment

Fast transfers | Lowest fees above ₹1000

[✓ To Bank A/c] To UPI ID

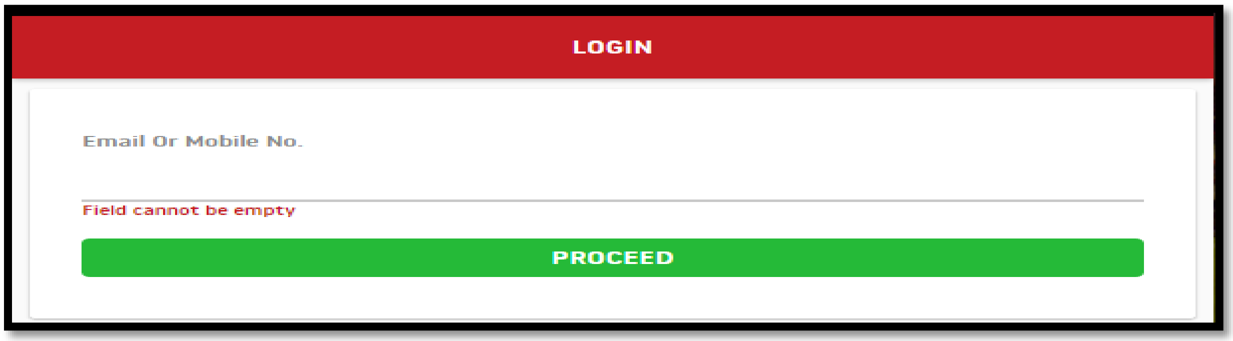
Bank Account Number

Size & Length	**BVA**
Max – 30 digits	Pass
Min- 11 digits	Pass
Max+1 digits	Fail
Min- 1 digits	Fail

Data Types	**ECP**
Integer	Pass
Charter	Fail

3. **Decision Table testing techniques-**

- Ex. Dream 11 login page


LOGIN

Email Or Mobile No.

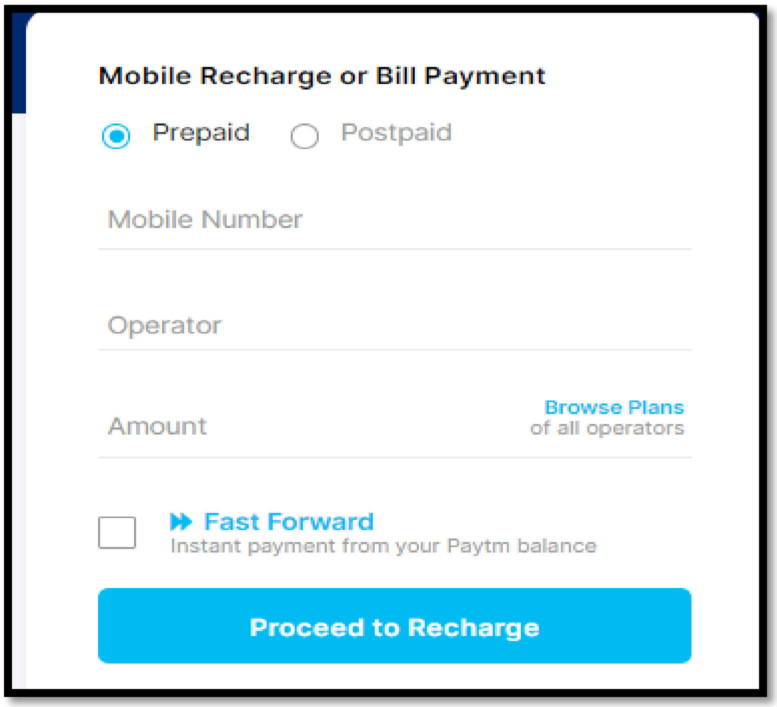
Field cannot be empty

PROCEED

Objects	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
Email or mobile no	**9923475781**	**9422334455**	**xyz@gmail.com**	**pqr@outlook .com**

Result	Pass	Pass	Pass	Pass

- Ex. Paytm recharge module



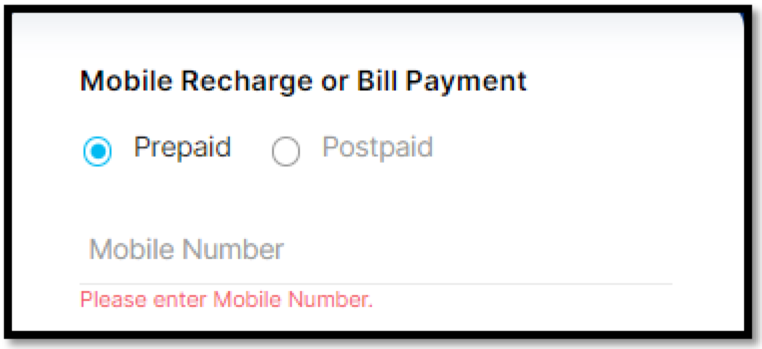
Objects	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Mobile no	Valid	In – valid	Valid	Valid	In-Valid
Operator	Valid	Valid	In-Valid	Valid	In-Valid
Amount	Valid	Valid	Valid	In-Valid	In-Valid
Result	Pass- Recharge	Fail- Error message	Fail- Error message	Fail- Error message	Fail- Error message

3. Error handling coverage

- In this testing we check, when we **enter invalid data / blank data** in the object then system is **displaying / showing error message or not**
- Validating the **different types error message which generated** in web page/ Build/ application, when we pass invalid test data- Build shows error message
- Validating the what are **different types of error message are present in the objects**
- Ex. if we enter 2 digits only in the mobile number text box, then system should highlight text box with red color with error message "**please enter 10 digit mobile number**"
- In this testing, we check system **show error message or not**

- **Ex.** Paytm- Recharge module

  If we will pass null or blank – Error =”Please enter mobile no”



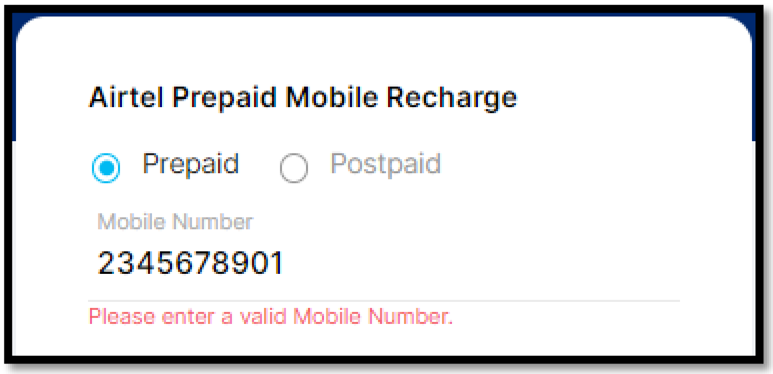
Mobile Recharge or Bill Payment

◉ Prepaid ○ Postpaid

Mobile Number

Please enter Mobile Number.

  If we pass invalid data – Error = “Please enter a valid mobile no”



Airtel Prepaid Mobile Recharge

◉ Prepaid ○ Postpaid

Mobile Number

2345678901

Please enter a valid Mobile Number.

4. **Backend coverage testing / database testing**

- Validating all **frond end operations** are **stored in database**
- Front end operation or **data entered in the front end is stored in the database** or backend
- As a tester, we check or validate, whenever data has been entered in the front end that data has been stored in the database or not **(validating front end with backend)**
- We check or validate data can be **fetched or not**
- So, we simply fire the **SQL queries** & we get response in the form of table (rows & column), where we check whether **data is stored or not**

Or

- SQL query is used to **fetch data or to check stored data**
- Ex. Paytm- Recharge model – Successfully & Un- Successfully
- Backend in database data either it is stored or not
- SQL quires write-

Select * from TN where order_id= ‘123456789’;

5. **Service level coverage testing**

- Being a tester, we validate / check **function sequence of the application/ product**
- Validating **sequential order of functionality application**
- Product owner prepares **functional flow diagram** & maintain the sequence of modules & sub modules
- So, in this type of testing, **sequentially functionality of modules, application / build** are tested as per the function flow diagram
- Ex. Paytm – Recharge module → Recharge Icon– **Mobile No. Circle, Amount, Processed to pay** →**Promo code** → Payment Tab → Thank message & Scratch card
- Ex. Paytm – Recharge module → Recharge Icon– **Mobile No. Circle, Amount, Fast forward, Processed to pay** →Payment Tab → Thank message & Scratch card

6. **Calculation base coverage testing**

- It’s a part of functionality testing, during calculation base coverage testing, as a tester we validate / check arithmetic operations (i.e. addition, subtraction, multiplication, & division)
- **Ex. Paytm** – Recharge module → Recharge Icon– Mobile No. Circle, Amount(499rs), Processed to pay →Promo code (10%)→ Payment Tab (499-49=450rs)→ Thank message & Scratch card
- **Ex. Flipkart** – add 3 items in the cart – price of item 1 is 500/- - price of item 2 is 300/- price of item 3 is 200 – show total amount is 1000/- - remove 1 item of 200 from the cart – show 800/-
- **Ex**. Movie tickets booking
- **Ex**. Train tickets booking
- **Ex**. Travels tickets booking

Non-Functionality Testing

- It's a process of **checking external functionality or external feature** of the application or build
- Validation **external feature of build/ application**
- In this testing, we check, whether the application is **running on particular operating system (OS) & browser or not**
- Non- Functionality Testing- types or different converges/ testing or includes- **(RCCIISPG)**

1. **R**ecovery testing
2. **C**ompatibility testing
3. **C**onfiguration Testing
4. **I**nstallation testing → **Not performed in SIT environment**
5. **I**ntersystem testing
6. **S**anitization testing
7. **P**arallelization testing → **Not performed in SIT environment**
8. **G**lobalization testing

1. Recovery Testing

- It is also called as **reliable or reliability testing**
- In this testing, as a tester, we **check whether, application / system** is able to recover or can **recover from abnormal condition / situation to normal condition / situation or not**
- We **validate**, whether application is **capable to handle abnormal situation** / condition **or not**
- **Recovery requirements / point** are given by the **client / customer** (i.e. system should recover from start point or should resume from stopped point)
- **Ex.**

1. **Downloading-** If we / you are downloading movie of 1GB & consider we have lost internet connection but 800MB has been downloaded, the connection resumes the

movie is started downloading from 800MB (from which point it should be decided by the client)

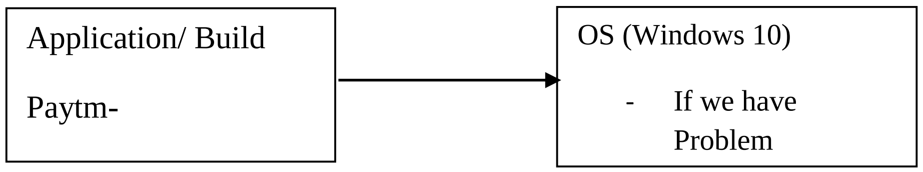
2. **Google- if** we are accessing Google and suddenly page has stop working due to Internet connection lost, then show you are offline message by Google & once internet connection back then connection resumes from staring to load Google
3. **Paytm** – While doing Payment –Add Account Number – Amount - Press the back button – Paytm application it will go again Paytm page.
4. **Amazon-** when we are buying something from amazon, after entering address we get redirected to payment page & suddenly application crash & when we reopen application we have to start again from payment page

2. Compatibility Testing

- In this testing, we simply check, whether **application / software is supporting to users expected platform or not**
- We check, whether **build is compatible with users expected platform or not**
- In compatible testing, there are **2 categories,**
  1. **Software Compatibility**- OS support, Browser support & Application Support etc.
  2. **Hardware Compatibly**- Parts, Printers, & External Devices etc.
- There are **2 types of compatibility testing**

1. Forward Compatibility Testing
2. Backward Compatibility Testing

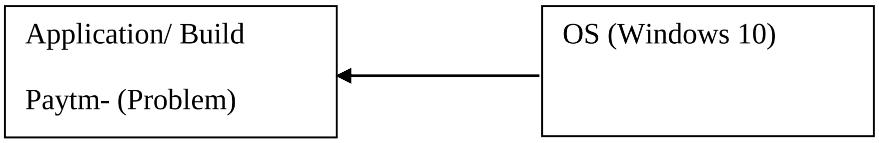
1. Forward Compatibility Testing



- In this testing, if **build / application is correct or ok but OS / Browser does not work properly**, then it comes under the forward compatibility testing
- If there is an issue in OS, then **IT / technical support / administrative team – work on it** or **resolve the problem**

- In this, we get **less number of defects / bugs**

1. Backward Compatibility Testing



- If **OS / Browser is ok / correct**, but **build does not work properly** then it comes under backward compatibility testing
- In this testing, we get **more no. of defects / bugs**

Compatibility Testing includes

1. Operating system compatibility testing- **Not Involved**
2. Browser compatibility testing

❖ There are **2 types of browser compatibility testing**
  1. Cross browser compatibility testing
  2. Version control / comparison compatibility testing

1. Cross browser compatibility testing

- In this testing, tester tests the **build on different different browsers** like Chrome, Internet explorer, Mozilla, Edge, Safari, Opera-mini, etc.
- Validating either **application/ build is supporting to all browser**
- **Ex.** Paytm- Rent payment module – Different browser- Chrome, Firefox, IE, Edge, Safari, Opera-mini, etc.

2. Version control / comparison compatibility testing

- In this testing, tester tests the **build on different different version of same browser**
- Validating either **application/ build is supporting to one browser with different version**

- **Ex.** Application/ Build test - **Paytm- Recharge module** on Chrome browser – V91, V90, V89, V85, V80, V75 etc.
- By using **VM (Virtual machines) & remote desktop** we can use same browser with different version

3. Configuration Testing / Hardware Testing

- I **am not part of this testing**, I am involved in **service level application testing**, but I am aware, how it works. There is a **separate team**, who does this testing
- During this testing, tester tests whether **application / software is supporting to different hardware's or not**
- Validation of **application/ software either support to hardware** devices or not
- **Hardware**- printers (dot matrix, laser)
- **Ex.** Paytm- Invoice download click- print page
- **Ex.** Paytm- Ticket booking / Movies → Invoice download →click on Print → Printer setting
- **Ex.** Paytm- Travel module- Ticket – Invoice download/ Ticket download- print button – hardware device pop
- **Ex.** Paytm- Recharge module- Download invoice- PDF/ invoice file- **Print icon** – Print page will display

4. Installation Testing

- I **am not a part of this testing**, I am involved in **service level application testing**, but I am aware about it also
- It's a process of checking **installation of software / application in to existing system of user as per user expected platform**

Application or Build or Build plus existing Software	Customers Expected Platform	Set program Execution
		Easy Interface
		Occupied Disc space
		Check uninstallation

1. **Set program Execution –**

In this, he has to check whether all the setup files are present / there or not

Package has all the files or not

2. **Easy interface –**

   Installation process should be user friendly, so user can navigate easily so, tester simply check the installation process

3. **Disc interface –**

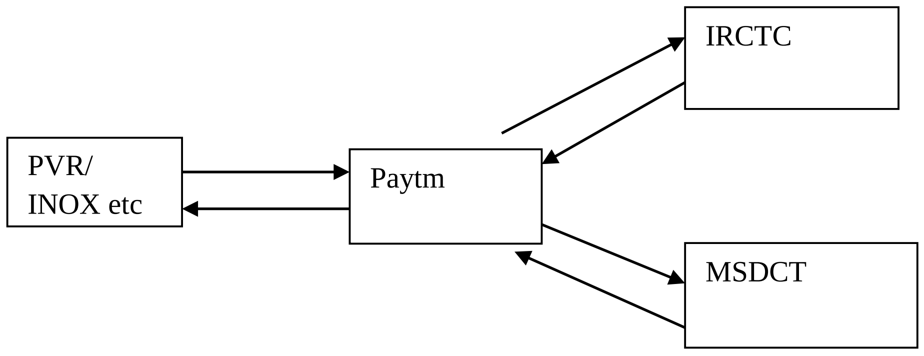
   Tester checks available disc space also & total disc space

4. **Check uninstallation –**

   Tester checks, whether installed software can uninstall from system or not

5. Inter System Testing

- It's a process of checking whether **our application / software shares data or information or resources with other application or not**
- The **data communication** is done through the **XML**
- **Banking domain** companies uses this type testing
- **Ex.** if you want to recharge JIO number from phone pay. Simply phone pay fetches information from JIO app, so this comes under the inter system testing
- **Ex.** withdraw money from other bank ATM
- **Ex.** Paytm- Electric bill will pay – MSDCT
- **Ex.** Paytm- Train ticket - IRCTC


IRCTC
PVR/ INOX etc
Paytm
MSDCT