Experiment 1: Write a "C"program to demonstrate the working of if…else

AIM: To demonstrate the working of "if…else"

Objective: Tounderstandtheworkingofif…else in Various Test cases

```
#include <stdio.h>
voidmain ( )
{
        int a,b;
clrscr( );
        printf("Enter A and B Values:");
        scanf("%d %d",&a, &b);
        if(a>b)
                printf("A is big");
        else
                printf("B is big");
 }
```

Testing:

Test Cases	A	B	Output	Remark
1. a Greater than b	10	5	"A is big"	Correct
2. a Lesser than a	15	25	"B is big"	Correct
3. a equal to b	25	25	"B is big"	Wrong

In the above program the pre condition is not given if a and b are equal

Experiment 2: Write a "C"program to demonstrate the working of "while Loop"

AIM: To demonstrate the working of "while Loop"

Objective: Tounderstandtheworkingof"while"4 loop in Various Test cases

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,a=0,b=1,c=0;
clrscr();
printf("Enter any Number:");
scanf("%d",&n);
  while(c<=n)
  {
printf(" %d,",c);
    a=b;
    b=c;
    c=a+b;
  }
getch();
}
```

Testing:

Test Cases	N Value	Output	Remark
N > 0	50	0,1,1,2,3,5,8,13,21,34	Correct
N=0	0	0	Correct
N<0	-10		Wrong

In the above program the pre condition for negative number checking is not given,if n is given a negative value then nothing will be the produced.

Experiment 3: Write a |"C"program to demonstrate the working of the "do… while" constructs:

AIM: To demonstrate the working of "do..while" construct

Objective: Tounderstandtheworkingof"do while"withdifferentrangeofvaluesandTestCases

```
#include <stdio.h>
voidmain ( )
{
  int i, n=5,j=0;
  clrscr();
  printf("enter a no:");
  scanf("%d",&i);
  do{
        if(i%2==0)
        {
  printf("%d", i);
  printf("is a even no \ti=%d\tj=%d\n",i,j);
  i++;
            j++;
        }
        else
        {
  printf("%d", i);
  printf("is a odd no \ti=%d,\tj=%d\n",i,j);
  i++;
            j++;
  }while(i>0&&j<n);
  getch();
  }
```

Input	Actualoutput
2	2isevennumber 3 is odd number 4isevennumber 5 is odd number 6isevennumber

Testcaseno:1
Testcasename:Positivevalueswithinrange

	Expected output	Actualoutput	Remarks
Input =2	**2isevennumber** **3is oddnumber** **4isevennumber** **5is oddnumber** **6isevennumber**	**2isevennumber** **3isoddnumber** **4isevennumber** **5isoddnumber** **6isevennumber**	**success**

Testcaseno:2
Testcasename:Negativevalueswithinarange

	Expected output	Actualoutput	Remarks
Input= -2	**-2isevennumber** **-3isodd number** **-4isevennumber** **-5isodd number** **-6isevennumber**	**-2isanevennumber**	**fail**

Experiment 4: Write a "C" Program To demonstrate the working of "switch…case"construct

Aim: To demonstrate the working of "switch…case" construct

Objective: Tounderstandtheworkingof"switch…case"withdifferentrangeofvaluesandTestCases

```
#include <stdio.h>
#include <conio.h>
voidmain()
{
        inta,b,choice;
        clrscr();
        printf("1. Sum\n 2.Difference\n 3.Product\n 4.Division\n5.Reminder");
        printf("Enter Your Choice :");
        scanf("%d", &choice);
        printf("Enter A Value:");
        scanf("%d",&a);
        printf("Enter B Value:");
        scanf(" %d",&b);
        switch(choice)
        {
                case1:
                printf("TheSumofA&Bis: %d",a+b);
                break;
                case2:
                printf("TheDifferenceofA&Bis: %d",a-b);
                break;
                case3:
                printf("TheProductofA&Bis: %d",a*b);
        break;
                case4:
                printf("TheDivisionof A by Bis:%d",a/b);
                break;
                case 5:
                printf("The Reminderof A by Bis:%d", a%b);
                break;
                default:
                printf("Invalid Choice");
        }
        getch();
}
```

Output:

Testcases:

Testcaseno:1
Testcasename:Positivevalueswithinrange

Input	Expected Output	Actualoutput	Remarks
EnterYourchoice:1 EnterA Value: 10 Enter B Values:5	**TheSum ofA&B is: 15**	**15**	**Success**
EnterYourchoice:2 EnterA Value: 10 Enter B Values:5	**The Difference of A& B is: 5**	**5**	**Success**
EnterYourchoice:3 EnterA Value: 10 Enter B Values: 5	**The Product of A& B is: 50**	**50**	**Success**
EnterYourchoice:4 EnterA Value: 10 Enter B Values: 5	**The Division of Aby B is: 2**	**2**	**Success**
EnterYourchoice:5 EnterA Value: 10 Enter B Values: 2	**The Reminder of Aby B is: 2**	**2**	**Success**

Testcaseno:2
Testcasename:Out of rangevalues

Input	Expectedoutput	Actualoutput	Remarks
Option:1 a=22222222222222 b=22222222222222	**44444444444444**	**-2**	**Fail**

Testcaseno:3
Testcasename:Dividebyzero

Input	Expected output	Actualoutput	Remarks
Option:4 a=10&b=0		**error**	**fail**

Experiment 5: Write a "C"program to demonstrate the working of "for Loop"

AIM: To demonstrate the working of "for Loop"

Objective: Tounderstandtheworkingof "for loop in Various Test cases

```
#include<stdio.h>
void main()
{
int n,i;
clrscr();
printf("Enter any 10 numbers:");
for(i=0;i<5;i++)
{
scanf("%d",&n);
  if(n%2==0)
printf("%d is evennumber",n);
  else
printf("%d is oddnumber",n);
}
getch();
}
```

Testing:

Test Cases	N Value	Output	Remark
Even Number	5	"5 is odd number"	Correct
Odd Number	10	"10 is even number"	Correct
Fraction Number	12.5	12 is even number 12 is even number 12 is even number	Wrong

In the above program the pre condition for fraction Number is not given, because fraction number is not a even or odd

Experiment 6: Write a Python program to Sort n numbers

AIM: To demonstrate the testing of Sorting program

Objective: TounderstandtheworkingofSortingwith Various range of numbers &Test cases

```
# Program for testing Sort Program
num=[]
n=int(input("Enter How many Numbers to sort:"))
print("Enter Numbers:")
for i in range(1,n+1):
k=input("Enter Number:");
num.append(k);
# Sorting Process
for i in range(0,n):
   for j in range(i+1,n):
  if num[i]>num[j]:
 temp=num[i]
   num[i]=num[j]
 num[j]=temp
# Printing Sorted List
print("Numbers After Sorting")
for i in range(0,n):
    print(num[i])
```

Output:

<u>Test Case 1: Giving number below 10 for sorting</u>

Enter How many Numbers to sort:10
Enter Numbers:
Enter Number:9
Enter Number:8
Enter Number:6
Enter Number:7
Enter Number:1
Enter Number:4

Enter Number:5
Enter Number:2
Enter Number:3
Enter Number:0

Expected Output Numbers After Sorting	Actual Output Numbers After Sorting	Remark
0	0	
1	1	
2	2	
3	3	
4	4	Success
5	5	
6	6	
7	7	
8	8	
9	9	

Test Case 2: Giving number above 9 for sorting

enter How many Numbers to sort10
Enter Numbers:
Enter Number:1
Enter Number:56
Enter Number:23
Enter Number:113
Enter Number:9
Enter Number:63
Enter Number:2
Enter Number:225
Enter Number:72
Enter Number:27

Expected Output Numbers After Sorting	Actual Output Numbers After Sorting
1	1
2	113
9	2
23	225
27	23
56	27
63	56
72	63
113	72
225	9

Remark

Since the Python reads all input as string, in this program the numbers are read as string, it needs to convert the string to numbers, but string to number conversion is not made, while sorting, it performed as string sorting, hence the output is wrong

Experiment 7: Write a Java Program to Test Binary Search process

AIM: To demonstrate the Binary Search program using Java

Objective: TounderstandtheworkingofBinary Search with different Test cases

```
import java.util.*;
public class STBinarySearch {
public static void main(String[] args)
{
int[] arr = new int[100];
   int item, location = -1,n,i;
   Scanner sc = new Scanner(System.in);
System.out.println("Enter how many numbers");
   n = sc.nextInt();
System.out.println("Enter Numbers");
   for(i=0;i<n;i++)
arr[i] = sc.nextInt();
System.out.println("Enter the Number which you want to search");
   item = sc.nextInt();
   location = binarySearch(arr,0,n-1,item);
if(location != -1)
System.out.println("the location of the Number is "+location);
   else
System.out.println("Number not found");
}

public static int binarySearch(int[] a, int beg, int end, int item)
{
   int mid;
if(end >= beg)
   {
      mid = (beg + end)/2;
      if(a[mid] == item)
      {
         return mid+1;
      }
      else if(a[mid] < item)
      {
         return binarySearch(a,mid+1,end,item);
      }
      else
      {
         return binarySearch(a,beg,mid-1,item);
```

```
      }

   }
   return -1;
}
}
```

Output:

Test Case 1: Numbers are given in Sorted order

Input:

Enter How many Numbers:
10
Enter Numbers
69
72
85
96
112
234
333
630
1256
2225
Enter the item which you want to search
112

Expected Output	Actual Output	Remark
The location of the item is 5	The location of the item is 5	Success

Test Case 2: Numbers are given in Random order

Input:

Enter how many Numbers:
10
Enter Numbers
96
52
100
10
69
72
15
1
90
12

Enter the item which you want to search
12

Expected Output	**Expected Output**
The location of the number is 10	**Number not found**

Remark: In this Test case the array list is given in random order. The Binary Search requires sorted Array list, but the program does not contain the code for sorting numbers in the array,hence it is unable to find the number and output is given as "Number not Found"

Experiment 8: Write a Program to Print Factorial of a given number with various Test Cases

AIM: To test the Factorial program with various Test Cases using C++

Objective: Tounderstandtheworkingof Factorial program with different Test Cases

```
//program to calculate Factorial of given number
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,fact=1,n;
clrscr();
printf("Enter any Numbers:");
scanf("%d",&n);
  for(i=1;i<=n;i++)
     fact*=i;
printf("Factorial of %d is %d",n,fact);
getch();
}
```

Output:

Test case 1: When the Positive Number given within the range as input

Input :
Enter any Number:5

Output:

Expected Output	Actual Output	Remark
Factorial of 5 is 120	Factorial of 5 is 120	Success

Test case 2: When the Negative Number given as input

Input :
Enter any Number: -5

Output:

Expected Output	Actual Output
Factorial of -5 is -120	Factorial of -5 is 1

Remark: When the input is given as Negative number, the loop is not at all satisfy the condition from the first iteration itself, hence the output is abnormal

Test case 3: When the input is given out of range

Input :
Enter any Number: 8

Output:

Expected Output	**Actual Output**
Factorial of 8 is 40320	**Factorial of 8 is -25216**

Remark: The input is given out of range, hence the output is abnormal

Experiment 9: To Calculate multiplication of Two Matrices

Aim: A program written in 'C' language for Matrix Multiplication fails"

Objective: Introspect the causes for Matrix Multiplication failure and write down the possible reasons for its failure.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int a[10][10],b[10][10],ar,ac,br,bc,i,j,k,sum;
clrscr();
printf("Enter No. of Rows and Columns of Matrix A:");
scanf("%d%d",&ar,&ac);
printf("Enter No. of Rows and Columns of Matrix B:");
scanf("%d%d",&br,&bc);
printf("Enter Matrix A Elements");
   for(i=0;i<ar;i++)
      for(j=0;j<ac;j++)
scanf("%d",&a[i][j]);
printf("Enter Matrix B Elements");
   for(i=0;i<br;i++)
      for(j=0;j<bc;j++)
scanf("%d",&b[i][j]);
clrscr();

printf("Matrix A\n");
   for(i=0;i<ar;i++)
   {
      for(j=0;j<ac;j++)
printf("%3d",a[i][j]);
printf("\n");
  }
printf("Matrix B\n");
   for(i=0;i<br;i++){
      for(j=0;j<bc;j++)
printf("%3d",b[i][j]);
printf("\n");
  }
printf("Multiplication of A and B Matrices\n");
   for(i=0;i<ar;i++)
   {
      for(j=0;j<bc;j++)
{  sum=0;
        for(k=0;k<ar;k++)
```

```
            sum=sum+a[i][k]*b[k][j];
printf("%3d",sum);
      }
printf("\n");
  }
getch();
}
```

Output:

Testcase1: Equalno. of Rows& Columns of A& B Matrices are given

Input

Enter No. of Rows and Columns in Matrix A: 3
3
Enter No. of Rows and Columns in Matrix B: 3
3

Enterthe Matrix A Elements: 1 Enterthe Matrix B Elements: 1
1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1

Output

Matrix-A**Remark**

1 1 1
1 1 1
1 1 1
Matrix-B
1 1 1
1 1 1**Success**
1 1 1
Multiplication A and B Matrices
3 3 3
3 3 3
3 3 3

Testcase 2: Columns of A Matrix is equal to Rows B Matrix is given

Input

Enter No. of Rows and Columns in Matrix A: 3
2
Enter No. of Rows and Columns in Matrix B: 2
3

Enterthe Matrix A elements: 2	Enterthe Matrix B elements: 1
5	4
7	7
3	6
8	9
9	10

Output	**Remark**
Matrix-A	
2 5	
7 3	
8 9	
Matrix-B	
1 4 7	**Success**
6 9 10	
Multiplication of A and B Matrices	
32 53 64	
25 55 79	
62 113 146	

Testcaseno: 3Columns of Matrix A is not equalto Rowsofmatrix B

Input

Enter No. of Rows and Columns in Matrix A: 3
2
Enter No. of Rows and Columns in Matrix B: 3
3

Enterthe Matrix A elements: 2	Enterthe Matrix B elements: 3
2	3
2	3
2	3
2	3
2	3
	3
	3
	3

Output

Matrix-A
2 2
2 2
2 2
Matrix-B
3 3 3
3 3 3
3 3 3
Multiplication A and B Matrices
12 12 12
12 12 12
12 12 12

Wrong Output

Remark

The rule of Matrix multiplication is that Column of Matrix A should equals to Rows of Matrix B. In this program checking of rows and columns of both matrices are not done. So the output is **Wrong.**

Experiment 10: ATM System with various Case studies

Aim: A program written in 'Java' language for ATM Maintenance

Objective: Maintain ATM Machine with various functionalityand to find Bugs

```
//import required classes and packages
import java.util.Scanner;
//create ATMExample class to implement the ATM functionality
public class ATM_PRG
{
   //Main Program
   public static void main(String args[] )
   {
int macno,mpin,i;
boolean found=false;
      long mamount;
long[ ] acno;
long[ ] pin;
float[ ] balance;
acno = new long[]
{12345,67891,44444,55555,66666,77777,88888,99999,11111,22222};
      pin = new long[ ]
{12345,97891,44444,55555,66666,77777,88888,99999,11111,22222};
      balance=new float[ ]
{100000.2f,55000.2f,64000.2f,15000.2f,64254.2f,145600.2f,85000.2f,12500.2f,6000.2
f,5500.2f};
String[] name = new String[] {"VInayaka","Bhanu Prakash","Pavan
Kumar","Pravallika","Mrudula","Vishvaksen","Mrunalini","Venkatesh","Br
ahmaiah","Vishnu"};
      while(true)
      {
      //create scanner class object to get choice of user
      Scanner sc = new Scanner(System.in);

System.out.println("-----------------------");
System.out.println("     ABC A T M");
System.out.println("-----------------------");
System.out.println("1 Withdraw");
System.out.println("2 Deposits");
System.out.println("3 Balance Enquiry");
System.out.println("4 EXIT");
System.out.print("Select Your Option: ");
      int choice = sc.nextInt();
```

```
System.out.print("Enter Your A/c Number: ");
macno=sc.nextInt();
      found=false;
      for(i=0;i<10;i++)
      {
            if (acno[i]==macno)
            {
System.out.print("Enter PIN: ");
mpin=sc.nextInt();
                if (pin[i]==mpin)
                {
                    found=true;
                    break;
                }
            }
       }
       if (found)
      {
          switch(choice)
         {
               case 1:
System.out.print("Withdrawa");
System.out.print("Enter Amount:");
                    //get the withdrawl money from user
mamount = sc.nextInt();
                   //check whether the balance is greater than or equal to the
withdrawal amount
                    if(balance[i] >= mamount)
                    {
                        //remove the withdrawl amount from the total balance
                        balance[i] = balance[i] - mamount;
System.out.println(" ");
System.out.println("Please collect your money");
                    }
                    else
                   {
                        //show custom error message
System.out.println(" ");
System.out.println("Insufficient Balance");
                    }
System.out.println("");
                   break;

              case 2:
```

```
System.out.print("Deposit");
System.out.print("Enter Deposit Amount:");
mamount = sc.nextInt();
                    //add the deposit amount to the total balanace
                    balance[i] = balance[i] + mamount;
System.out.println(" ");
System.out.println("Your Money has been successfully depsited");
System.out.println("");
                    break;
              case 3:
                    //displaying the total balance of the user
System.out.println(" ");
System.out.println("Balance : "+balance[i]);
System.out.println("");
                    break;
              case 4:
                    //exit from the menu
System.exit(0);
          }
       }
     }
   }
}
```

Experiment 11: Test cases for any known application

Aim: To Write Test cases for known application - Banking

Objective: To Write test cases for Banking Applications with various test cases

Characteristics of a Banking Application

Before you begin testing, it's important to note the standard features expected of any banking application. So that, you can gear your test efforts to achieve these characteristics.

A standard banking application should meet all these characteristics as mentioned below.

- It should support thousands of concurrent user sessions
- A banking application should integrate with other numerous applications like trading accounts, Bill pay utility, credit cards, etc.
- It should process fast and secure transactions
- It should include massive storage system.
- To troubleshoot customer issues, it should have high auditing capability
- It should handle complex business workflows
- Need to support users on multiple platforms (Mac, Linux, Unix, Windows)
- It should support users from multiple locations
- It should support multi-lingual users
- It should support users on various payment systems (VISA, AMEX, MasterCard)
- It should support multiple service sectors (Loans, Retail banking etc.)
- Foolproof disaster management mechanism

Test Phases in Testing Banking Applications

For testing banking applications, different stages of testing include

- **Requirement Analysis:** It is done by business analyst; requirements for a particular banking application are gathered and documented
- **Requirement Review:** Quality analysts, business analysts, and development leads are involved in this task. The requirement gathering document is reviewed at this stage, and cross-checked to ensure that it does not affect the workflow
- **Business Requirements Documentation:** Business requirements documents are prepared by quality analysts in which all reviewed business requirements are covered

- **Database Testing:** It is the most important part of bank application testing. This testing is done to ensure data integrity, data loading, data migration, stored procedures, and functions validation, rules testing, etc.
- **Integration Testing:** Under Integration Testing all components that are developed are integrated and validated
- **Functional Testing:** The usual software testing activities like Test Case preparation, test case review and test case execution is done during this phase
- **Security Testing:** It ensures that the software does not have any security flaws. During test preparation, QA team needs to include both negative as well as positive test scenarios so as to break into the system and report it before any unauthorized individual access it. While to prevent from hacking, the bank should also implement a multi-layer of access validation like a one-time password. For Security Testing, automation tools like IBM AppScan and HPWeb Inspect are used while for Manual Testing tools like Proxy Sniffer, Paros proxy, HTTP watch, etc. are used
- **Usability Testing:** It ensures that differently able people should be able to use the system as normal user. For example, ATM with hearing and Braille facility for disabled
- **User Acceptance Testing:** It is the final stage of testing done by the end users to ensure the compliance of the application with the real world scenario.

Sample Test Case for Net Banking Login Application

Security is a prime for any banking application. Therefore, during test preparation, QA team should include both negative and positive test scenarios in order to sneak into the system and report for any vulnerability before any unauthorized individual get access to it. It not only involves writing negative test cases but may also include destructive testing.

Following are generic test cases to check any banking application

Sample test cases	
For Admin	• Verify Admin login with valid and Invalid data • Verify admin login without data • Verify all admin home links • Verify admin change password with valid and invalid data • Verify admin change password without data • Verify admin change password with existing data • Verify admin logout
For new Branch	• Create a new branch with valid and invalid data • Create a new branch without data

	• Create a new branch with existing branch data • Verify reset and cancel option • Update branch with valid and invalid data • Update branch without data • Update branch with existing branch data • Verify cancel option • Verify branch deletion with and without dependencies • Verify branch search option
For New Role	• Create a new role with valid and invalid data • Create a new role without data • Verify new role with existing data • verify role description and role types • Verify cancel and reset option • Verify role deletion with and without dependency • verify links in role details page
For customer & Visitors	• Verify all visitor or customer links • Verify customers login with valid and invalid data • Verify customers login without data • Verify banker's login without data • Verify banker's login with valid or invalid data
For New users	• Create a new user with valid and invalid data • Create a new user without data • Create a new user with existing branch data • Verify cancel and reset option • Update user with valid and invalid data • Update user with existing data • Verify cancel option • Verify deletion of the user

Challenges in testing Banking domain & their Mitigation

Challenges tester might face during testing banking domain are

Challenge	Mitigation
Getting access to production data and replicating it as test data, for testing is challenging	Ensure that test data meets regulatory compliances requirements and guidelines Maintain the data confidentiality by following techniques like data masking, synthetic test data, testing system integration, etc.
The biggest challenge in testing banking system is during the migration of the system from the old system to the new system like testing of all	Ensure Data Migration Testing is complete Ensure Regression Test cases are

the routines, procedures and plans. Also how the data will be fetched, uploaded and transferred to the new system after migration	executed on old and new systems, and the results match.
There may be the cases where requirements are not documented well and may lead to functional gaps in test plan. Many non-functional requirements are not fully documented, and testers do not know whether to test it or not	The test should participate in the project right from Requirement Analysis phases and should actively review the Business Requirements
The most important point is to check whether the said system follows the desired policies and procedures	Compliance or Regulatory Policies testing must be done
The scope and the timelines increases as banking application are integrated with other application like internet or Mobile banking	Ensure Time budget for Integration Testing is accounted if your banking application has many external interfaces

Conclusion: Banking domain is the most vulnerable area for cyber-theft, and safeguarding the software requires precise testing. It gives a clear idea of what it takes for banking domain testing and how important it is. One must understand that –

- Majority of banking software are developed on **Mainframe** and **Unix**
- Testing helps to lessen possible glitches encounter during software development
- Proper testing and compliance to industry standards, save companies from penalties
- Good practices help develop good results, reputation and more business for companies
- Both manual and automated testing have respective merits and usability

Experiment 12: Create a test plan document for any application (e.g. Library Management System)

Aim: To Create a test plan document for Library Management System

Objective: To study about Create a test plan document for Library Management System

Sample Test Cases for Library Management System

The library manager system under test is – rider library management system. It is very simple and effective system for management of library users and the books. It has the features that you can expect from most of the competitive library systems.

What are the requirements of library management system?

1. User able to register and login.
2. User can search the added books, and check in or out.
3. User can pay the fine or extend the duration of borrowed period.
4. User can change the password and other profile details.
5. User can add the books.
6. User can place the holds and modify existing holds.
7. User can manage the inventory of the books.

These are some of the common features expected from the library management system. So you now have some test scenarios to check for. In addition to these test scenarios, you have GUI based software to check for the bugs, usability and functionality.

Testing Login of Library System

As you can see the screenshot of the application, you're presented with the login screen. This page has feature to add username and password. It also has the links to request username and password if the user has forgotten it.

- Check if the username field accepts valid username and password field accepts valid password.
- Check if the wrong username and valid password allows access to any specific account.
- Check if the valid username and wrong password allows access to any specific account.
- Check if the forgot username link leads to username recovery page.
- Check if the forgot password link leads to the password recovery page.
- Check if the invalid username and password triggers any warning.
- Check if the invalid credentials open the random account.
- Check if the user is logged in, allows you to logout by using the link at the bottom of the application.
- Check if the logout link function as expected.

Testing User Management

- Check if the member can be searched using the firstname or lastname.
- Check if the member transactions are updated.
- Check if the member transactions are shown in the table with sorted column information.
- Check if the user data can be modified if you are admin.
- Check if the new user is possible to add into the system using members tab.
- Check if the password can be reseted in this tab.
- Check if the user can be removed using delete member feature.

Testing Search system of Library

Search system should allow you to search for either member profiles or books. The tabs should let you choose between the user and book. Please check for the options regarding the library management's search system.

- Check if the search function allows searching of books as per title, ISBN, author, genre or all of the criteria.
- Check if the search filter exists as per – books, cds, magazines, videos and software or all of them.
- Check if the search filter has categories feature.
- Check if the search system has the table for listing the search results.
- Check if the search system has either enter button functionality or 'magnify" glass button to perform search query.
- Check if the search system has profile searching filter and categories options.
- Check if the profile search has the results listed as table with profile information containing member info.

Testing Library Resources Inventory

Library has the resources system where you can either search for the books available or you can add or remove the books in the system. This tab should have the resources like books, magazines, courseware, CDs or other resources. Check the screenshot below for more visual information.

- Check if the resources can be searched using the search feature.
- Check if you can add the resource using type, and other categorized information.
- Check if the resources can be searched using barcode or the category title.
- Check if you can modify or edit the resource.
- Check if you can save the resource information.
- Check if you can add copy information for the resource.
- Check if you can add category for the resource.
- Check each field for the limit of the text fields and also valid input for the form.

Experiment 13: Study of any web testing tool

Aim: Study of Web Testing Tool - Selenium Test Tool

Objective: To Study of Selenium Test Tool

Selenium Overview

Selenium Definition: It is a free automation testing suite of tools used for only testing web applications. There is no need to feel disappointed due to the fact that it only helps in testing web applications because various other software have got you covered. There are many tools available for testing desktop and mobile applications, such as IBM's RFI, HP's QPI, Appium, etc. However, the aim of this Selenium tutorial is to make you understand the testing of dynamic web applications and why Selenium is the best for it.

Since Selenium is an open-source tool, there is no licensing cost involved, which is a significant benefit over other testing tools.

History of Selenium

Selenium was the first tool that allowed users to control a browser with the help of any language. It allowed professionals to automate various processes, but it had a set of drawbacks since it was not possible to perform automation testing on certain things with JavaScript. Besides, with web applications getting complex, the restrictions of the tool only started to increase.

Soon, Simon Stewart, from Google, got tired of the limitations of Selenium. He required a testing tool that was capable of communicating with the browser directly, and hence, he came up with WebDriver. A few years later, Selenium merged with WebDriver. This tool allowed professionals to do automation testing by using a single tool, which was much more efficient.

Who Developed Selenium?

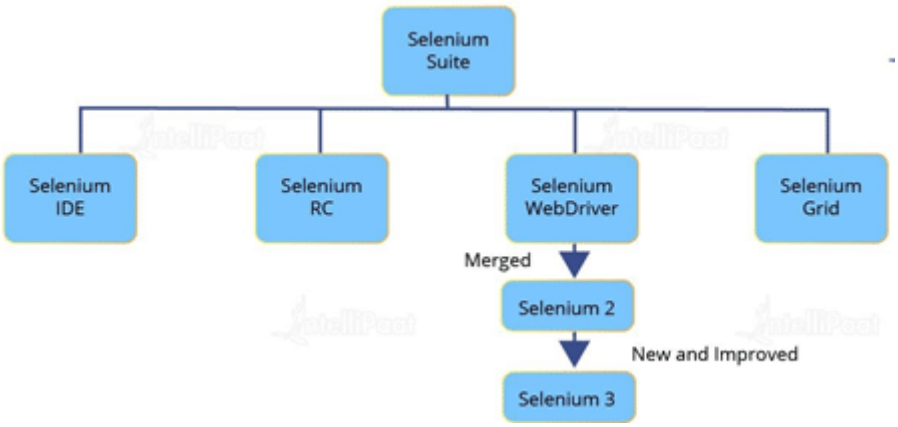
There are numerous developers who built Selenium as it is not a single tool but a collection of several tools. Jason Huggins, an engineer who worked in Thought Works, was the first to realize that the web application that he was working on often required testing. This is when he came up with Selenium. He figured out that it was inefficient to conduct manual testing on applications repeatedly as it took a lot of time and effort. He built a JavaScript program that could control the actions of the browser automatically and called it JavaScript TestRunner. This tool was later called Selenium.

Selenium Tools

Selenium is not simply one tool but a collection of software, each catering to the

different testing needs of an organization. It has four components:

- **Selenium Integrated Development Environment (IDE)**
- **Selenium Remote Control (RC)**
- **Selenium WebDriver**
- **Selenium Grid**



Now, let us discuss each component in detail.

What is Selenium IDE?

IDE stands for Integrated Development Environment, which is a plug-in of Firefox, a browser. This is the simplest framework and, therefore, requires developers to switch to Selenium RC for advanced test cases.

Selenium RC

RC used to be the main testing framework for a long period in a Selenium project. RC was the first tool for automated testing that enabled users to use any programming language of their choice. It offers the following features:

- Designing a test using a more expressive language than Selenese
- Running the test against different browsers, except HTMLUnit, on different operating systems
- Deploying the test across multiple environments using Grid
- Testing the application against a new browser that supports JavaScript
- Testing the web application with complex AJAX-based scenarios

Selenium WebDriver

WebDriver is comparatively better than RC and IDE in various aspects. WebDriver allows professionals to use modern methods to automate the actions of browsers and does not depend on JavaScript for testing. It instead controls the browser easily by communicating with it directly. Professionals can use WebDriver for the following reasons:

- To use a particular programming language in designing the test suit
- To test applications that are rich in AJAX-based functionality
- To execute tests on the HTMLUnit browser
- To create customized test result

Selenium Grid

Grid is used with RC to run parallel tests across various machines and browsers. Listed below are a set of things that professionals can do using Grid:

- Running RC scripts in multiple browsers and operating systems at the same time
- Running a large test suite that will complete within the soonest time possible

Browser Drivers

All browsers have their own drivers that control all the actions that take place within that browser. Once the JSON Wire Protocol provides all the necessary information to the respective browser driver, the driver takes control over the browser so that testers can automatically execute Selenium test scripts. After this, the browser driver uses the HTTP server to send respective feedback in the HTTP protocol.

Listed below are the browser drivers for some of the most popular browsers:

- **OperaDriver**
- **EdgeDriver**
- **FirefoxDriver**
- **SafariDriver**
- **ChromeDriver**

What Types of Testing Can Be Automated With Selenium?

Selenium can help in automating numerous types of testing. Let us discuss the different types of testing briefly.

Unit Testing

This type of testing is done by developers in cases where a bug fix or feature enhancement is made to a part of a website application. Unit testing can be challenging if the professionals need to work on numerous code changes in various application modules.

There are various frameworks in unit testing that are designed for specific languages, such as NUnit for Python, JUnit for Java, etc., to automate the process of unit testing. Selenium is easily compatible with all these frameworks, allowing developers to validate their unit testing across multiple combinations of operating systems and browsers.

Black-box Testing or System Testing

In the case of system testing or black-box testing, the testers are responsible for checking the system's compliance based on particular requirements. They need to test the functionality of the application's module from an end-to-end perspective. In this testing, the testers do not execute any prior tests, and they have no context of the given code.

Besides, a QA engineer needs to draft a script verifying all the functionalities of the respective system. Selenium allows testers to automate the scripts of black-box testing, which helps in saving the bandwidth of professionals to brainstorm and think of more unique scenarios for testing.

Integration Testing

Here, the members of the QA team perform a set of actions to ensure that all the units of the application work properly, both independently and when combined. They ensure that the modules produce the same results even after being integrated. This is done when two or more modules of the application are put together to display one functionality.

With the help of Selenium Automation, testers can easily perform thorough integration testing after every release of the application. This helps in evaluating the functionality and behavior of various modules combined together to build the web application.

End-to-end Testing

QA engineers perform end-to-end testing of web applications from the perspective of end-users. The QA engineers are required to come up with a set of test cases to make sure that the web application functions smoothly at various points. The end-to-end testing process takes a large amount of time especially if the web application has numerous features and pages that need to be tested on multiple devices, browsers, and operating systems.

Selenium Automation makes this process easier as it allows professionals to run parallel testing to automate the browser. Further, Selenium Automation also allows developers to create reports on performance parameters, test case statuses, etc.

Regression Testing

Regression testing allows developers to gauge the end-to-end functionality of the given website application as and when there is a code pushed from one staging environment to another. This is to make sure that the new code does not hamper the present functionality of the web application. This is one of the significant reasons why professionals perform regression testing after every release cycle.

Regression testing also ensures that the current application works easily after the integration of new features. Unfortunately, it is a monotonous process as professionals need to test the entire web application even if the change is minute. Selenium makes it

possible to automate the continuous process of regression testing, saving a lot of time for professionals.

Performance and Load Testing

End users are interested in the performance of the web application and nothing more. Stakeholders set a target for the performance, and QA engineers are responsible for running various tests to check if the application works as per the needs and expectations of the end-users. Rather than running the web application manually on numerous possible combinations of operating systems and browsers, professionals can use Selenium and automate the processes to calculate the performance matrix of the application.