GLOBALIZATION TESTING

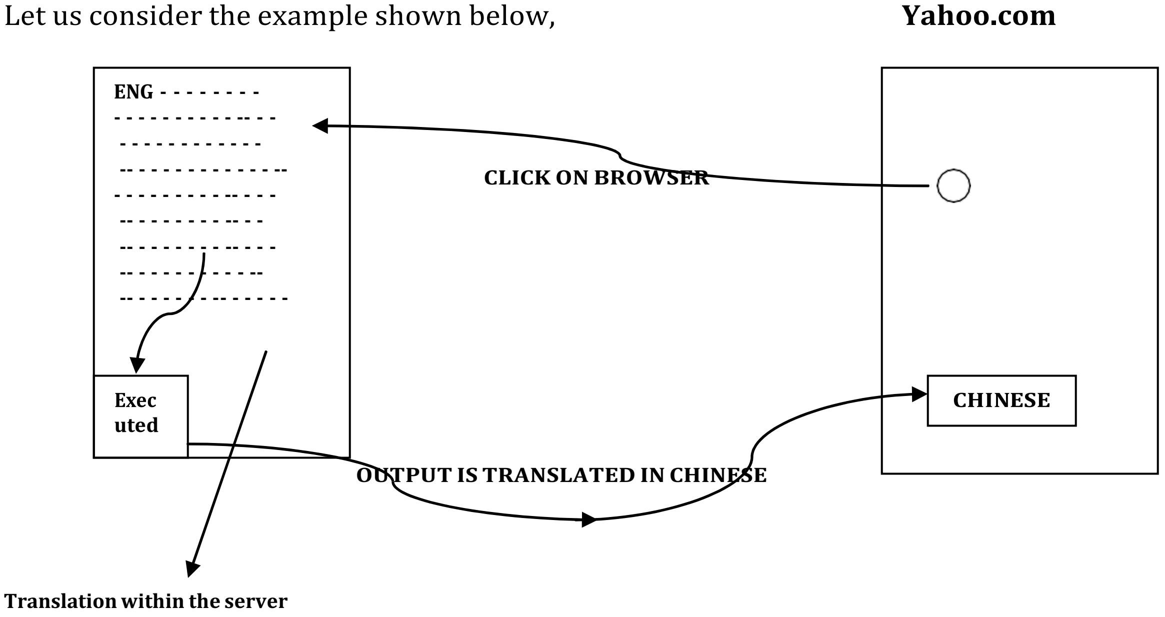
Developing the application for multiple languages is called **globalization** and testing the application which is developed for multiple languages is called **globalization testing.**

There are **2 types** of globalization testing,

- Internationalization Testing (I18N testing)
- Localization Testing (L10N testing)

Internationalization Testing or I18N testing

Let us consider the example shown below,


Yahoo.com
ENG - - - - - - - -
CLICK ON BROWSER
Exec
uted
OUTPUT IS TRANSLATED IN CHINESE
CHINESE
Translation within the server

Suppose if we want the application in Chinese, then we click on the browser it will take to the server where the program is in English, from there it is executed and the output is translated into Chinese and displayed in Chinese language.

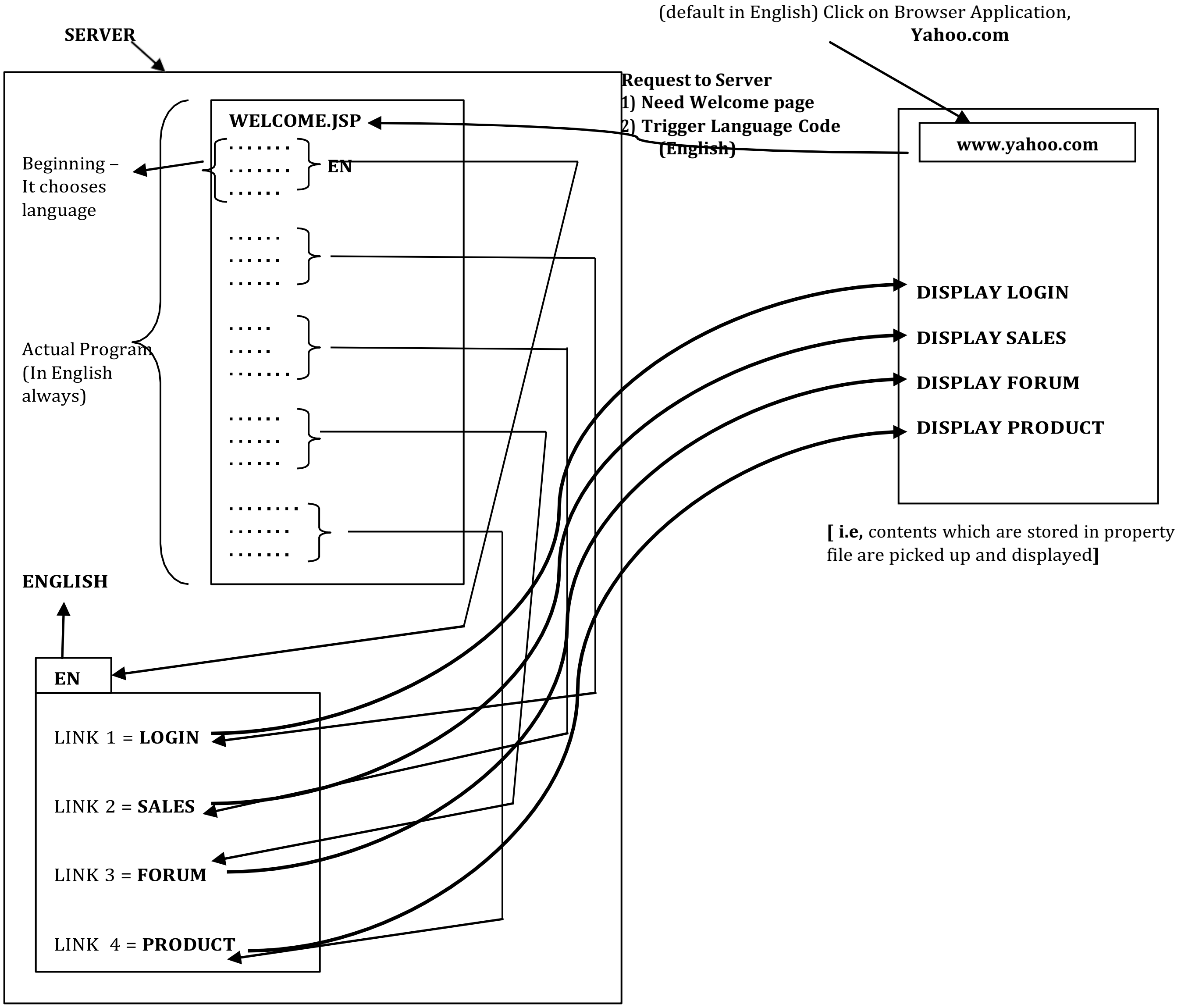
How do we do this translation of language ? – By using a translator ? – NO. Then how do we do it ? Before thinking, let us see the **drawbacks in using a translator**.

When the programs are translated from English to different languages, using translator the following occurs,

- Meanings are changed
- Not conveying the feeling. Thus usage of translators are ruled out. For ex – in English, the word "welcome" when translated to Kannada using translators will mean "*baavige baa"!!!*

The above problem is handled by using **property files.** ***Property Files are nothing but files containing source data (like, a Notepad). In Java, it is called Resource Bundle.***

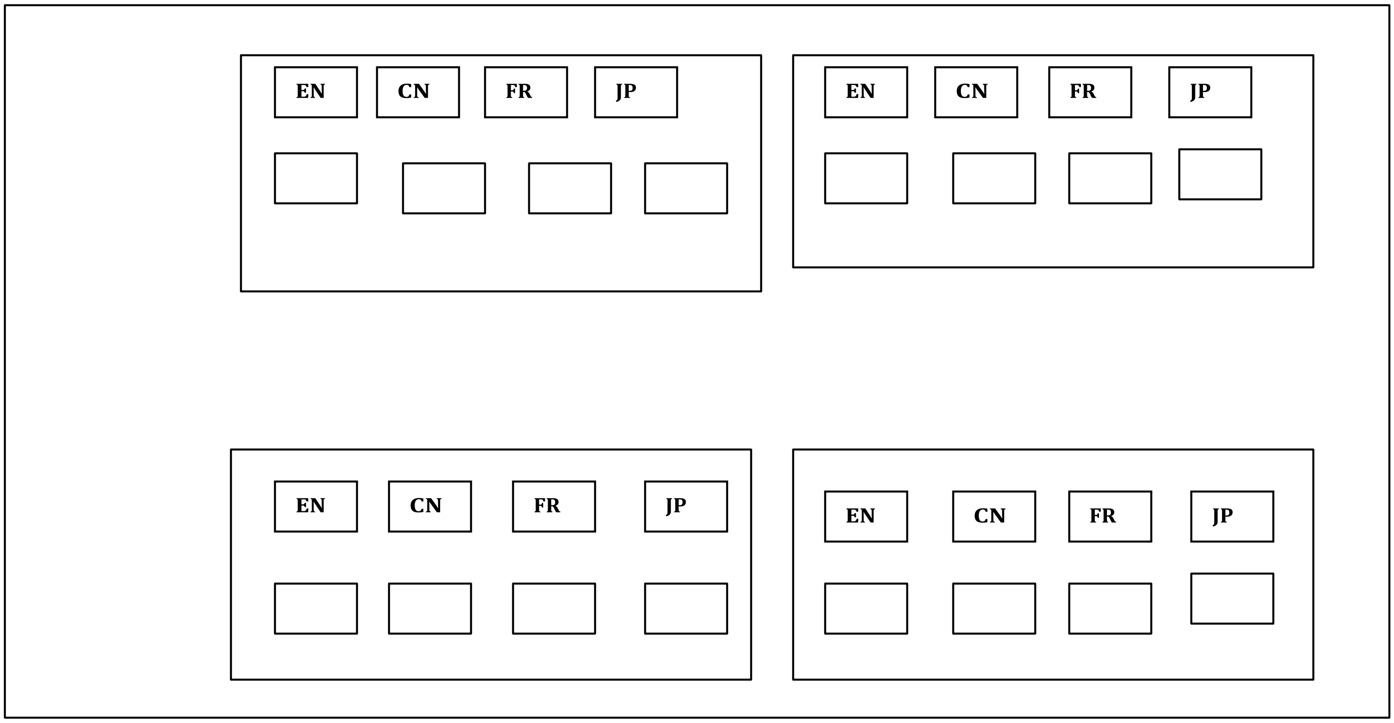
Let us see how I18N testing works for English language,



Write a common program in English for all languages and have different property files for different languages. Suppose, if we have 10 different languages, then we should have 10 different property files. **We see how the above request works,**

Click on the browser, it takes a request to the server saying the following – need Welcome page, trigger the language code EN
It takes to the program where the language is executed and takes the property file of ENGLISH.
Now, the English property file is selected and the next coding in the program connects to its respective links. **For example,** if the next set of coding is for Login – it connects to Link 1, picks up the data stored there and displays it in English. Same procedure followed for other links also (like sales, forum and products).
Again, the same procedure followed for other languages also.

Who writes these property files ?
1st, the developers writes a property file in English. Language experts takes this and manually write the property file and gives it to developers.
Now, the developers place this property file as shown below in the server.

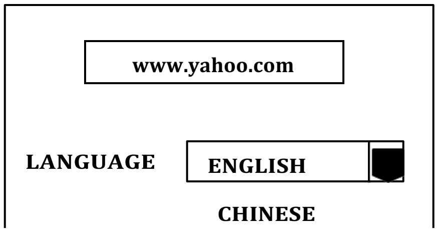


Let us see how we will translate into Chinese language and test for it.
Same procedure like above
Click on browser. Select for language Chinese before you click.
After clicking, it takes it to Welcome page with a trigger language code CN
So, it takes to the CN property file and display the contents (Chinese) stored in the file according to the link it gets connected (**for ex,** LINK 1 to Login display the contents of LOGIN. Similarly for other like Sales, Forum, Product also).

WELCOME.JSP

. . . . .
. . . .   **CN**
. . . . .

. . . . .
. . . . . .
. . . . .



www.yahoo.com

LANGUAGE **ENGLISH**

CHINESE

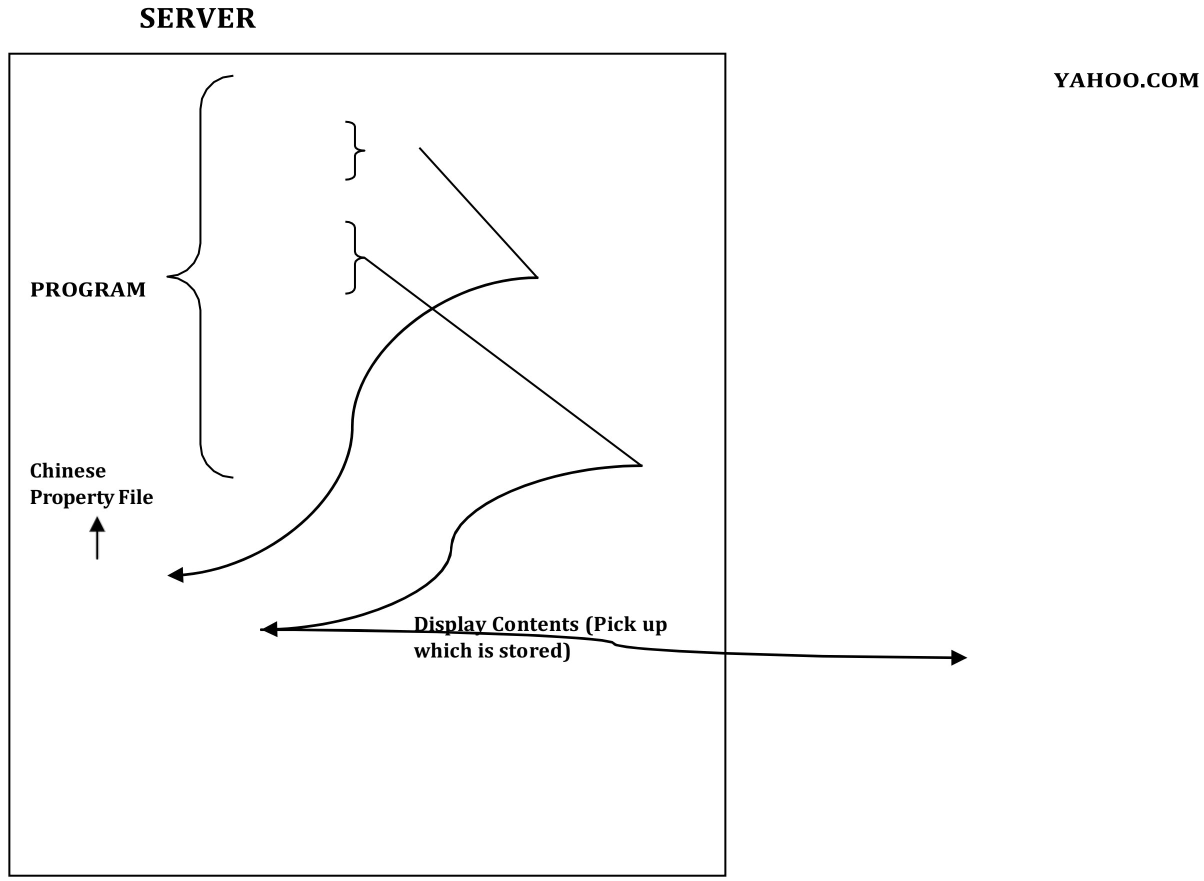
LOGIN (8 Property Files)

SALES (8 Property Files)

EN – English
CN – Chinese
FR – French
JP – Japanese

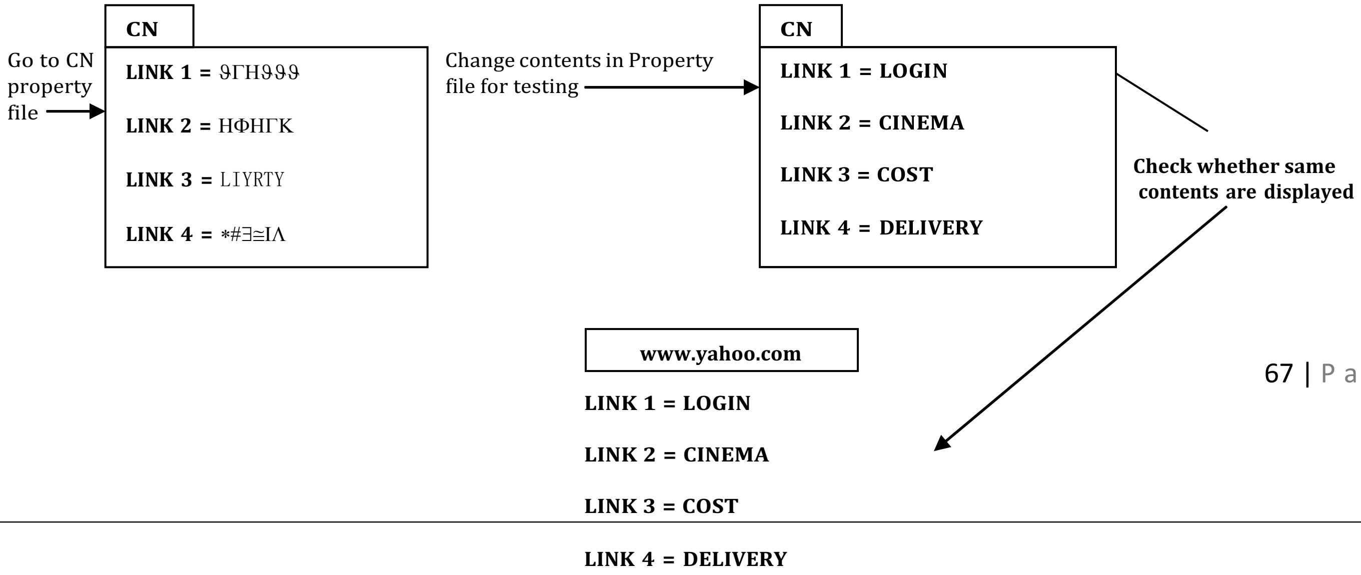
FORUM (8 Property Files)

PRODUCTS (8 Property Files)



Now, we have changed the language from English to Chinese. Now, *how do we test whether it is in Chinese displayed or some other language ?*

Go to Chinese property file and change the contents in the file. **For ex,** see below figure,



DISPLAYED –Testing is fine
NOT DISPLAYED – Go for changes

These links are called KEYWORDS

After we do changes in property file, we once again select the Chinese language and see whether the change we have made in the content is displayed as same. If it is displayed, then testing is fine.
As a TE, we must do **changes in property file, and not in the program**(actual program).
If it is not displayed, catch the bugs and the bugs are to be fixed only in the program.

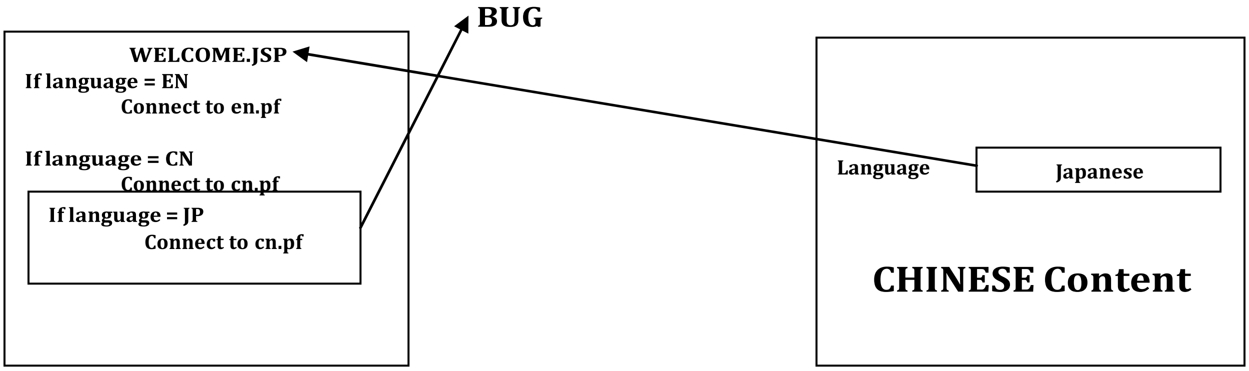
Thus, we do I18N testing for the following,

- Check whether content is in right language
- Check whether right content is in right place

Now, let us see the possible bugs in I18N testing :-

1) If I select English – displayed in English
If I select Chinese – displayed in Chinese
If I select Japanese – displayed in Chinese

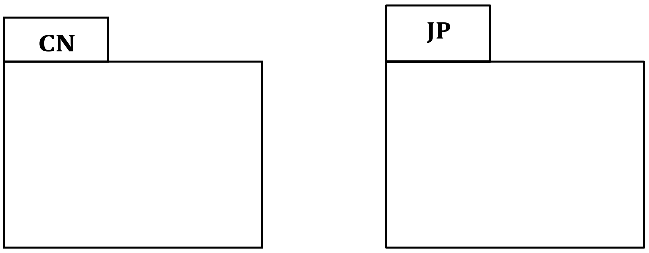
Why does this happen ? – **observe**,



```
WELCOME.JSP
If language = EN
        Connect to en.pf

If language = CN
        Connect to cn.pf
If language = JP
        Connect to cn.pf
```

Display content in Chinese

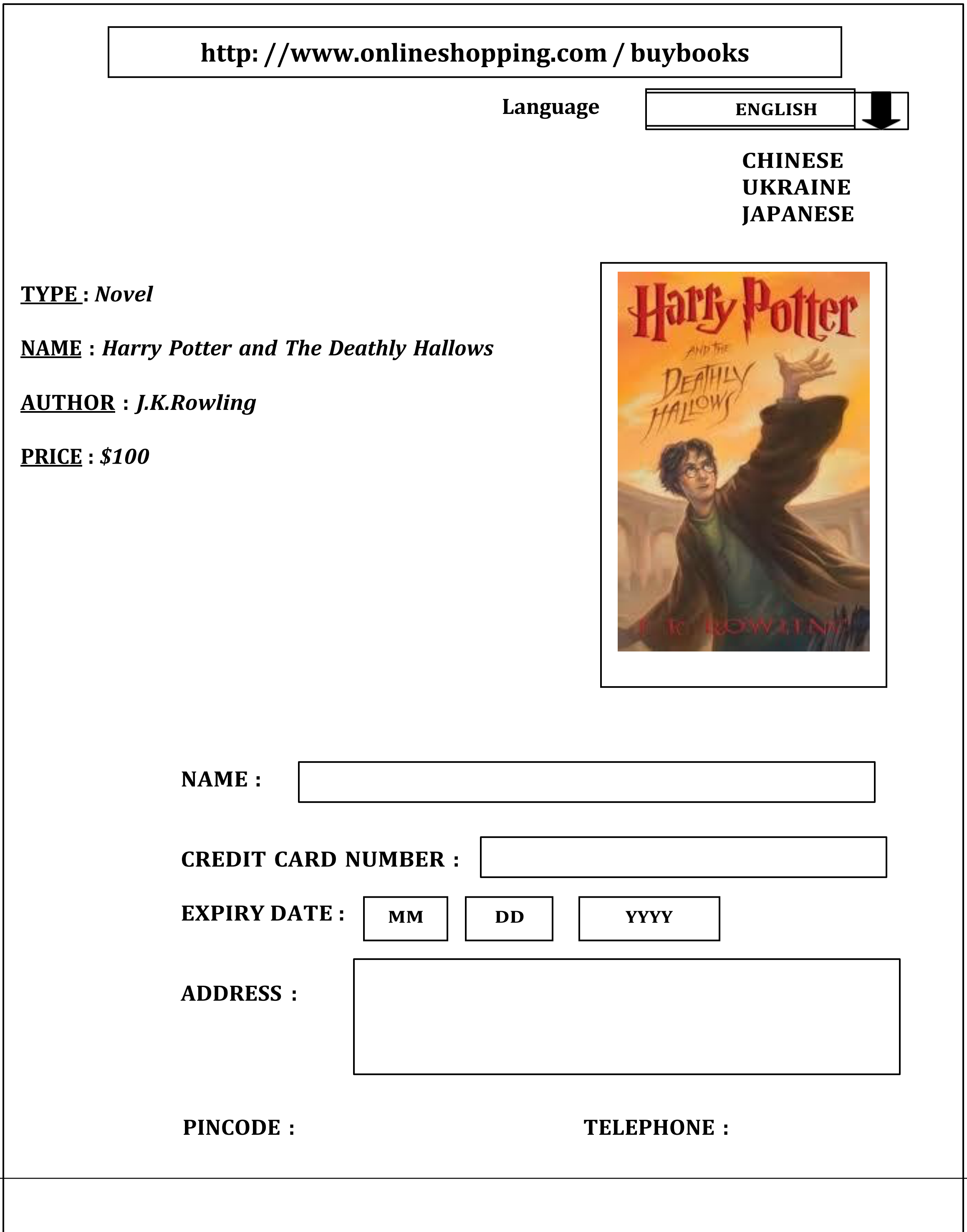


Why if I select Japanese, it is displaying in Chinese ?
Developers they do copy and paste the coding and forget to change the respective property file like for Japanese – instead of *connect to jp.pf* , they forget to change it and it still remains in the *connect to cn.pf* and thus displays the contents in Chinese language

2) Check for reverse direction language i.e, how the break works. There are 2 types of languages – Uni-directional language and Bi-directional language. Uni –directional language starts either from the left or right. Bi-directional language can start either from right or left. So we have to check if the text is displayed properly as per the language

3) Alignment Problem :- We have to check whether the alignment specification for different languages is followed properly. Right alignment or left alignment.

Now, let us do ***globalization testing for online shopping for books*** *as shown below* :



http: //www.onlineshopping.com / buybooks

Language **ENGLISH**

CHINESE
UKRAINE
JAPANESE

TYPE : *Novel*

NAME : *Harry Potter and The Deathly Hallows*

AUTHOR : *J.K.Rowling*

PRICE : *$100*

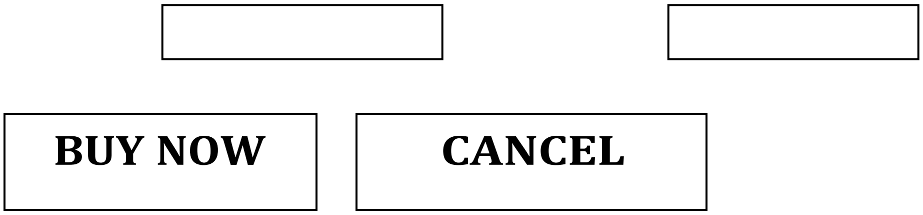
NAME :

CREDIT CARD NUMBER :

EXPIRY DATE : MM DD YYYY

ADDRESS :

PINCODE : **TELEPHONE :**



How do we connect it to other languages and test it ? – go to respective property files – change the contents and test it. If all the content is in right language, then I18N testing is successful.

<u>Localization Testing</u> (L10N testing)

Format testing is nothing but Localization testing (OR) Testing done for format specification according to region/country is called L10N testing.

Let us see the different format testing we do in L10N testing,

a) *<u>Currency Format testing</u>*

Here, we do not worry about the functionality of the format (like $ is converted to Rs or not). We only test whether the $ should be in the first or the last.
Ex : 100$, $100, Rs100. The standard should be as per country standards.

b) *<u>Date Format testing</u>*

Here, check whether the date format is according to its country format. This is also L10N testing.
Ex : in US, date format is : MM – DD – YY
in India, date format is : DD – MM – YYYY

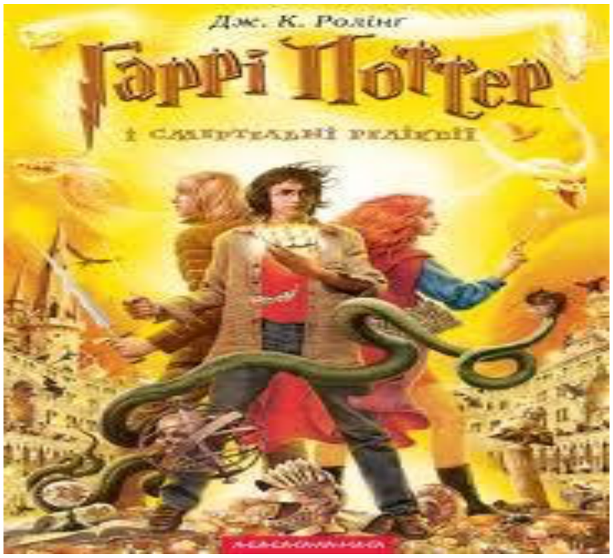
c) *<u>PinCode Format testing</u>*

There are countries having Pincode with characters like AB100. Here,
Checking for the format in pincode is L10N testing.
Checking whether AB is translated to Chinese is I18N testing.

Date format
Currency format **L10N testing**
Pincode format

d) *<u>Image Format testing</u>*

In image – only name on the image can be changed – the image cannot be changed. Thus we must have multiple images depending on the country. When we click on Ukraine, it goes to Ukraine server, takes the contents and also selects Ukrainian image and thus contents are displayed in Ukraine. This is shown below,



For I18N testing – no images with text is allowed. In case, we want to have images – then we must have different images for different languages.

If we do any color change, then it is called as L10N testing (like national flag where the color is specific to its country).

Tool tip – ALT + TAB – move the mouse on the image – keep for 1second – we get a small box explaining the image – the tool tip must be changed to the corresponding language – if it does not, then it is a bug.

Now, we see a person from India sitting in China and wants to browse in English.

In this case, we have to go and change the locale which is available like below,

http : // hp.cpm / sales.html locale = en_US

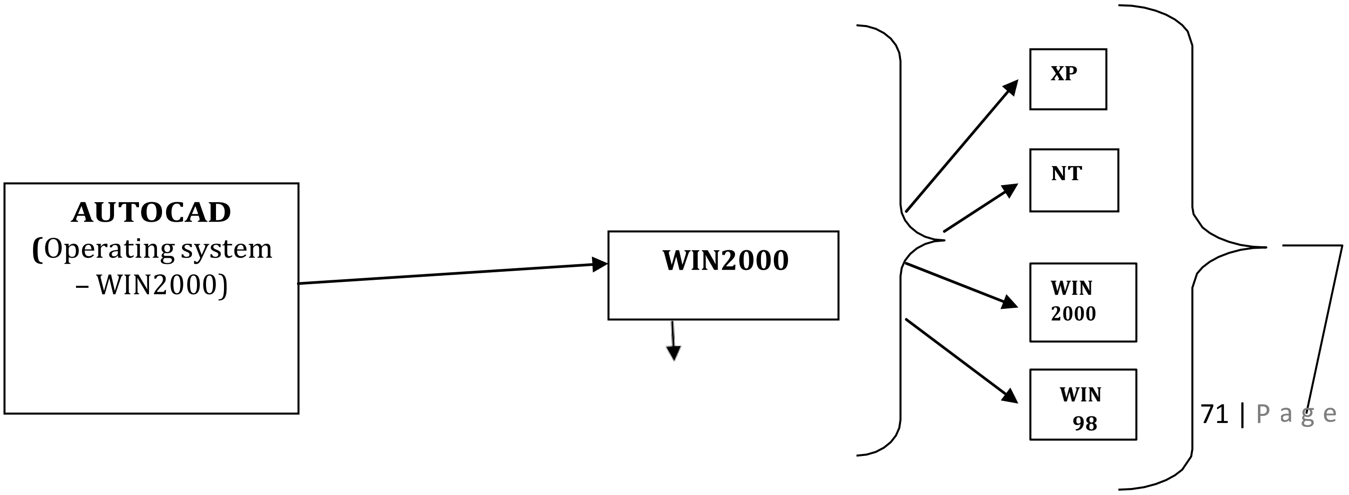
Language code -> en_US ->country code
fr_FR
de_DE
en_DE

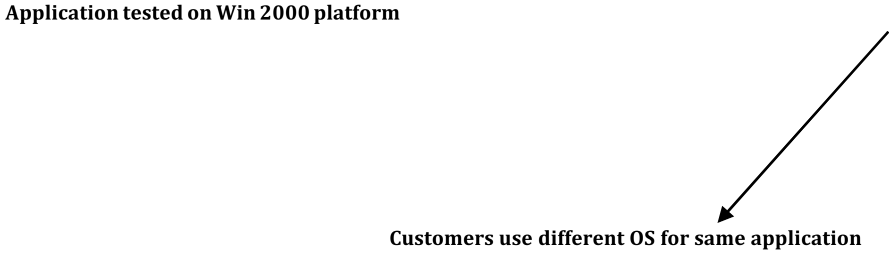
To be there in selected language – Go to Control Panel – change to language – come to browser – ctrl+shift (or) alt + Shift – change the text to that language.

COMPATIBILITY TESTING (CT)

Testing the functionality of an application in different software and hardware environment is called Compatibility testing.

Consider an example below,




Application tested on Win 2000 platform
Customers use different OS for same application

AutoCad with Win2000 was tested on Win2000 platform. Once it is tested, the product is sold but the customer who buys it uses in different platforms like XP, NT, Win2000, Win98 etc.

Customer therefore (or) End-users therefore blame the s/w, they don‟t think about the OS they are using because the application was tested on Win2000 and the user may be using it in XP.

Thus in order to overcome the above scenario, we do test the application in all platforms and then sell the product.

In the above example, we run the application in different platforms and if we find a bug in any of the platforms we have tested (say, Windows 98), then the s/w is given back to the developer for fixing the bug. Once, the bugs are fixed, then we have to continue testing not only on the platform which has bug, but also once again for all the platforms because fine tuning may cause a change in other platforms. Thus, maybe we may use one of the following scenarios after fixing the bug,

a) if we are sure of the area of the bug, then test only that area.

b) if we are not sure of the bug – then go for testing all the platforms.

Now, the question arises do compatibility testing is done for all platforms ? – NO. Observe how it is tested, Suppose only 8 TEs are present for testing, in this case –we don‟t have sufficient people for testing on all platforms. So do we stop testing ? – NO. Then how will we test – see below,

Now the TE will see for maximum use of platform by customers and do test for only those platforms.

PLATFORM
XP – 70 %
Win2000 – 20 %
Windows NT – 7 %
Windows 98 – 1 %
Windows ME – 1 %

In this example, TEs they look for most widely used platform i.e, in this case XP, Win 2000, NT, etc and do testing for all the above said platforms and do not test for the minimum used platform. If we are not testing other platforms doesn‟t mean it will not work. It will work on other platforms also, but if any bugs or issues arises – we are not fixing and not responsible for it. For this scenario to be avoided, sometimes in the product they mention for which OS it supports.

Now, we can think where we can do Compatibility testing ? – when we cannot force the users to use an application, then we go for CT.

Now, let us consider an example below to see where and how we do CT. (Figure in the next page)

If we take HP, IBM, SUN – these are h/w companies – they manufacture servers, when they want to sell the product – they can‟t sell the product only – therefore these companies have their own servers.

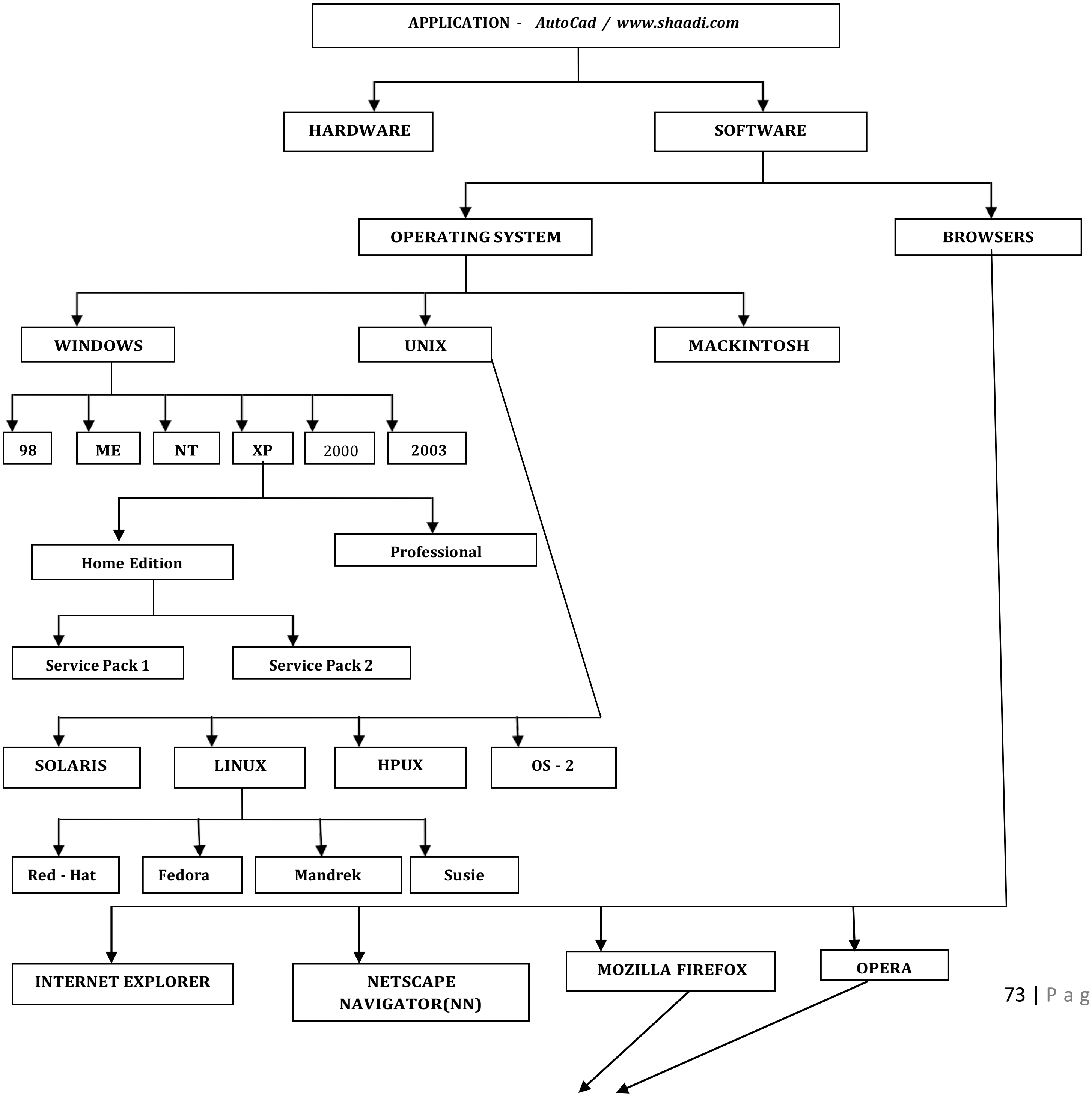
SOLARIS and UNIX can be tested by using OpenSource Internet Explorer (IE)

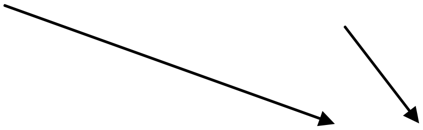
By default, IE comes with OS. IE is always tightly embedded with OS, so we cannot remove OS from IE [if we try remove, may be the application may not work].

IE generates lots of revenue to the company as it is robust and has got loops of security in it.

Opera claims it is the fastest browser. IE is not Microsoft original product [it is actually got and combined with other applications]

Developer builds www.shaadi.com and installs it in the testing server.
For the above application, (shaadi.com) – 4 Test Engineers are given 5days for testing. The testing is as follows by 1 engineer which is the same for the rest of the testing team,
1 TE opens OS – WIN 98 – Test for IE 5.0
Opens OS – Win 98 – Test for IE 5.5
Opens OS – WIN 98 – Test for IE 6.0
Thus for 4 TE to test 1 OS and 8browsers within 5days, it takes 40days to test in 1 OS.
So far all OS to test 40*6 = 240, it takes 240days for the TE to test the application.
Like this list of all flavours of platform are available, but we have to test for the OS as browser which is in maximum use. Thus for Windows, we do test for WIN 2000 and Windows XP which is widely used





DIFFERENT VERSIONS OF ALL THESE

Similarly Mackintosh is used by Apple computers and they sell in Europe and US.
Similarly for browsers we do test for the following cases which are widely used.
IE – for 5.5 and 6.0 – we do CT
NN – for 6.0 and 7.0 – we do CT and also for Mozilla Firefox we do CT
So, we always think about ROI (Return on Investment) and do the testing and also follow the criteria below,

- Number of users are less – we don‟t prefer CT
- Number of users are more – we prefer CT

WINDOWS 2000 Professional – normally corporate companies

Now, **let us see the set-up of the environment for AutoCad testing with OS**
Install Win 2000 in your machine and start testing for CT (1 of the platforms)
Similarly test for the next platform – IE (IE 5.5 and 6.0). open the IE 5.0 and test for it.
In this window, allow only 1instance of IE i.e, 1st we should test for IE 5.5, remove and then allow many instances (i.e, to have both 5.5 and 6.0 at the same time)
After we finish testing for the 2 available most widely used platform, we move on to the next platform NN 6.0 and 7.0 – open NN 6.0. here, the window allows 2instances i.e, to have both 6.0 and 7.0 at the same time.

How to handle bugs if we are testing for different platforms
Testing functionality, integration and end-to-end testing on various platforms is what we do in Compatibility testing.

Compatibility issue – a feature not working in 1 OS, but working fine in all other OS. It happens because program written by developer is not working in only 1platform, but working on all other platforms.
The compatibility issue is sent to developer for fixing the bugs and sent for testing. Here, to retest the application, we have to uninstall the OS and not the browser and re-install the OS and test.

Functionality issue – also called *Functionality defect issue* – a feature not working in all platforms/OS.

It is also possible to have different platforms in the same machine. For this, each platform(OS) should be in different drive like below. We can run 3OS in same m/c in different drives – C: Windows 98, D: XP,
E : Windows 2000

Hardware Compatibility Test

- Test on different processors,
- Test on different RAM
- Test on different Motherboard
- Test on different VGA cards

In Test on different processors – we test for **make –** Intel, AMD processor and also test for **speed** – 3.1GHz, 2.7GHz

In Test on different RAM –we test for **make –** Samsung, Transient and also test for **size** – 1GB, 2GB

In Test on different Motherboard – we test for **make**

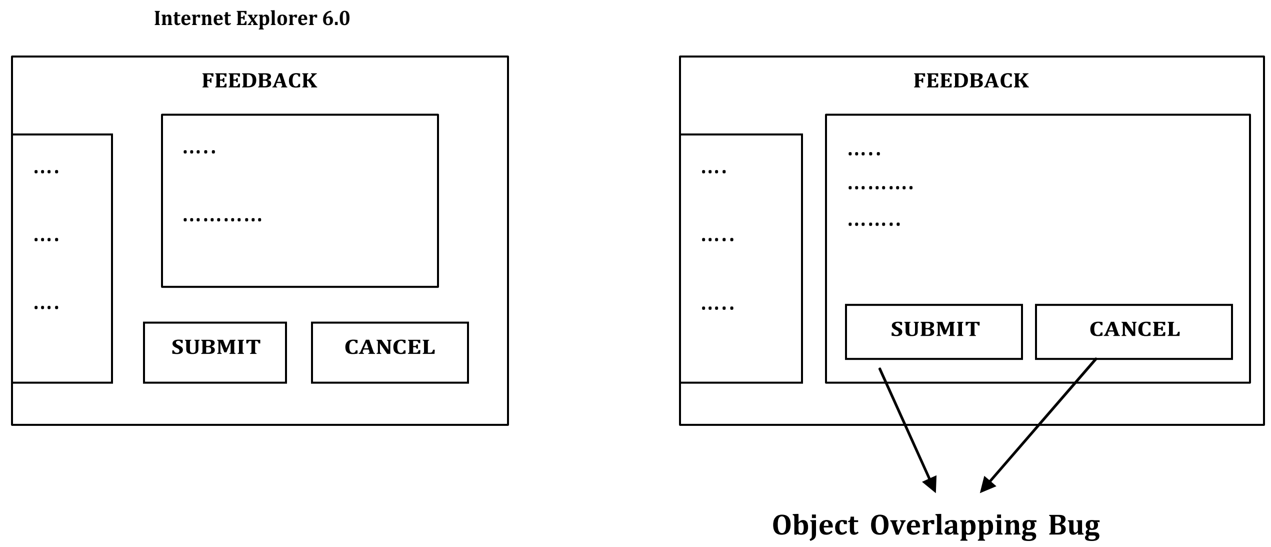
In Test on different VGA cards – we test for **make –** Nvidia, ATI

H/w compatibility test is not done for Web applications – because all programs run on the server and not on the local computers. We do h/w CT only for stand-alone applications.

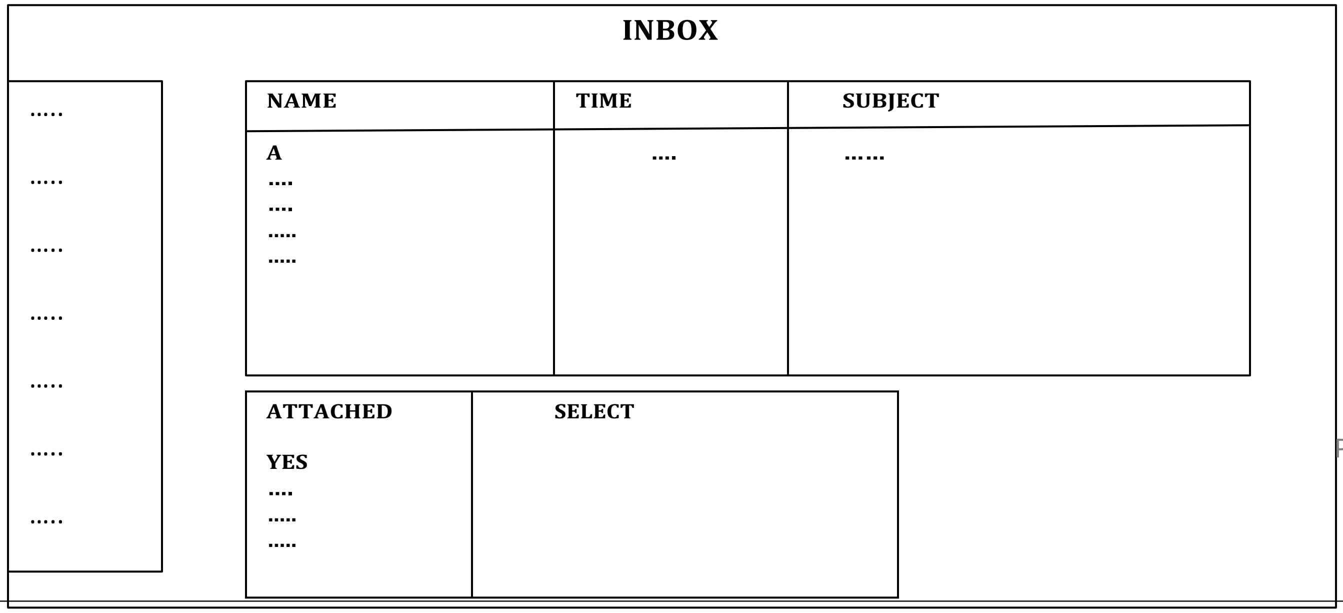
The various Compatibility bugs are,

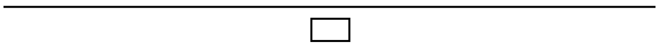
- Scattered content
- Alignment issues
- Broken frames
- Change in look and feel of the application
- Object overlapping
- Change in font size, style and color
- Object overlapping

Object Overlapping



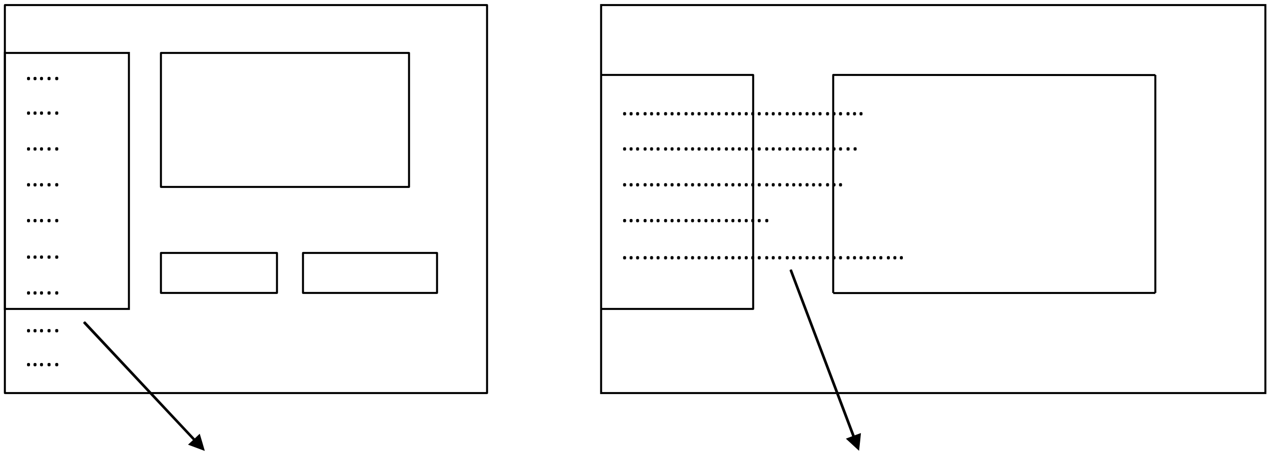
Scattered Content





All the fields (Name, Time, Subject, Attached, Select) must be in the same row. Instead, they are on different lines. This is called scattered content.

Broken Frames



Vertical Broken Frames **Horizontal Broken Frames**

Thus, we should develop the application in such a way that it works on many platforms.

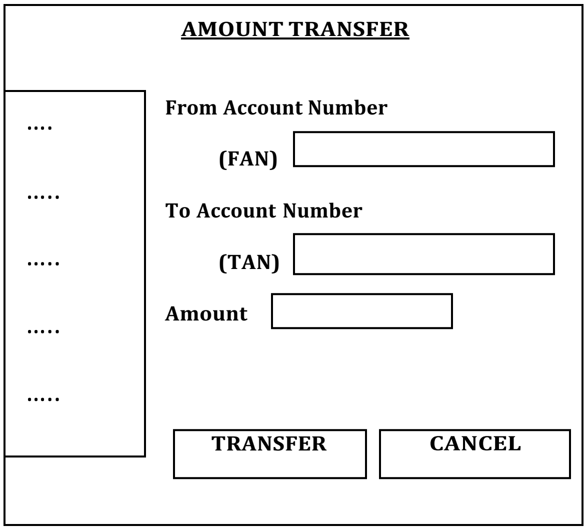
INTERVIEW QUESTIONS

1) How do you test a Web Application? What are the types of test do you do on Web Application?
Ans) Web Application means any dotcom – gmail, yahoo, etc
The types of test are,

- ***Functional Testing***
- ***Integration Testing***
- ***System Testing***
- ***Compatibility Testing – Test in different OS, different browsers, different versions***
- ***Usability Testing – check whether it is user friendly***
- ***Accessibility Testing***
- ***Ad- hoc Testing***
- ***Smoke Testing***
- ***Regression Testing***
- ***Security Testing***
- ***Performance Testing***
- ***Globalization Testing – only if it is developed for multiple languages***

2) Why we do Compatibility testing?
Ans) we might have developed the s/w in 1platform – chances are there that users might use it in different platforms – thus it could lead to defects and bugs – people may stop using the s/w – thus business will be affected and hence we do Compatibility Testing.

TEST CASES



SRS

CBO – Online (CitiBank Online)

1) **Welcome Page**

Log in

Username text field should accept only 6-8characters

Password text field should accept only 4-7characters

2) **Loans**

. . . . .
. . . . .
. . . . .

. . . .
. . . .
. . . . . .
. . . . .
. . . . .

70) **Amount Transfer**

FAN Textfield

Should accept only 10 – digit integer

Should accept only those account numbers generated by Manager

TAN Textfield

Should accept only 10 –digit integer

Should accept only those account numbers generated by Manager

Amount Textfield

Should accept only the integers between100 – 5000

Should not accept more than balance

. . . . . .
. . . . . .
. . . . . .
. . . . . .

The testing quality depends on,

- Mood of the TE
- Testing is not consistent
- Varies from person to person

So, we write test cases.

Test case is a document which covers all possible scenarios to test all the feature(s).
It is a set of input parameters for which the s/w will be tested. The SRS are numbered so that developers and testing team will not miss out on any feature.

When do we write test cases?
Customer gives requirements – developer start developing and they say they need about 4 months to develop this product – during this time, testing team start writing test cases – once it is done, they send it to test lead who reviews it and adds some more scenarios – developers finish developing the product and the product is given for testing – the test engineer then looks at the test cases and starts testing the product – the TE never looks at the requirements while testing the product – thus testing is consistent and does not depend on the mood and quality of the test engineer.

The list of values derived to test the Amount text field is – Blank, -100, hundred, 100, 6000, Rs100, $100, $+?, 0100, Blankspace100, 100.50, 0, 90
When writing test cases, actual result should never be written as the product is still being developed. Only after execution of test cases should the actual result be written.

Why we write test cases?

- **To have better test coverage** – cover all possible scenarios and document it, so that we need not remember all the scenarios
- **To have consistency in <u>test case execution</u>** – seeing the test case and testing the product
- **To avoid training every new engineer on the product** – when an engineer leaves, he leaves with lot of knowledge and scenarios. Those scenarios should be documented, so that new engineer can test with the given scenarios and also write new scenarios.
- **To depend on process rather than on a person**

Let‟s say a test engineer has tested a product during 1st release, 2nd release and has left the company for the 3rd release. As this TE has mastered a module and has tested the application rigorously by deriving many values. If that person is not there for 3rd release, it becomes difficult for the new person – hence all the derived values are documented, so that it can be used in feature.

When developers are developing the 1st product (1st release), TE writes test cases. In the 2nd release, when new features are added, TE writes test cases. In the next release, when features are modified – TE modifies test cases or writes new test cases.

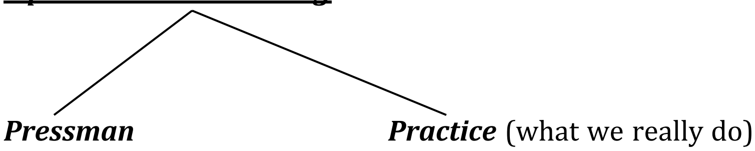
<u>Test Case Design Techniques</u> are,

- Error Guessing
- Equivalence Partitioning
- Boundary Value Analysis (BVA)

<u>Error Guessing</u> :

Guessing the error. If the Amount text field asks for only integers, we enter all other values, like – decimal, special character, negative etc. Check for all the values mentioned above.

Equivalence Partitioning

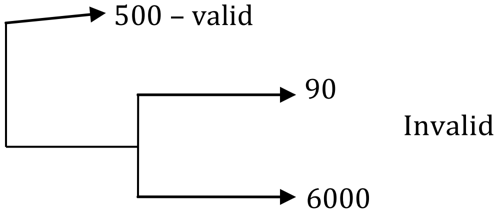


Pressman ***Practice*** (what we really do)

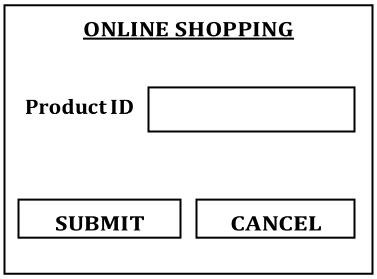
According to Pressman,

1) If the input is a range of values, then design the test cases for 1valid and 2invalid values.

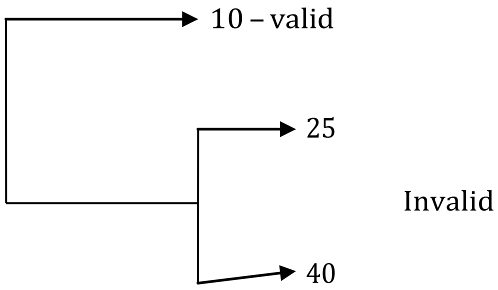
For ex, Amount text field accepts range of values



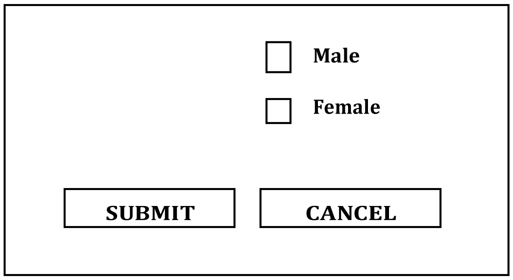
2) If the input is a set of values, then design the test cases for 1valid and 2invalid values.



Printer = 10 ; Scanner = 20 ; Mouse = 30 ;



3) If the input is Boolean, then design the test cases for both true and false values. Ex – checkboxes, radiobuttons etc.



In PRACTICE, we do the following,

Testing the application by deriving the below values,

90 100 1000 2000 3000 4000 5000 6000

Lets see a program. Understand the logic and analyse why we use Practice method,

```
If (amount <100 or >5000)
{
        Error message
}
If (amount between 100 & 2000)
{
        Deduct 2%
}
If (amount > 2000)
{
        Deduct 3%
}
```

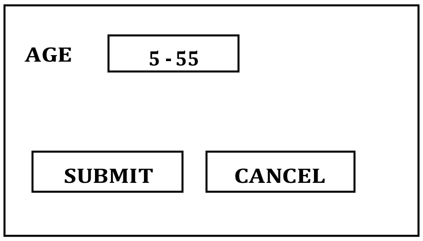
When Pressman techniques are used, the first 2 programs are tested, but if Practice method is used, all these are covered.

It is not necessary that for all applications, practice methodology needs to be used. Sometimes, Pressman is also fine.

But, if the application has any deviation, splits or precision – then we go for Practice method.

If Practice methodology has to be used, it should be – **a)** Case specific **b)** Product specific

c) Number of divisions depends on the precision (2% or 3% deduction)



AGE 5 - 55

Here, Pressman technique is enough to test for Age text field (1 valid and 2invalid)
But, if the Age text field is for insurance (10years and above compulsory and different policies for different age groups) – then we need to use Practice method. Depending on this, divisions of values are done.

BVA – Boundary Value Analysis
If input is a range of values between A – B, then design test case for A, A+1, A-1 and B, B+1, B – 1.

Thus, a number of bugs can be found when applying BVA because developer tends to commit mistakes in this area when writing code.

```
If ( Amount < = 100 )
{
        Throw error
}
If ( Amount > = 5000 )
{
        …..
}
```

If „equals‟ is there, then even 100 value is expected.

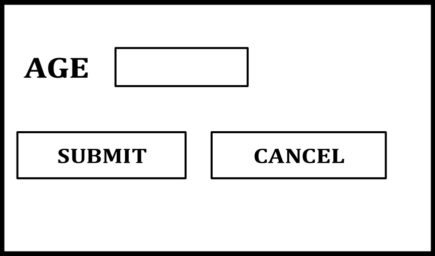
When comparing Equivalence Partitioning and BVA, testing values are repeated – if that is the case, we can neglect Equivalence Partitioning and perform only BVA as it covers all the values.

INTERVIEW QUESTIONS

1) What are test case design techniques ?
Ans) Explain about Error Guessing, Equivalence Partitioning (Pressman only) and BVA.
If they ask more, only then talk about Practice method

2) What are Testing techniques / Black Box techniques / Testing methodologies / Testing types ?
Ans) First talk about WBT and BBT.
If the interviewer is not satisfied with the answer, then immediately start answering about test case design techniques.

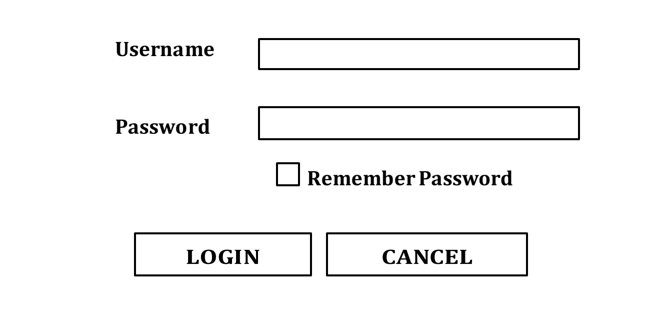
3)



How do you test the above field ?
Ans) Use test case design techniques and derive the values.
I‟ll test the text field by using these test case design techniques – Error Guessing, Equivalence Partitioning and Boundary Value Analysis

Interview Questions (continued)

4)



How do you test the above fields ?

Ans) For any application like this, we must always start testing with end-to-end scenarios.

- *Enter valid username and password – goes to inbox? – logout*
- *Enter valid username and password – remember password – goes to inbox – logout*
- *Open gmail in the browser – username and password should always be there – click on login button – goes to homepage*
- *Change password – now logout and open gmail in the browser – old password should be there – should not login – remove remember password – type in new password – login*
- *Open gmail – type in new password – remember it*
- *Try and open in both Internet Explorer and Mozilla FireFox simultaneously – what happens depends on the requirements*

And then start testing for CANCEL button

After all this, if he asks for more scenarios – then go for functional testing for each text field

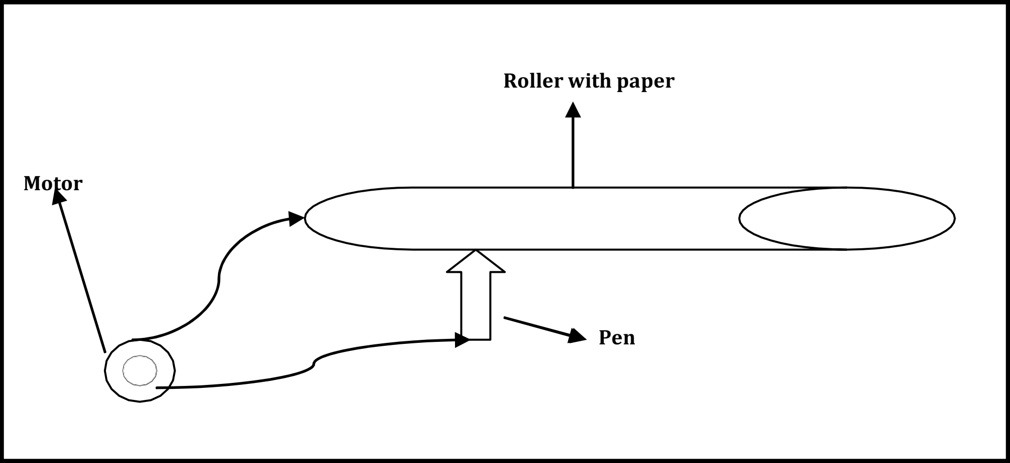
5) How to test a pen ?

Ans) we can do both Manual and Automation testing. 1st let us see how to do Manual Testing,

- *Functional Testing – each part as per requirement – refill, pen body, pen cap, pen size*
- *Integration Testing – Combine pen and cap, and integrate other different parts and see whether they work fine*
- *Smoke Testing – basic functionality – writes or not*
- *Ad-hoc Testing – throw the pen down and start writing, keep it vertically up and write, write on the wall*
- *Usability Testing – whether user friendly or not – whether we can write it for longer periods of time comfortably*
- *Compatibility Testing – different environments, different surfaces, weather conditions – keep it in oven and then write, keep it in freezer and write, try and write on water*
- *Performance Testing – writing speed*
- *Recovery Testing – throw it down and write*
- *Globalization Testing – I18N testing – whether the print on the pen is as per country language and culture – L10N testing – price standard, expiry date format*
- *Reliability Testing – drop it down and write, continuously write and see whether it leaks or not*
- *Accessibility Testing – usable by handicapped people*

Interview Questions (continued …)

Now, we will see how we can do Automation Testing on this pen :-
Take a roller. Now put some sheets of paper on the roller. Connect the roller onto a motor. Connect the pen to the motor and switch on the motor. The pen starts scribbling on the paper. Once the pen has stopped writing, now see how many pages it has written , number of lines on each page, length of each line and multiplying all this – we can get for how many kilometers the pen can write.



FUNCTIONAL Test Cases

We have already shown the template for writing test cases.
The main intention of a TE is to write test cases effectively and also efficiently.
Here, the actual result is written after execution of the test cases and most of the time it would be the same as the expected result. It will be different if only the test step is failed. So, actual result field can be skipped, and we can elaborate about the bug in comments field.
Also, the input field can be removed and these details can be added in description field.

Its not that the above mentioned template is a standard one, the template can vary in each company and also with each application depends on TE and Test Lead. But, for testing 1 application, all TEs should follow a standard template which is formulated.

Test cases should be written in such a way that even if a new TE should/can understand and execute the same.

When entering URL hostname, hostname is mentioned as development team throws the s/w to testing team after say 4months – so we are not sure of the hostname.
Username and password are not mentioned as they keep changing. The user might need to change it every 15days depending upon the application.

When writing functional test cases - we 1st check for which field we write test cases and then elaborate accordingly. If say, amount transfer is the field we are writing FT, then elaborate this is not login feature.
Input is required in FT, if the application is data driven, then i/p column is required else it is time consuming.

Lessons for writing Functional Test Cases

- Before we start writing test case, come up with options and select the best option and then only start writing test case
- In Expected result, use „should be‟ or „must be‟
- Elaborate only those steps on which we have focus. Do not elaborate on all the steps.
- Highlight object names
- Do not hard code the test case. Write a generic test case
- Organize steps properly so that it reduces lot of execution time.

INTEGRATION Test Cases

Something which is covered in FT case should not be written in IT case and something written in IT case should not be written in ST case.
Always start any test case with navigation steps to help new user understand it.

Strategy to write Integration test cases,

- Understand the product
- Identify the possible scenarios
- Prioritize
- Write the test cases according to priority

When TE is writing test cases, the factors he needs to consider are,

- If the test cases are in detail
- Coverage, all test case values or scenarios are depicted
- Think in execution point of view
- Template used to write test cases must be unique

Best test case is when less number of steps are involved, coverage is more, when these test cases are thrown to anyone, they need to understand.

SYSTEM Test Cases

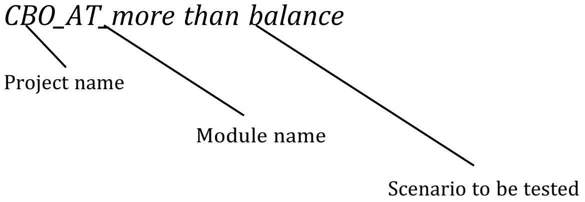
Consider the scenario given in **Pg 35 and Pg 36.** Let us write system test cases for the end-to-end business flow. The basic scenario is shown in the above pag

<u>**Header of Test Case :**</u>

We always fill the body of the test case first before filling up the header of the test case.

1) Test case name :

CBO_AT_more than balance



2) Requirement number :

32.3 Amount Transfer

3) Module name :

Amount Transfer

4) Pre-condition :

Test Engineer should know the balance of user A and user B

Pre-condition is a set of actions or settings that you should have done before executing step number 1.

For ex, If in an application, we are writing test cases for add users, edit users and delete users – the precondition would be – see if user A is added before editing it and deleting it.

5) Test data :

The data we should have before executing step number 1. **Ex –** username, password, account number of users.

The test lead may be give the test data like username or password to test the application. Or the TE may himself generate the username and password.

6) Severity :

It can be *major, minor* or *critical.*

To analyse the severity of the test case, it depends on the header.

Choose the severity according to module. If in the module, there are many features. In that, even if 1 feature is critical, we claim that test case to be critical. It depends on the feature for which we are writing the test case.

In Gmail,

Compose mail -> Critical

Sent items -> Minor

Feedback -> Major

In a banking application,
Amount transfer -> Critical
Feedback -> Minor

We write severity because we can prioritize our execution based on the severity of the feature.

7) Test case type :
It can be functional test cases or integration test cases or system test case or positive or negative or positive and negative test cases.
8) Brief description :
Following test case ensures that Amount Transfer feature is not allowing more than balance.
Test engineer has written test case for a particular feature. If he comes and reads the test cases for a moment, he will not know for what feature has written it. So, this gives a brief description of for what feature test cases are written.

Step number is important. If say, step number 10 is failing – we can document defect report and hence prioritize working and also decide if it‟s a critical bug.

Header format of a Test Case

Test Name : CBO_AT_more than balance
Requirement Number : 70.3 (in SRS document, amount transfer is numbered)
Project Name : Citibank Online
Module Name : Amount Transfer
Severity : critical (depends on feature we are testing)
Pre-condition : ***1)***Test engineer should know the balance amount of user A & B. ***2)*** execution of Balance check test case
Test Case Type : Functional testing, Integration testing, System testing
Test Data : Username and password of user A & B, account number
Brief Description : briefly explains the scenario

<u>Footer of a Test Case</u> :
1) Author : Who wrote this test case
2) Reviewed by :
3) Approved by :
4) approval date :