

SOFTWARE TESTING METHODOLOGY

LABORATORY MANUAL

B.TECH (IVYEAR—I SEM)

(2022-2023)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



**MALLAREDDY COLLEGE OF ENGINEERING&TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)**

Recognized under 2(f) and 12 (B) of UGC ACT 1956 Affiliatedto JNTUH, Hyderabad, Approved byAICTE- Accreditedby NBA& NAAC- ‘A’ Grade- ISO9001:2008Certified) Maisammaguda,Dhulapally(Post Via.Hakimpet),Secunderabad– 500100,TelanganaState, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

- To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong which involves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students in to competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce internationally accepted competitive and world class professionals.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Applythe knowledge to build, asses, andanalyze thesoftwareandhardwareaspectsofit .
2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working intteams.
3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutionsto them.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the Starting time); those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
 - i. Student should enter into the laboratorywith:
 - ii. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
 - iii. Laboratory Record updated up to the last session experiments and other utensils (if any) neededin the lab.
 - iv. Proper Dress code and Identity card.
 - v. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
 - vi. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
 - vii. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
 - viii. Computer labs are established with sophisticated and high end branded systems, whichshould be utilized properly.
 - ix. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
 - x. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hourswill be treated seriously and punished appropriately.
 - xi. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Sno	List of programs	Page no
1.	Write programs in C- Language to demonstrate the working of the following a. constructs: i) do... .while ii) while....do iii) if...else iv) switch v) for	1
2.	A program written in C- language for Matrix Multiplication fails Introspect the causes for its failure and write down the possible reasons for its failure.	11
3.	A program written in C- language for Matrix Addition Introspect the causes for its failure and write down the possible reasons for its failure.	14
4.	Take any system (e.g. ATM system) and study its system specifications and report the various bugs.	16
5.	Write the test cases for any known application (e.g. Banking application)	19
6.	Write the test cases for GMAIL	24
7.	Write the test cases for FACEBOOK, TWITTER etc.,	25
8.	Create a test plan document for any application (e.g. Library Management System)	30
9.	Study of any web testing tool (e.g. Selenium)	33
10.	Test case for calculator in windows application	50
11.	BUG TRACKING TOOL Study of bug tracking tool (e.g. Bugzilla).	53
12.	Study of any open source-testing tool (e.g. Test Link)	58

SOFTWARE TESTING

EXPERIMENT: 1

NAME OF THE EXPERIMENT: Write program in „C„ language to demonstrate the working of the following constructs

Objective

To understand the working of do while with different range of values and test cases

```
#include <stdio.h>
void main (){
    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.");
            i++;
        }
        else
            printf("%d", i);
            printf("is a odd no.\n");
            i++;
        j++;
    } while(i>0&&j<n);
    getch();
}
```

Input	Actual output
--------------	----------------------

2	2 is even number3 is odd number 4 is even number5 is odd number 6 is even number
---	--

Test cases:**Test case no: 1****Test case name:** Positive values within range

Input =2	Expected output	Actual output	Remarks
	2 is even number	2 is even number	
	3 is odd number	3 is odd number	success
	4 is even number	4 is even number	
	5 is odd number	5 is odd number	
	6 is even number	6 is even number	

Test case no:2**Test case name:** Negative values within a range

Input = -2	Expected output	Actual output	Remarks
	-2 is even number	-2 is an even number	
	-3 is odd number		fail
	-4 is even number		
	-5 is odd number		
	-6 is even number		

Test case no: 3**Test case name:** Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	12345678912222222213	234567891222222215	fail

//B. Aim: To demonstrate the working of while construct**Objective**

To understand the working of while with different range of values and test cases

```
#include<stdio.h>
#include <conio.h>
void main (){
    int i, n=5,j=1;
    clrscr();
    printf("enter a no");
    scanf("%d",&i);
    while (i>0 && j<n)
    { if(i%2==0)
    {
        printf("%d",i);
        printf("is a even number");
        i++;
        j++;
    }
    else{
        printf("%d",i);
        printf("is a odd number");
        i++;
        j++; }
    }
}
```

Input Actual output

2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number
---	--

Test cases:**Test case no: 1**

Test case name: Positive values within range

Input =2	Expected output	Actual output	Remarks
	2 is even number	2 is even number	
	3 is odd number	3 is odd number	
	4 is even number	4 is even number	
	5 is odd number	5 is odd number	
	6 is even number	6 is even number	success

Test case no:2

Test case name: Negative values within a range

Input = -2	Expected output	Actual output	Remarks
	-2 is even number	-2 is an even number	
	-3 is odd number		fail
	-4 is even number		
	-5 is odd number		
	-6 is even number		

Test case no: 3

Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	12345678912222222213	234567891222222215	fail

//C. Aim: To demonstrate the working of if else construct**Objective:**

To understand the working of if else with different range of values and test cases

```
#include <conio.h>
void main()
{
int i;
for(i=1 ;i<=5;i++)
{
if( i%2==0)
{
printf("number is even no:%d\n",i);
i++;
}
printf("number is odd no:%d\n",i);
}
getch();
}
```

Input	Actual output
i=1	number is odd no:1 number is even no:2 number is odd no:3 number is even no:4 number is odd no:5

Test cases:**Test case no: 1****Test case name:** Positive values within range

Input =2	Expected output	Actual output	Remarks
2	is even number	2 is even number	
3	is odd number	3 is odd number	
4	is even number	4 is even number	
5	is odd number	5 is odd number	
6	is even number	6 is even number	

Test case no:2**Test case name:** Negative values within a range

Input = -2	Expected output	Actual output	Remarks
0	is even number	0 is an even number	
-1	is odd number	-1 is even no	
-2	is even number	-2 is odd no	fail

Test case no: 3 Test case name: Out of range values testing

DEPARTMENT OF CSE

Input	Expected output	Actual output	Remarks
123456789122222222222	12345678912222222213	234567891222222215	fail

// D. To demonstrate the working of switch construct

Objective

To understand the working of switch with different range of values and test cases

```
void main() {  
    int a,b,c;  
    clrscr();  
    printf("1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice");  
    scanf("%d", &i);  
    printf("Enter a,b values");  
    scanf("%d%d",&a,&b);  
    switch(i) {  
        case 1: c=a+b;  
            printf(" The sum of a & b is: %d",c);  
            break;  
        case 2: c=a-b;  
            printf(" The Diff of a & b is: %d",c);  
            break;  
        case 3: c=a*b;  
            printf(" The Mul of a & b is: %d",c);  
            break;  
        case 4: c=a/b;  
            printf(" The Div of a & b is: %d",c);  
            break;  
        default:  
            printf(" Enter your choice");  
            break;  
    }  
}
```

Output:**Input**

Enter Ur choice: 1
Enter a, b Values: 3, 2

Output

The sum of a & b is:5

Enter Ur choice: 2
Enter a, b Values: 3, 2

The diff of a & b is: 1

Enter Ur choice: 3
Enter a, b Values: 3, 2

The Mul of a & b is: 6

Enter Ur choice: 4
Enter a, b Values: 3, 2

The Div of a & b is: 1

Test cases:**Test case no: 1**

Test case name: Positive values within range

Input

Enter Ur choice: 1 Enter
a, b Values: 3, 2

Expected output**Actual output****Remarks**

5

Enter Ur choice: 2 Enter
a, b Values: 3, 2

The sum of a & b is:5

Success

Enter Ur choice: 3 Enter
a, b Values: 3, 2

The diff of a & b is: 1

Enter Ur choice: 4 Enter
a, b Values: 3, 2

The Mul of a & b is: 6

Test case no:2

The Div of a & b is: 1

Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
Option: 1 a= 22222222222222 b=22222222222222	44444444444444	-2	fail

Test case no: 3

Test case name: Divide by zero

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

// E. Aim: To demonstrate working of for construct**Objective:**

To understand the working of for with different range of values and test cases

```
#include <stdio.h>
#include <conio.h>
void main () {
    int i;
    clrscr();
    printf("enter a no");
    scanf("%d",&i);
    for(i=1 ;i<=5;i++) {
        if(i%2==0) {
            printf("%d", i);
            printf(" is a even no");
            i++;
        }
        printf("%d", i);
        printf(" is a odd no"); i++;
    }
    getch();
}
```

Output: Enter

no:5

0 is a even no
 1 is a odd no
 2 is a even no
 3 is a odd no
 4 is a even no
 5 is a odd no

Test cases:**Test case no: 1**

Test case name: Positive values within range

Input =2	Expected output	Actual output	Remarks
	0 is even number	0 is even number	
	1 is odd number	1 is odd number	success
	2 is even number	2 is even number	

Test case no:2

Test case name: Negative values within a range

Input = -2	Expected output	Actual output	Remarks
	0 is even number	0 is an even number	
	-1 is odd number	-1 is even no	fail
	-2 is even number	-2 is odd no	

Test case no: 3

Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	12345678912222222213	234567891222222215	fail

Viva questions:

1. Write a function to swap any two numbers?
2. How can we find out prime numbers from 1 to 50?

EXPERIMENT: 2

NAME OF THE EXPERIMENT:Write a C program that uses functions to perform the following:

- ii) Multiplication of Two Matrices

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
    system("cls");
    printf("enter the number of row=");
    scanf("%d",&r);
    printf("enter the number of column=");
    scanf("%d",&c);
    printf("enter the first matrix element=\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("enter the second matrix element=\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    printf("multiply of the matrix=\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            mul[i][j]=0;
            for(k=0;k<c;k++)
            {

```

```
mul[i][j]+=a[i][k]*b[k][j];  
} }
```

```
}

for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}

return 0;
}
```

output:

enter the number of row=2

enter the number of

column=2enter the first

matrix element=1 2 3 4

enter the second matrix

element=1 2 3 4

multiply of the

matrix=7 10

15 22

FAILURE CASES:

output:

Enter the size of a: 2 3 Enter the

size of b: 2 3

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix —a[] should be equalto number of rows in matrix —b[].

Enter the size of a: p qEnter

the size of b: q s

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix —a[] should be equalto number of rows in matrix —b[], and rows & columns should be integer values.

1. Enter the size of a:

1.5 2Enter the size of b: 2 3

Matrix multiplication is not possible.

Reason to fail: to do multiplication of matrices the number of columns in matrix —al should be equal to number of rows in matrix —bl, and rows & columns should be integer values.

2. Enter the size of a: 350 480

Enter the size of b: 480 620

Matrix multiplication is not possible.

Reason to fail: size of buffer will be not be sufficient to handle this multiplication.

3. Enter the size of a: -1

-2Enter the size of b: -2 3

Matrix multiplication is not possible

Test case no: 1**Test case name:** Equal no.of rows & cols

Input	Expected output	Actual output	Remark
Matrix 1 rows & cols= 3			
3			
Matrix2 rows & cols= 3			
3			
Matrix1:1 1 1			
1 1 1	3 3 3	3 3 3	
1 1 1	3 3 3	3 3 3	Success
	3 3 3	3 3 3	
Matrix2:1 1 1			
1 1 1			
1 1 1			

Test case no:2**Test case name:** Cols of 1st matrix not equal to rows of 2nd matrix

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols= 2 2	Operation Can't be Performed		fail
Matrix2 rows & cols= 3 2			

Test case no: 3**Test case name:** Out of range values testing

Input	Expected output	Actual output	Remarks
Matrix 1 rows & cols= 2 2			
Matrix2 rows & cols= 2 2			

viva questions:

1. syntax for multiplication
2. syntax for matrix multiplication
3. what is the logic for matrix multiplication?

EXPERIMENT: 3**NAME OF THE EXPERIMENT: Write a C program that uses functions to perform the following:**

- ii) Addition of Two Matrices

```
#include <stdio.h>

int main()
{
    int m, n, c, d, first[10][10], second[10][10], sum[10][10];

    printf("Enter the number of rows and columns of
matrix\n");scanf("%d%d", &m, &n);
    printf("Enter the elements of first matrix\n");

    for (c = 0; c < m;
         c++) for (d = 0; d <
n; d++)
        scanf("%d", &first[c][d]);
    printf("Enter the elements of second
matrix\n");for (c = 0; c < m; c++)
    for (d = 0 ; d < n; d++)
        scanf("%d",
&second[c][d]);
    printf("Sum of entered matrices:-\n");for (c = 0; c < m; c++) {
        for (d = 0 ; d < n; d++) {
            sum[c][d] = first[c][d] + second[c][d];
            printf("%d\t", sum[c][d]);
        }
        printf("\n");
    }

    return 0;
}
```

output:

enter the number of row=2

enter the number of column=2

enter the first matrix element=

1 2 3 4

enter the second matrix element=

1 2 3 4

multiply of the matrix=2

4

6 8

FAILURE CASES:

output:

Enter the size of a: 2 3

Enter the size of b: 2 3

Matrix addition is not possible.

Reason to fail: to do addition of matrices the number of columns in matrix —a[] should be equal to the number of rows in matrix —b[].

Enter the size of a: p q

Enter the size of b: q s

Matrix multiplication is not possible.

Reason to fail: to do addition of matrices the number of columns in matrix —a[] should be equal to the number of rows in matrix —b[], and rows & columns should be integer values.

Enter the size of a:

Enter the size of b:

Matrix multiplication is not possible.

Reason to fail: to do addition of matrices the number of columns in matrix —a[] should be equal to the number of rows in matrix —b[], and rows & columns should be integer values.

Enter the size of a:

Enter the size of b:

Matrix multiplication is not possible.

Reason to fail: size of buffer will be not be sufficient to handle this addition.

Enter the size of a: -1 -2

Enter the size of b: -2 3

Matrix addition is not possible.

Reason to fail: to do addition of matrices the number of columns in matrix —a|| should be equal to number of rows in matrix —b||, and rows& columns should be positive integer values.

Test case no: 1

Test case name: Equal no.of rows & cols

EXPERIMENT: 4

NAME OF THE EXPERIMENT: Take any system (e.g. ATM system) and study its systems specifications and report the various bugs.

Program:

```
#include<stdio.h>
#include<conio.h>
unsigned long amount=25000, deposit, withdraw;
int choice, pin, i;
char transaction ='y';
void main()
{
clrscr();
while (pin != 1097)
{
printf("ENTER YOUR PIN NUMBER: ");
scanf("%d", &pin);
if (pin != 1097)
printf("PLEASE ENTER VALID PASSWORD\n");
}
do
{
printf(" Welcome to ATM Service \n");
printf("1. Check Balance\n");
printf("2. Withdraw Cash\n");
printf("3. Deposit Cash\n");
printf("4. Quit\n");
printf("\n\n");
```

DEPARTMENT OF CSE

```
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
printf("\n YOUR BALANCE =Rs.%lu ", amount);

break;
case 2:
printf("\n ENTER THE AMOUNT: ");
scanf("%lu", &withdraw);
if (withdraw % 100 != 0)
{
printf("\n PLEASE ENTER THE AMOUNT IN MULTIPLES OF 100");
}
else if (withdraw >(amount - 1000))
{
printf("\n INSUFFICIENT BALANCE");
}
else
{
amount = amount - withdraw;
printf("\n\n PLEASE COLLECT YOUR CASH");
printf("\n YOUR CURRENT BALANCE =RS.%lu", amount);
}
break;
case 3:
printf("\n ENTER THE AMOUNT: ");
scanf("%lu", &deposit);
amount = amount + deposit;
printf(" YOUR BALANCE =RS.%lu", amount);
break;
case 4:
```

DEPARTMENT OF CSE

```
printf("\n THANK YOU USING OUR ATM SERVICES");

break;

default:

printf("\n INVALID CHOICE");

}

printf("\n\n\n DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): ");

fflush(stdin);

scanf("%c", &transaction);

if (transaction == 'n'|| transaction == 'N')
```

```
i = 1;

} while (!i);

printf("\n\n THANKS FOR USING OUR ATM SERVICE");

getch();

}

output:
```

ENTER YOUR PIN NUMBER: 1097

Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 1

YOUR BALANCE =Rs.25000

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 2

ENTER THE AMOUNT: 20000

PLEASE COLLECT YOUR CASH

DEPARTMENT OF CSE

YOUR CURRENT BALANCE =RS.5000

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 1

YOUR BALANCE =Rs.5000

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 3

ENTER THE AMOUNT: 200

YOUR BALANCE =RS.5200

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 1

YOUR BALANCE =Rs.5200

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): Welcome to ATM Service

1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit

Enter your choice: 4

Features to be tested:

1. Validity of the card.

2. Withdraw Transaction flow of ATM.
3. Authentication of the user's.
4. Dispense the cash from the account.
5. Verify the balance enquiry. 6. Change of PIN number.

Features to be tested:

1. Validity of the card.
2. Withdraw Transaction flow of ATM.
3. Authentication of the user's.
4. Dispense the cash from the account.

Bug-Id	Bug Name
ATM_001	Invalid Card
ATM_002	Invalid PIN
ATM_003	Invalid Account type
ATM_004	Insufficient Balance
ATM_005	Transaction Limit
ATM_006	Day limit
ATM_007	Invalid money denominations
ATM_008	Receipt not printed
ATM_009	PIN change mismatch

BUG REPORT:

Bug Id: ATM_001
Bug Description: Invalid card
Steps to reproduce: 1. Keep valid card in the ATM.
Expected Result: Welcome Screen
Actual Result: Invalid card
Status : Pass/Fail

Bug Id: ATM_002
Bug Description: Invalid PIN entered
Steps to reproduce:
1. Keep a valid card in ATM. 2. Enter the authorized PIN. 3. Menu screen should be displayed.
Expected Result: Menu screen displayed
Actual Result: Invalid PIN screen is displayed
Status : Pass/Fail

Bug Id: ATM_003
Bug Description: Invalid Account type selected.
Steps to reproduce:
1. Enter a valid user PIN number. 2. Select the withdraw option on the main menu. 3. Choose the correct type of account (either savings or current account).
Expected Result: Enter the Amount screen displayed
Actual Result: Invalid Account type screen is displayed.
Status : Pass/Fail

Bug Id: ATM_004
Bug Description: Insufficient Balance
Steps to reproduce:
1. Menu screen should be displayed. 2. Select the withdraw option. 3. Select the correct type of account. 4. Enter the sufficient amount to withdraw from the account. 5. Dispense the cash screen & amount to be deducted from account
Expected Result: Collect the amount screen displayed
Actual Result: Insufficient balance in the account
Status : Pass/Fail

Bug Id: ATM_005

Bug Description: Withdraw Limit per transaction.

Steps to reproduce:

1. Menu screen should be displayed.
2. Select the withdraw option.
3. Select the correct type of account.
4. Enter sufficient amount to withdraw from the account Transaction within the limit.
5. Dispense the cash screen & amount to be deducted from account.

Expected Result: Cash is dispensed and collect the receipt **Actual Result:**

Transaction limit exceeded screen is displayed **Status :** Pass/Fail

Bug Id: ATM_006

Bug Description: Withdraw limit per day

Steps to reproduce:

1. Keep a valid card in ATM.
2. Enter the authorized PIN.
3. Enter the amount to withdraw from the account.
4. Amount enter is over the day limit (>40000)
5. Amount enter is over the day limit and display screen is displayed.

Expected Result: Cash is dispensed and collect the receipt.

Actual Result: Day limit exceeded screen is displayed.

Status : Pass/Fail

Bug Id: ATM_007

Bug Description: Amount enter denominations

Steps to reproduce:

1. Keep a valid card in ATM.
2. Enter the authorized PIN.
3. Enter the amount which should be in multiples of 100.
4. Cash Dispenser screen is displayed.

Expected Result: Collect the amount screen is displayed.

Actual Result: Amount enter not in required denominations.

EXPERIMENT: 5

Write the test cases for any known application (e.g. Banking application)

Banking application

```
#include <stdio.h>
//#include <conio.h>
#include <string.h>
#include <stdlib.h>

// Structure declaration
struct acc_type
{
    char bank_name[20];
    char bank_branch[20];
    char acc_holder_name[30];
    int acc_number;
    char acc_holder_address[100];
    float available_balance;
};

struct acc_type account[20];

/*
printf("The above structure can be declared using
typedef like below");

typedef struct acc_type
{
    char bank_name[20];
    char bank_branch[20];
    char acc_holder_name[30];
    int acc_number;
    char acc_holder_address[100];
    float available_balance;
```

```
 }Acc_detail;

 Acc_detail account[20];

 */

int num_acc;

void Create_new_account();
void Cash_Deposit();
void Cash_withdrawl();
void Account_information();
void Log_out();
void display_options();

/* main program */
int main()
{
    char option;
    char f2f[50] = "http://fresh2refresh.com/";
    num_acc=0;
    while(1)
    {
        printf("\n***** Welcome to Bank Application *****\n");
        printf("\nThis demo program is brought you by %s",f2f);
        display_options();
        printf("Please enter any options (1/2/3/4/5/6) ");
        printf("to continue : ");

        option = getch();
        printf("%c \n", option);
        switch(option)
        {
            case '1': Create_new_account();
```

```
        break;
    case '2': Cash_Deposit();
        break;
    case '3': Cash_withdrawl();
        break;
    case '4': Account_information();
        break;
    case '5': return 0;
    case '6': system("cls");
        break;
    default : system("cls");
        printf("Please enter one of the options");
        printf("(1/2/3/4/5/6) to continue \n ");
        break;
    }
}
return 0;
}
```

/*Function to display available options in this application*/

```
void display_options()
{
    printf("\n1. Create new account \n");
    printf("2. Cash Deposit \n");
    printf("3. Cash withdrawl \n");
    printf("4. Account information \n");
    printf("5. Log out \n");
    printf("6. Clear the screen and display available ");
    printf("options \n\n");
}
```

/* Function to create new account */

```
void Create_new_account()
{
    char bank_name[20];
    char bank_branch[20];
    char acc_holder_name[30];
    int acc_number;
    char acc_holder_address[100];
    float available_balance = 0;
    fflush(stdin);
    printf("\nEnter the bank name      : ");
    scanf("%s", &bank_name);
    printf("\nEnter the bank branch      : ");
    scanf("%s", &bank_branch);
    printf("\nEnter the account holder name   : ");
    scanf("%s", &acc_holder_name);
    printf("\nEnter the account number(1 to 10): ");
    scanf("%d", &acc_number);
    printf("\nEnter the account holder address : ");
    scanf("%s", &acc_holder_address);

    strcpy(account[acc_number-1].bank_name,bank_name);
    strcpy(account[acc_number-1].bank_branch,bank_branch);
    strcpy(account[acc_number-1].acc_holder_name,
    acc_holder_name);
    account[acc_number-1].acc_number=acc_number;
    strcpy(account[acc_number-1].acc_holder_address,
    acc_holder_address);
    account[acc_number-1].available_balance=available_balance;

    printf("\nAccount has been created successfully \n\n");
    printf("Bank name      : %s \n",
    account[acc_number-1].bank_name);
```

```
printf("Bank branch      : %s \n",
account[acc_number-1].bank_branch);
printf("Account holder name   : %s \n",
account[acc_number-1].acc_holder_name);
printf("Account number      : %d \n",
account[acc_number-1].acc_number);
printf("Account holder address : %s \n",
account[acc_number-1].acc_holder_address);
printf("Available balance    : %f \n",
account[acc_number-1].available_balance);
//num_acc++;

}
```

// Displaying account informations

```
void Account_information()
{
register int num_acc = 0;
//if (!strcmp(customer,account[count].name))
while(strlen(account[num_acc].bank_name)>0)
{
printf("\nBank name      : %s \n",
account[num_acc].bank_name);
printf("Bank branch      : %s \n",
account[num_acc].bank_branch);
printf("Account holder name   : %s \n",
account[num_acc].acc_holder_name);
printf("Account number      : %d \n",
account[num_acc].acc_number);
printf("Account holder address : %s \n",
account[num_acc].acc_holder_address);
printf("Available balance    : %f \n\n",
account[num_acc].available_balance);
}
```

DEPARTMENT OF CSE

```
account[num_acc].available_balance);
num_acc++;
}

}

// Function to deposit amount in an account

void Cash_Deposit()
{
auto int acc_no;
float add_money;

printf("Enter account number you want to deposit money:");
scanf("%d",&acc_no);
printf("\nThe current balance for account %d is %f\n",
acc_no, account[acc_no-1].available_balance);
printf("\nEnter money you want to deposit : ");
scanf("%f",&add_money);

while (acc_no==account[acc_no-1].acc_number)
{
    account[acc_no-1].available_balance=
    account[acc_no-1].available_balance+add_money;
    printf("\nThe New balance for account %d is %f\n",
    acc_no, account[acc_no-1].available_balance);
    break;
}acc_no++;
}

// Function to withdraw amount from an account

void Cash_withdrawl()
{
```

```
auto int acc_no;
float withdraw_money;

printf("Enter account number you want to withdraw money:");
scanf("%d",&acc_no);
printf("\nThe current balance for account %d is %f \n",
acc_no, account[acc_no-1].available_balance);
printf("\nEnter money you want to withdraw from account ");
scanf("%f",&withdraw_money);

while (acc_no==account[acc_no-1].acc_number)
{
    account[acc_no-1].available_balance=
    account[acc_no-1].available_balance-withdraw_money;
    printf("\nThe New balance for account %d is %f \n",
    acc_no, account[acc_no-1].available_balance);
    break;
}acc_no++;
}
```

OUTPUT:

```
***** Welcome to Bank Application *****  
This demo program is brought you by http://fresh2refresh.com/  
1. Create new account  
2. Cash Deposit  
3. Cash withdrawl  
4. Account information  
5. Log out  
6. Clear the screen and display available options  
Please enter any options (1/2/3/4/5/6) to continue : 1  
Enter the bank name : ICICI  
Enter the bank branch : Chennai  
Enter the account holder name : Thiagarajan  
Enter the account number<1 to 10>: 1  
Enter the account holder address : Namakkal  
Account has been created successfully  
Bank name : ICICI  
Bank branch : Chennai  
Account holder name : Thiagarajan  
Account number : 1  
Account holder address : Namakkal  
Available balance : 0.000000  
***** Welcome to Bank Application *****  
This demo program is brought you by http://fresh2refresh.com/  
1. Create new account  
2. Cash Deposit  
3. Cash withdrawl  
4. Account information  
5. Log out  
6. Clear the screen and display available options
```

```
Please enter any options (1/2/3/4/5/6) to continue : 2
Enter the account number you want to deposit money : 1

The current balance for account 1 is 0.000000

Enter money you want to deposit to the account : 5000

The New balance for account 1 is 5000.000000

***** Welcome to Bank Application *****

This demo program is brought you by http://fresh2refresh.com/
1. Create new account
2. Cash Deposit
3. Cash withdrawl
4. Account information
5. Log out
6. Clear the screen and display available options

Please enter any options (1/2/3/4/5/6) to continue : 4

Bank name          : ICICI
Bank branch        : Chennai
Account holder name : Thiyagarajan
Account number     : 1
Account holder address : Namakkal
Available balance   : 5000.000000

***** Welcome to Bank Application *****

This demo program is brought you by http://fresh2refresh.com/
1. Create new account
2. Cash Deposit
3. Cash withdrawl
4. Account information
5. Log out
6. Clear the screen and display available options
```

```
Please enter any options <1/2/3/4/5/6> to continue : 3
Enter the account number you want to withdraw money : 1

The current balance for account 1 is 5000.000000

Enter money you want to withdraw from the account 1500

The New balance for account 1 is 3500.000000

***** Welcome to Bank Application *****

This demo program is brought you by http://fresh2refresh.com/
1. Create new account
2. Cash Deposit
3. Cash withdrawl
4. Account information
5. Log out
6. Clear the screen and display available options

Please enter any options <1/2/3/4/5/6> to continue : 4

Bank name : ICICI
Bank branch : Chennai
Account holder name : Thiagarajan
Account number : 1
Account holder address : Namakkal
Available balance : 3500.000000

***** Welcome to Bank Application *****

This demo program is brought you by http://fresh2refresh.com/
1. Create new account
2. Cash Deposit
3. Cash withdrawl
4. Account information
5. Log out
6. Clear the screen and display available options

Please enter any options <1/2/3/4/5/6> to continue : -
```

Test cases for banking applications

Banking applications are considered to be one of the most complex applications in today's software development and testing industry. **What makes Banking application so complex?** What approach should be followed in order to test the complex workflows involved? In this article we will be highlighting different stages and techniques involved in testing Banking applications.

The characteristics of a Banking application are as follows:

Multi tier functionality to support thousands of concurrent user sessions

Large scale Integration , typically a banking application integrates with numerous other applications such as Bill Pay utility and Trading accounts Complex Business workflows Real Time and Batch processing High rate of Transactions per seconds Secure Transactions Robust Reporting section to keep track of day to day transactions Strong Auditing to troubleshoot customer issues Massive storage system Disaster Management.

The above listed ten points are the most important characteristics of a Banking application.

Banking applications have multiple tiers involved in performing an operation. For Example, a

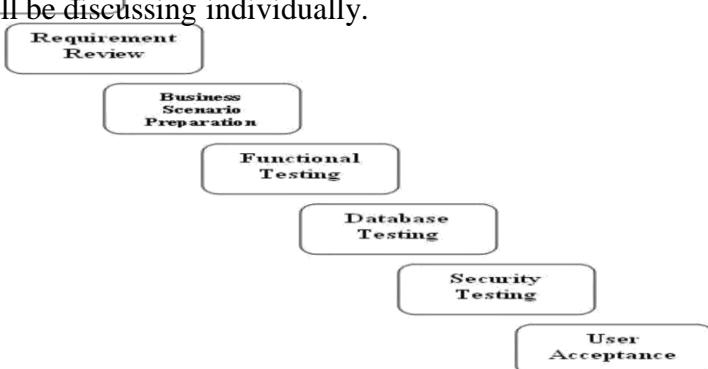
banking application may have:

1. Web Server to interact with end users via Browser
2. Middle Tier to validate the input and output for web server
3. Data Base to store data and procedures
4. Transaction Processor which could be a large capacity Mainframe or any other Legacy system to carry out Trillions of transactions per second.

If we talk about testing banking applications it requires an **end to end testing methodology involving multiple software testing techniques to ensure:**

Total coverage of all banking workflows and Business Requirements
Functional aspect of the application
Security aspect of the application
Data Integrity
Concurrency
User Experience

Typical stages involved in testing Banking Applications are shown in below workflow which we will be discussing individually.



1) Requirement Gathering:

Requirement gathering phase involves documentation of requirements either as Functional Specifications or Use Cases. Requirements are gathered as per customer needs and documented by Banking Experts or Business Analyst. To write requirements on more than one subject experts are involved as banking itself has multiple sub domains and one full fledged banking application will be the integration of all. For Example: A banking application may have separate modules for Transfers, Credit Cards, Reports, Loan Accounts, Bill Payments, Trading Etc.

2) Requirement Review:

The deliverable of Requirement Gathering is reviewed by all the stakeholders such as QA Engineers, Development leads and Peer Business Analysts. They cross check that neither existing business workflows nor new workflows are violated.

3) Business Scenario Preparations:

In this stage QA Engineers derive Business Scenarios from the requirement documents (Functions Specs or Use Cases); Business Scenarios are derived in such a way that all Business Requirements are covered. Business Scenarios are high level scenarios without any detailed steps, further these Business Scenarios are reviewed by Business Analyst to ensure all of Business Requirements are met and its easier for BAs to review high level scenarios than reviewing low level detailed Test Cases.

1) Functional Testing:

In this stage functional testing is performed and the usual software testing activities are performed such as:

Test Case Preparation:

In this stage Test Cases are derived from Business Scenarios, one Business Scenario leads to several positive test cases and negative test cases. Generally tools used during this stage are Microsoft Excel, Test Director or Quality Center.

Test Case Review:

Reviews by peer QA Engineers

Test Case Execution:

Test Case Execution could be either manual or automatic involving tools like QC, QTP or another.

2) Database Testing:

Banking Application involves complex transaction which are performed both at UI level and Database level, Therefore Database testing is as important as functional testing. Database in itself is an entirely separate layer hence it is carried out by database specialists and it uses techniques like

Data loading

Database Migration

Testing DB Schema and Data

typesRules Testing

Testing Stored Procedures and

FunctionsTesting Triggers

Data Integrity

3) Security Testing:

Security Testing is usually the last stage in the testing cycle as completing functional and non functional are entry criteria to commence Security testing. Security testing is one of the major stages in the entire Application testing cycle as this stage ensures that application complies with Federal and Industry standards. Security testing cycle makes sure the application does not have any web vulnerability which may expose sensitive data to an intruder or an attacker and complies with standards like OWASP.

In this stage the major task involves in the whole application scan which is carried out using tools like IBM Appscan or HP WebInspect (2 Most popular tools).

Once the Scan is complete the Scan Report is published out of which False Positives are

filtered out and rest of the vulnerability are reported to Development team for fixing depending on the Severity.

Other **Manual tools for Security Testing** used are: Paros Proxy, Http Watch, Burp Suite, Fortifytools Etc.

Apart from the above stages there might be different stages involved like Integration Testing and Performance Testing.

In today's scenario **majority of Banking Projects are using:** Agile/Scrum, RUP and Continuous Integration methodologies, and Tools packages like Microsoft's VSTS and Rational Tools.

As we mentioned RUP above, RUP stands for Rational Unified Process, which is an iterative software development methodology introduced by IBM which comprises of four phases in which development and testing activities are carried out.

Four phases are:

- i) Inception
- ii) Collaboration
- iii) Construction and
- iv) Transition

RUP widely involves IBM Rational tools.

In this article we discussed **how complex a Banking application could be** and what are the **typical phases involved in testing the application**. Apart from that we also discussed current trends followed by IT industries including software development methodologies and tools.

Test cases for opening bank account

1. Input parameters checking -

Name

-Date of Birth

- Photo -

Address Proof

-Identityproof

-Introducers (if applicable) -PAN card

-Initial deposit

-Whether checkbook / ATM card / Online banking facilities are needed or not -Customer signature

Type of account -

Savings account -

Salary account -Joint

account - Current

account - Secondary

account -RDaccount

-Account for a company

Test cases

-Checking mandatory input parameters -

Checking optional input parameters -Check whether able to create account entity.

-Check whether you are able to deposit an amount in the newly created account (and thus

updating thebalance)

- Check whether you are able to withdraw an amount in the newly created account (after deposit) (andthus updating the balance)
- Check whether company name and its pan number and other details are provided in case of salaryaccount
- Check whether primary account number is provided in case of secondary account

- Check whether company details are provided in cases of company's current account
- Check whether proofs for joint account is provided in case of joint account
- Check whether you are able deposit an account in the name of either of the person in an joint account.

- Check whether you are able withdraw an account in the name of either of the person in an joint account.

- Check whether you are able to maintain zero balance in salary account
- Check whether you are not able to maintain zero balance (or mini balance) in non-salary account.

viva questions

1. Can you explain boundary value analysis?
2. Can you explain equivalence partitioning?
3. Can you explain random/monkey testing?
4. What are semi-random test cases?
5. What is negative and positive testing?
6. How did you define severity ratings in your project?

EXPERIMENT: 6

NAME OF THE EXPERIMENT:

Test Case for Gmail – Inbox Functionality

1. Verify that a newly received email is displayed as highlighted in the Inbox section.
2. Verify that a newly received email has correctly displayed sender email Id or name, mail subject and mail body(trimmed to a single line).
3. Verify that on clicking the newly received email, the user is navigated to email content.
4. Verify that the email contents are correctly displayed with the desired source formatting.
5. Verify that any attachments are attached to the email and are downloadable.
6. Verify that the attachments are scanned for viruses before download.
7. Verify that all the emails marked as read are not highlighted.
8. Verify that all the emails read as well as unread have a mail read time appended at the end on the email list displayed in the inbox section.
9. Verify that count of unread emails is displayed alongside ‘Inbox’ text in the left sidebar of Gmail.
10. Verify that unread email count increases by one on receiving a new email.
11. Verify that unread email count decreases by one on reading an email (marking an email as read).
12. Verify that email recipients in cc are visible to all users.
13. Verify that email recipients in bcc are not visible to the user.
14. Verify that all received emails get piled up in the ‘Inbox’ section and get deleted in cyclic fashion based on the size availability.
15. Verify that email can be received from non-Gmail email IDs like – yahoo, Hotmail etc.

Test Cases for GMail – Compose Mail Functionality

1. Verify that on clicking ‘Compose’ button, a frame to compose a mail gets displayed.
2. Verify that user can enter email IDs in ‘To’, ‘cc’ and ‘bcc’ sections and also user will get

suggestions while typing the emailIds based on the existing emailIds in user's email list.

3. Verify that the user can enter multiple comma-separated emailIds in 'To', 'cc' and 'bcc' sections.
4. Verify that the user can type Subject line in the 'Subject' textbox.
5. Verify that the user can type the email in the email-body section.
6. Verify that users can format mail using editor-options provided like choosing font-family, font-size, bold-italic-underline, etc.
7. Verify that the user can attach file as an attachment to the email.
8. Verify that the user can add images in the email and select the size for the same.
9. Verify that after entering emailIds in either of the 'To', 'cc' and 'bcc' sections, entering Subject line and mail body and clicking 'Send' button, mail gets delivered to intended receivers.
10. Verify that sent mails can be found in 'Sent Mail' sections of the sender.
11. Verify that mail can be sent to non-gmail emailIds also.
12. Verify that all sent emails get piled up in the 'Sent Mail' section and get deleted in cyclic fashion based on the size availability.
13. Verify that the emails composed but not sent remain in the draft section.
14. Verify the maximum number of email recipients that can be entered in 'To', 'cc' and 'bcc' sections.
15. Verify the maximum length of text that can be entered in the 'Subject' textbox.
16. Verify the content limit of text/images that can be entered and successfully delivered as mail body.
17. Verify the maximum size and number of attachment that can be attached with an email.
18. Verify that only the allowed specifications of the attachment can be attached with an email.
19. Verify that if the email is sent without Subject, a pop-up is generated warning user about no subject line. Also, verify that on accepting the pop-up message, the user is able to send the email.

EXPERIMENT: 7

User Timeline Test Cases for Facebook

1. Verify that user can set profile pic uploaded from his or her computer.
2. Verify that user can set profile pic uploaded from mobile.
3. Verify that user can set profile pic from photos present on his facebook account's photo section.
4. Verify that user can set profile from webcam or mobile camera.
5. Verify that user can set cover pic uploaded from his or her computer.
6. Verify that user can set cover pic uploaded from mobile.
7. Verify that user can set cover pic from photos present on his facebook account's photo section.
8. Verify that user can set cover from webcam or mobile camera.
9. Verify that uploading image of unsupported type should lead to error message.
10. Verify that uploading image of size exceeding maximum allowed size should lead to error message.
11. Verify that uploading image of size less than the allowed minimum size should lead to error message.
12. Verify that uploading image of larger dimension than permitted should lead to error message.
13. Verify that uploading image of smaller dimension than permitted should lead to error message.
14. Verify that change in profile pic should get reflected in each post/comment of the user's timeline.
15. Verify that user can add/edit their account information displayed to other users.
16. Verify that users can post text in their timeline and the same gets displayed to their friends.
17. Verify that users can post images in their timeline and the same gets displayed to their friends.
18. Verify that users can post links with or without preview in their timeline and the same gets displayed to their friends.
19. Verify that user can tag friends in their posts.
20. Verify that users can see all the posts in their timeline.
21. Verify that users can see comments, likes and reactions in the posts present in their timeline.
22. Verify that users can post comments, like and react to the posts present in their timeline.

Friends and their Timelines Test Cases for Face book

1. Verify that the user can search for friends in face book's 'Find friends' search functionality.
2. Verify that users can send a friend request to any user by visiting their page.
3. Verify that the user can navigate through their Friend's friend and send a friend request to them.
4. Verify that the user can approve or decline received friend request.
5. Verify that the user can unfriend any existing friend.
6. Verify that users can see the timeline of their friends.

DEPARTMENT OF CSE

7. Verify that users can post text in their friend's timeline.
8. Verify that users can post images in their timeline and the same gets displayed to their friends.
9. Verify that users can post links with or without preview in their friend's timeline.
10. Verify that users can tag friends in their posts on a friend's timeline.
11. Verify that users can see all the posts in their friend's timeline.
12. Verify that users can see comments, likes, and reactions in the posts present in their friend's timeline.
13. Verify that users can post comments, like and react to the posts present in their friend's timeline.

Face book Notification Test Scenarios

1. Verify that users receive different notifications on face book 'Notifications' icon.
2. Verify that users receive different notifications on email or cell phone based on the settings chosen when not logged in to Face book.
3. Verify that users receive a notification when their friend request gets approved.
4. Verify that users receive a notification when they get a friend request.
5. Verify that users receive a notification when they get tagged by someone on posts or comments.
6. Verify that users receive a notification when they get comments, like or reactions on their posts.
7. Verify that users receive notification when someone posts on their timeline.

Test Cases – Login Page

Following is the possible list of functional and non-functional test cases for a login page:

Functional Test Cases:

Sr.	No.	Functional Test Cases	Type- Negative/ Positive TestCase
1		Verify if a user will be able to login with a valid username and valid password.	Positive
2		Verify if a user cannot login with a valid username and an invalid password.	Negative
3		Verify the login page for both, when the field is blank and Submit button is clicked.	Negative
4		Verify the ‘Forgot Password’ functionality.	Positive
5		Verify the messages for invalid login.	Positive
6		Verify the ‘Remember Me’ functionality.	Positive
7		Verify if the data in password field is either visible as asterisk or bullet signs.	Positive
8		Verify if a user is able to login with a new password only after he/she has changed the password.	Positive
9		Verify if the login page allows to log in simultaneously with different credentials in a different browser.	Positive
10		Verify if the ‘Enter’ key of the keyboard is working correctly on the login page.	Positive
Other Test Cases			
11		Verify the time taken to log in with a valid username and password.	Performance & Positive Testing
12		Verify if the font, text color, and color coding of the Login page is as per the standard.	UI Testing & Positive Testing
13		Verify if there is a ‘Cancel’ button available to erase the entered text.	Usability Testing
14		Verify the login page and all its controls in different browsers	Browser Compatibility & Positive Testing.

Non-functional Security Test Cases:



Showentries

Search:

Sr.	Security test cases	Type-
No.		Negative/ Positive TestCase
1	Verify if a user cannot enter the characters more than the specified range in Negative eachfield (Username and Password).	
2	Verify if a user cannot enter the characters more than the specified range in Positive eachfield (Username and Password).	

Sr.	Security test cases	Type-
No.		Negative/ Positive TestCase
3	Verify the login page by pressing 'Back button' of the browser. It should not allow you to enter into the system once you log out.	Negative
4	Verify the timeout functionality of the login session.	Positive
5	Verify if a user should not be allowed to log in with different credentials from the same browser at the same time.	Negative
6	Verify if a user should be able to log in with the same credentials in different browsers at the same time.	Positive
7	Verify the Login page against SQL injection attack.	Negative
8	Verify the implementation of SSL certificate.	Positive

Test Cases for Gmail Login page

The screenshot shows the 'Create your Google Account' page. At the top, there's a 'First name' input field and a 'Last name' input field. Below that, a 'Username' field is followed by '@gmail.com'. A note says 'You can use letters, numbers & periods'. To the right, there's a large blue hexagonal icon containing a white person silhouette, with smaller icons for Google services like YouTube, Photos, and Maps below it. A text overlay says 'One account. All of Google working for you.' Further down, there's a 'Password' field with a note 'Use 8 or more characters with a mix of letters, numbers & symbols', a 'Confirm password' field with a lock icon, and a 'Sign in instead' link. A prominent blue 'Next' button is at the bottom center. At the very bottom, there's a language selection dropdown set to 'English (United States)', and links for 'Help', 'Privacy', and 'Terms'.

Google

Create your Google Account

to continue to Gmail

First name

Last name

Username @gmail.com

You can use letters, numbers & periods

Password

Confirm password

Use 8 or more characters with a mix of letters, numbers & symbols

Sign in instead

Next

English (United States)

Help Privacy Terms

Sr. No. **Test Scenarios**

- 1 Enter the valid email address & click next. Verify if the user gets an option to enter the password.
- 2 Don't enter an email address or phone number & just click the Next button. Verify if the user will get the correct message or if the blank field will get highlighted.
- 3 Enter the invalid email address & click the Next button. Verify if the user will get the correct message.
- 4 Enter an invalid phone number & click the Next button. Verify if the user will get the correct message.
- 5 Verify if a user can log in with a valid email address and password.
- 6 Verify if a user can log in with a valid phone number and password.
- 7 Verify if a user cannot log in with a valid phone number and an invalid password.
- 8 Verify if a user cannot log in with a valid email address and a wrong password.
- 9 Verify the 'Forgot email' functionality.
- 10 Verify the 'Forgot password' functionality.

Test Scenarios for the Sign-up page

1. Verify the messages for each mandatory field.
2. Verify if the user cannot proceed without filling all the mandatory fields.
3. Verify the age of the user when the DOB is selected.
4. Verify if the numbers and special characters are not allowed in the First and Last name.
5. Verify if a user can sign-up successfully with all the mandatory details.
6. Verify if a user can log in with the valid details.
7. Verify if the Password and Confirm Password fields are accepting similar strings only.
8. Verify if the Password field will prompt you for the weak passwords.
9. Verify if duplicate email address will not get assigned.
10. Verify that hints are provided for each field on the form, for the ease of use.

EXPERIMENT: 8

NAME OF THE EXPERIMENT:Test plan document for library application

The Library Management System is an online application for assisting a librarian in managing a book library in a University. The system would provide basic set of features to add/update clients, add/update books, search for books, and manage check-in / checkout processes. Our test group tested the system based on the requirement specification.

INTRODUCTION

This test report is the result for testing in the LMS. It mainly focuses on two problems: what we will test and how we will test.

Result GUI test

Pass criteria: librarians could use this GUI to interface with the backend library database without any difficulties

Result: pass

Database test

Pass criteria: Results of all basic and advanced operations are normal (refer to section 4)
Result: pass

Basic function test

Add astudent

Pass criteria:

- Each customer/student should have following attributes: Student ID/SSN (unique), Name, Address and Phone number.

Result: pass

- The retrieved customer information by viewing customer detail should contain the four attributes. Result: pass

Update/delete student

Pass criteria:

The record would be selected using the student ID Result:

pass

– Updates can be made on full. Items only: Name, Address, Phone number

– Result: passThe record can be deleted if there are no books issued by user.

Result: Partially pass. When no books issued by user, he can be deleted. But when there are books Issued by this user, he was also deleted. It is wrong.

- The updated values would be reflected if the same customer's ID/SSN is called for. Result: pass
- If customer were deleted, it would not appear in further search queries. Result: pass

Add a book

Pass criteria:

- Each book shall have following attributes: Call Number, ISBN, Title, Author name. Result: pass
- The retrieved book information should contain the four attributes. Result: pass

Update/delete book

Pass criteria:

- The book item can be retrieved using the call number. Result: did not pass. Can not retrieve using the call number

The data items which can be updated are: ISBN, Title, Author name Result: pass

The book can be deleted only if no user has issued it. Result: partially pass. When no user has issued it, pass. When there are user having issued it, did not pass.

The updated values would be reflected if the same call number is called for. Result: pass. If book were deleted, it would not appear in further search queries. Result: pass

Search for book

Pass criteria:

The product shall let Librarian query books,, detail information by their ISBN number or Author or Title.

Result: pass

The search results would produce a list of books, which match the search parameters with following Details: Call number, ISBN number, Title, Author Result: pass

The display would also provide the number of copies which is available for issue Result: pass. The display shall provide a means to select one or more rows to a user-list Result: pass

A detailed view of each book should provide information about check-in/check out status, with the borrower,,s information.

DEPARTMENT OF CSE

Result: pass

The search display will be restricted to 20 results per page and there would be means to navigate from sets of search results.

Result: pass

The user can perform multiple searches before finally selecting a set of books for check in or checkout. These should be stored across searches.

Result: pass

A book may have more than one copy. But every copy with the same ISBN number should have same detail information.

Result: pass

The borrower's list should agree with the data in students,

accountResult: pass

Check-in book

Pass criteria:

Librarians can check in a book using its call number

Result: pass

The check-in can be initiated from a previous search operation where user has selected a set of books.

Result: pass

The return date would automatically reflect the current system date.

Result: did not pass.

Any late fees would be computed as difference between due date and return date at rate of 10 cents a day.

Result: did not pass

A book, which has been checked in once, should not be checked in again

Result: pass Check-out book

Pass criteria:

Librarians can check out a book using its call number Result: pass

The checkout can be initiated from a previous search operation where user has selected a set of books.

Result: pass

The student ID who is issuing the book would be

DEPARTMENT OF CSE

enteredResult: pass

The issue date would automatically reflect the current system date.

The due date would automatically be stamped as 5 days from current date.

Result: did not pass

A book, which has been checked out once, should not be checked out again Result: pass A student who has books due should not be allowed to check out any books Result:

did not pass

The max. No of books that can be issued to a customer would be 10. The system should not allow checkout of books beyond this limit.

Result: pass

View book

detail pass

criteria:

This view would display details about a selected book from search operation Result: pass The details to be displayed are: Call number, ISBN, Title, Author, Issue status (In library or checked out), If book is checked out it would display, User ID & Name, Checkout date, Due date Result: for checkout date and due date, did not pass

Books checked in should not display user

summaryResult: pass

Books checked out should display correct user details.Result: pass

View student detail

Pass criteria:

Librarians can select a user record for detailed viewResult: pass

The detail view should show:

a. User name, ID, Address & Phone number
Result: pass

b. The books issued by user with issue date, due date, call number, title
Result: did not pass

c. Late fees & Fines summary

DEPARTMENT OF CSE

and total Result: did not
pass

The display should match existing user
profileResult: pass

The books checked out should have their statuses marked
Result: pass The book search query should show the user id
correctly. Result: pass

Network test

Pass criteria: Results of operations (ping, ftp and ODBC connectivity check) are normal
Result: did not test this item, because no enough machines and no available environment.

Viva questions

1. How to create a test plan document for Library Management System?
2. what is object repository
3. How many test cases can u write 1) File - open dialog box in notepad
please write 5 if software failed in customer environment what we called a)error
b)fault c)defect d)failure 4.What test plan should contains?
5. What is test strategy?
6. Define test Plan?What is the difference between Master Test Plan and Test Plan?

EXPERIMENT: 9

NAME OF THE EXPERIMENT: Study of any web testing tool (e.g. Selenium)

What is Selenium?

JavaScript framework that runs in your web browser Works anywhere JavaScript is supported Hooks for many other languages Java, Ruby, Python Can simulate a user navigating through pages and then assert for specific marks on the pages All you need to really know is HTML to start using it right away

Selenium IDE

Selenium Integrated Development Environment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test.

It provides a Graphical User Interface for recording user actions using Firefox which is used to learn and use Selenium, but it can only be used with Firefox browser as other browsers are not supported.

However, the recorded scripts can be converted into various programming languages supported by Selenium and the scripts can be executed on other browsers as well.

Selenium - IDE Download

Step 1 – Launch Firefox and navigate to the following URL - <http://seleniumhq.org/download/>.

Under the Selenium IDE section, click on the link that shows the current version number as shown below.



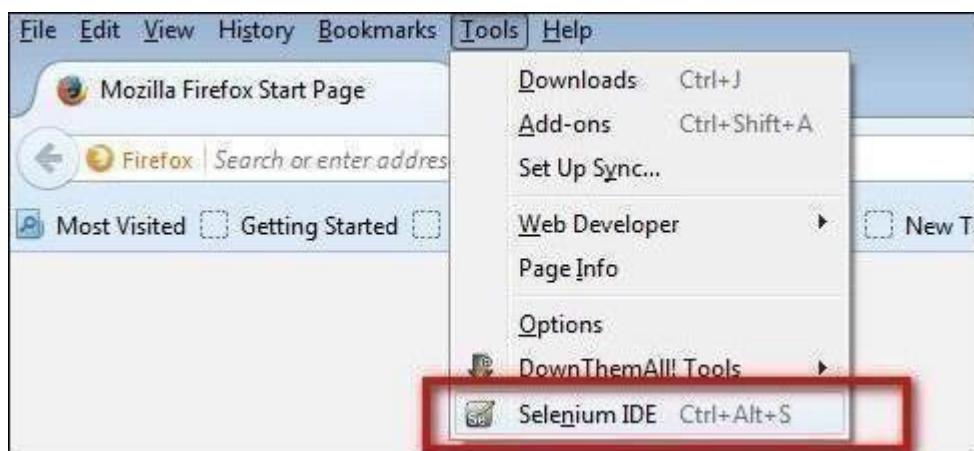
Step 2 – Firefox add-ons notifier pops up with allow and disallow options. User has to allow the installation.



Step 3 – The add-ons installer warns the user about untrusted add-ons. Click 'InstallNow'.



Step 4 – The Selenium IDE can now be accessed by navigating to Tools >> SeleniumIDE.



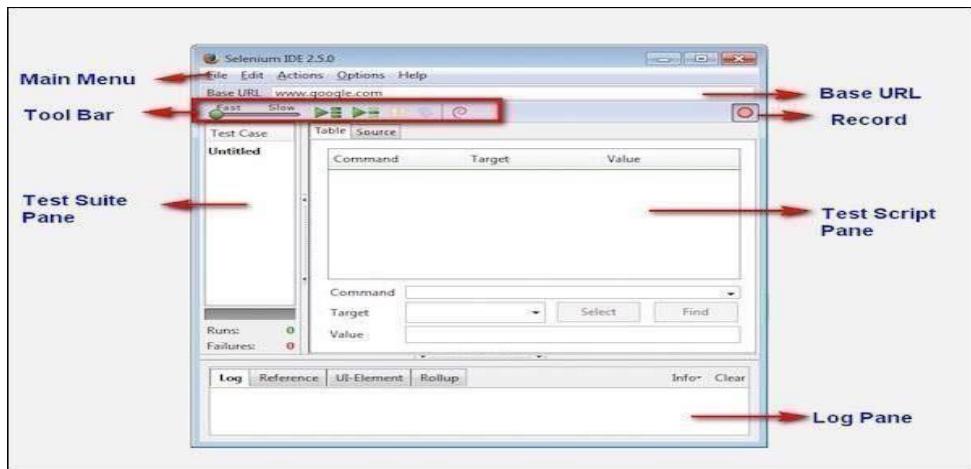
Step 5 – The Selenium IDE can also be accessed directly from the quick access menu bar as shown below.



Selenium IDE Features

This section deals with the features available in Selenium IDE.

The following image shows the features of Selenium IDE with the help of a simple tool-tip.



The features of the record tool bar are explained below.

Control	Control Name	Description
	Speed Control	This helps in controlling the speed of the test case runs.
	Run All	Executes the entire test suite that contains multiple test cases.
	Run	Executes the currently selected test.
	Pause/Resume	Allows user to pause or resume the script execution. Enabled only during the execution.
	Step	Helps user to debug the test by executing only one step of a test case at a time.

	Test Runner Mode	Allows user to execute the test case in a browser loaded with the selenium Core. It is an obsolete functionality that likely to be deprecated.
	Apply Rollup Rules	This features allows repetitive sequences of selenium commands to be grouped into a single action.
	Record	This features helps user to Records the user's browser actions.

Creating Selenium IDE Tests

This section deals with how to create IDE tests using recording feature.

The following steps are involved in creating Selenium tests using IDE –

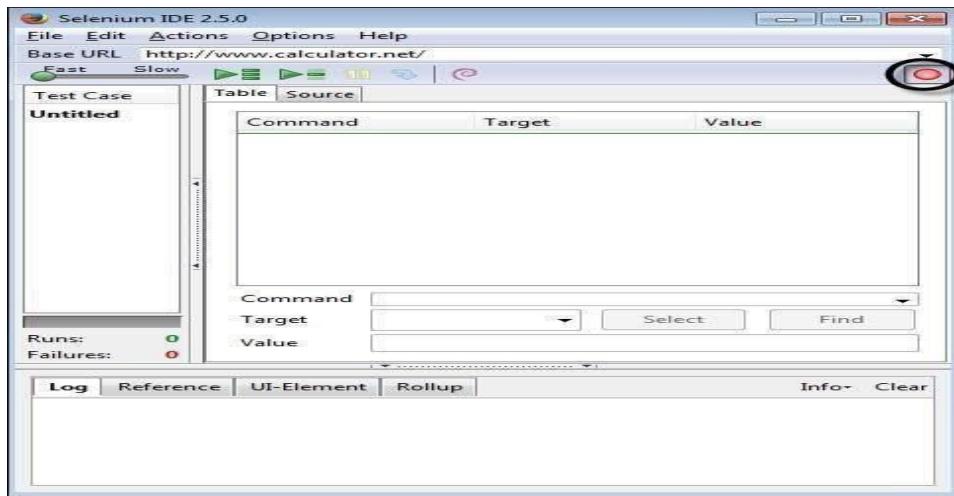
- Recording and adding commands in a test
- Saving the recorded test
- Saving the test suite
- Executing the recorded test

Recording and Adding Commands in a Test

We will use www.ncalculators.com to demonstrate the features of Selenium.

Step 1 – Launch the Firefox browser and navigate to the website - <https://www.ncalculators.com/>

Step 2 – Open Selenium IDE from the Tools menu and press the record button that is on the top-right corner.



Step 3 – Navigate to "Math Calculator" >> "Percent Calculator" >> enter "10" as number1 and 50 as number2 and click "calculate".



Step 4 – User can then insert a checkpoint by right clicking on the webelement and select "Show all available commands" >> select "assert text css=b 5"



Step 5 – The recorded script is generated and the script is displayed as shown below.

Test Case	Table	Source																								
Untitled *	<table border="1"> <thead> <tr> <th>Command</th><th>Target</th><th>Value</th></tr> </thead> <tbody> <tr> <td>open</td><td>/</td><td></td></tr> <tr> <td>clickAndWait</td><td>xpath://a[contains(text(),'Math')][2]</td><td></td></tr> <tr> <td>clickAndWait</td><td>link=Percent Calculator</td><td></td></tr> <tr> <td>type</td><td>id=cpar1</td><td>10</td></tr> <tr> <td>type</td><td>id=cpar2</td><td>50</td></tr> <tr> <td>clickAndWait</td><td>css=input[type="image"]</td><td></td></tr> <tr> <td>assertText</td><td>css=b</td><td>5</td></tr> </tbody> </table>	Command	Target	Value	open	/		clickAndWait	xpath://a[contains(text(),'Math')][2]		clickAndWait	link=Percent Calculator		type	id=cpar1	10	type	id=cpar2	50	clickAndWait	css=input[type="image"]		assertText	css=b	5	
Command	Target	Value																								
open	/																									
clickAndWait	xpath://a[contains(text(),'Math')][2]																									
clickAndWait	link=Percent Calculator																									
type	id=cpar1	10																								
type	id=cpar2	50																								
clickAndWait	css=input[type="image"]																									
assertText	css=b	5																								

Saving the Recorded Test

Step 1 – Save the Test Case by navigating to "File" >> "Save Test" and save the file in the location of your choice. The file is saved as .HTML as default.

The test can also be saved with an extension htm, shtml, and xhtml.

Command	Target	Value
open	/	
clickAndWait	xpath=(//a[contains(text(), 'Math')])[2]	
clickAndWait	link=Percent Calculator	
type	id=cpar1	10
type	id=cpar2	50
clickAndWait	css=input[type="image"]	
assertText	css=b	5

Saving the Test Suite

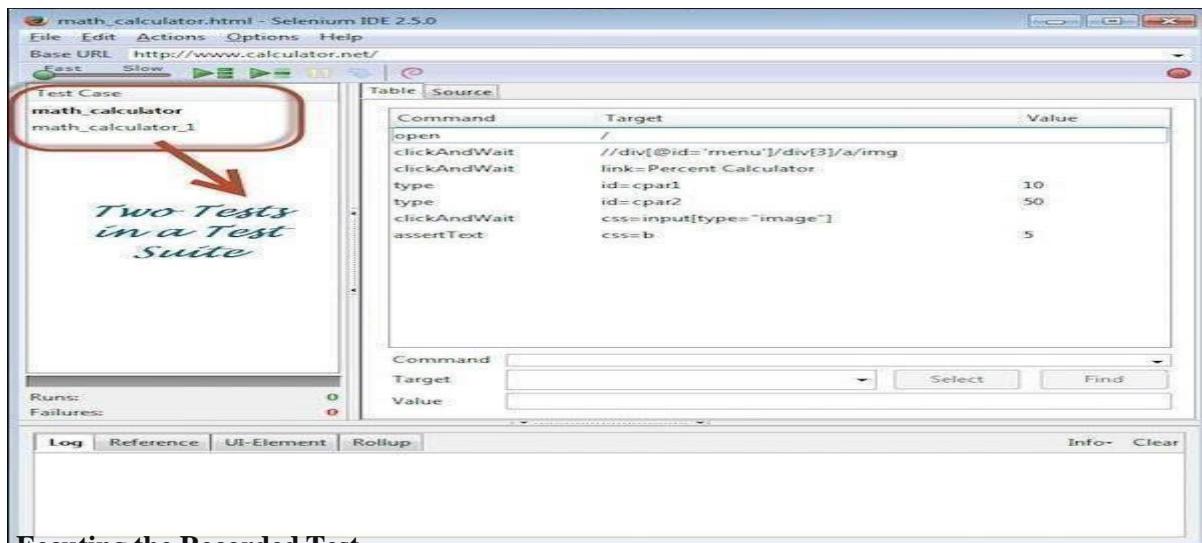
A test suite is a collection of tests that can be executed as a single entity.

Step 1 – Create a test suite by navigating to "File" >> "New Test Suite" as shown below.

- New Test Case** Ctrl+N
- Open...** Ctrl+O
- Save Test Case** Ctrl+S
- Save Test Case As...**
- Export Test Case As...**
- Recent Test Cases**
- Add Test Case...** Ctrl+D
- Properties...**
- New Test Suite**
- Open Test Suite...**
- Save Test Suite**
- Save Test Suite As...**
- Export Test Suite As...**
- Recent Test Suites**
- Close (O)** Ctrl+W

Step 2 – The tests can be recorded one by one by choosing the option "New Test Case" from the "File" Menu.

Step 3 – The individual tests are saved with a name along with saving a "Test Suite".



Executing the Recorded Test

Executing the Recorded Test

The recorded scripts can then be executed either by clicking "Play entire suite" or "Play current test" button in the toolbar.

Step 1 – The Run status can be seen in the status pane that displays the number of tests passed and failed.

Step 2 – Once a step is executed, the user can see the result in the "Log" Pane.

Step 3 – After executing each step, the background of the test step turns "Green" if passed and "Red" if failed as shown below.

Selenium IDE Script Debugging

This section deals with debugging the Selenium IDE script.

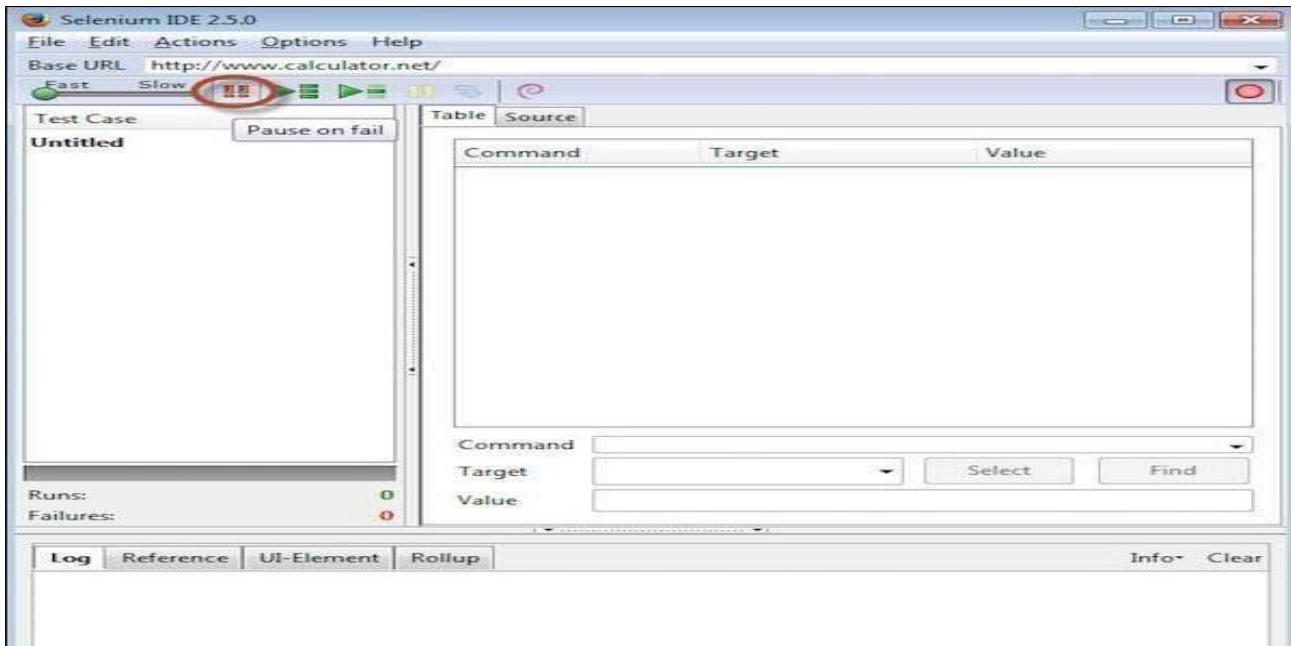
Debugging is the process of finding and fixing errors in the test script. It is a common step in any script development. To make the process more robust, we can make use a plugin "Power Debugger"

for Selenium IDE.

Step 1 – To install Power Debugger for Selenium IDE, navigate to <https://addons.mozilla.org/en-US/firefox/addon/power-debugger-selenium-ide/> and click "Add to Firefox" as shown below.



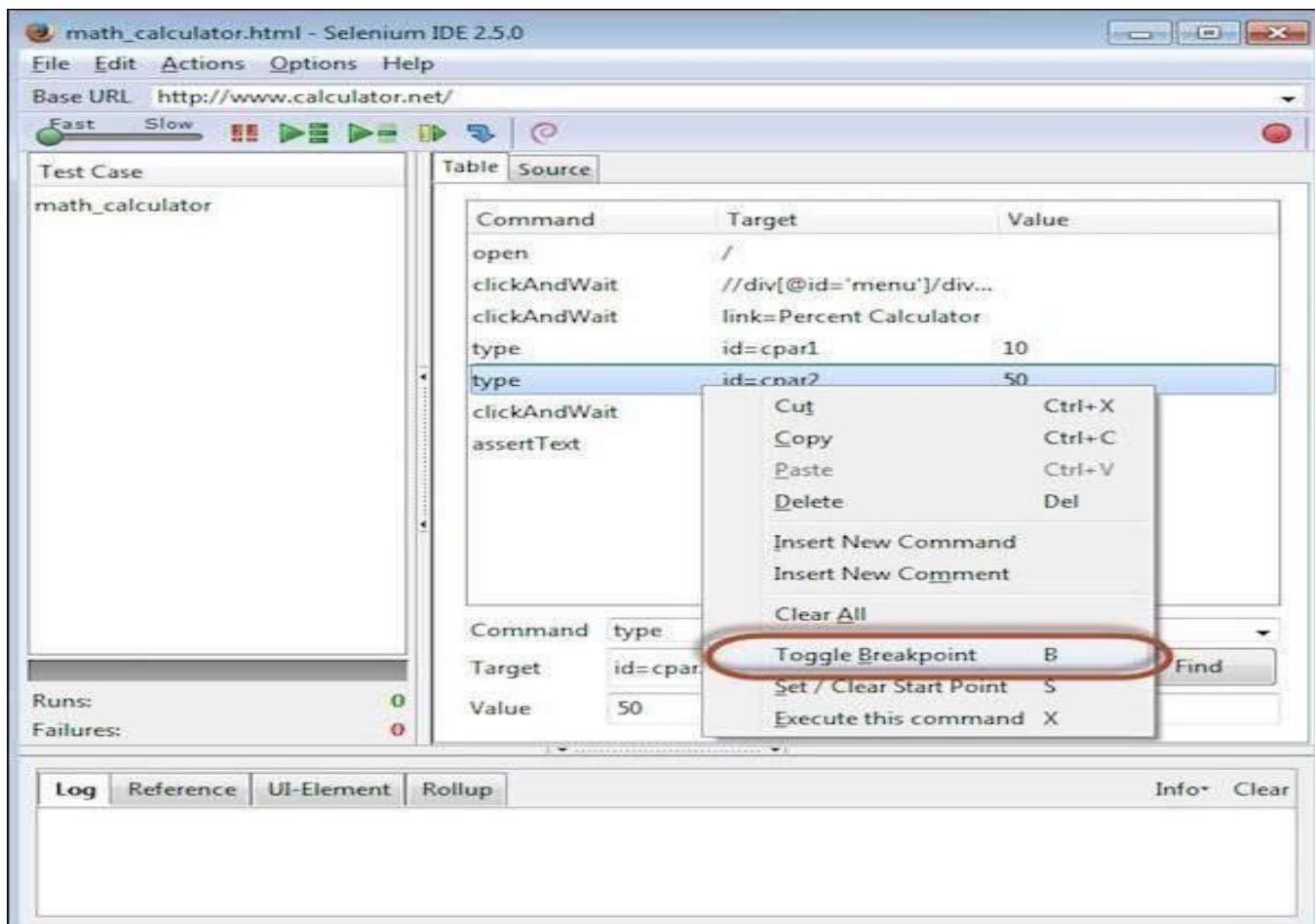
Step 2 – Now launch 'Selenium IDE' and you will notice a new icon, "Pause on Fail" on recording toolbar as shown below. Click it to turn it ON. Upon clicking again, it would be turned "OFF".



Step 3 – Users can turn "pause on fail" on or off any time even when the test is running.

Step 4 – Once the test case pauses due to a failed step, you can use the resume/step buttons to continue the test execution. The execution will **NOT** be paused if the failure is on the last command of any test case.

Step 5 – We can also use breakpoints to understand what exactly happens during the step. To insert a breakpoint on a particular step, "Right Click" and select "Toggle Breakpoint" from the context-sensitive menu.



Step 6 – Upon inserting the breakpoint, the particular step is displayed with a pause icon as shown below.

Command	Target	Value
open	/	
clickAndWait	//div[@id='menu']/div[2]/a/img	
clickAndWait	link=Percent Calculator	
type	id=cpar1	10
type	id=cpar2	50
clickAndWait	css=input[type="image"]	
assertText	css=b	5

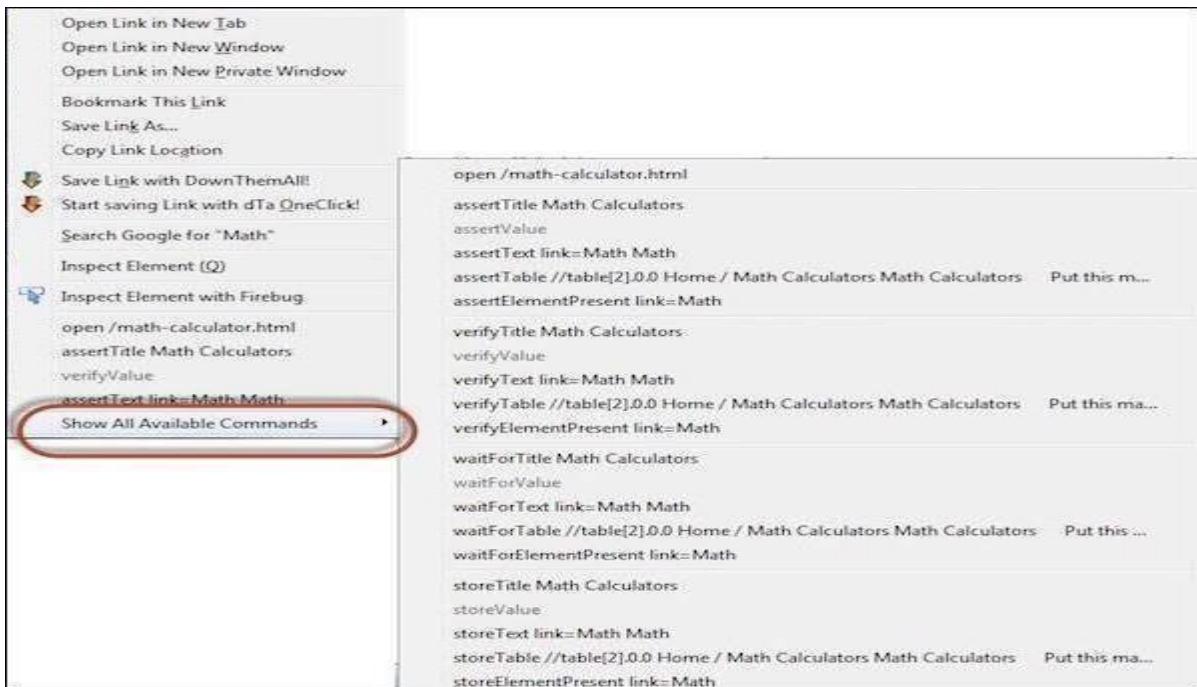
Step 7 – When we execute the script, the script execution is paused where the breakpoint is inserted. This will help the user to evaluate the value/presence of an element when the execution is in progress.

Inserting Verification Points

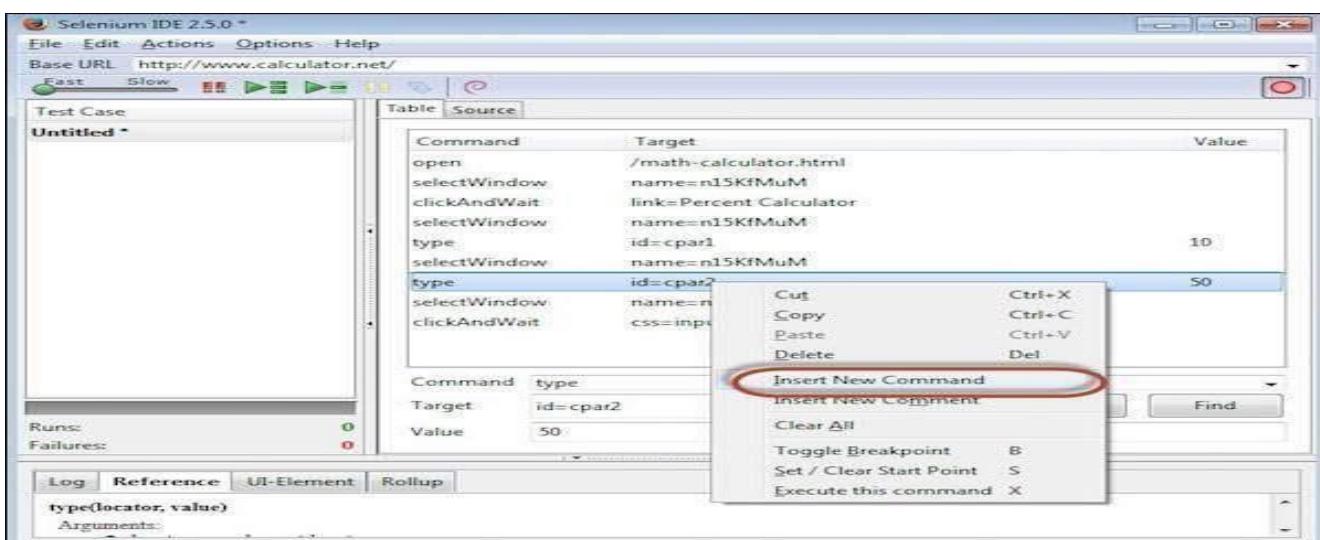
This section describes how to insert verification points in Selenium IDE.

The test cases that we develop also need to check the properties of a web page. It requires assert and verify commands. There are two ways to insert verification points into the script.

To insert a verification point in recording mode, "Right click" on the element and choose "Show all Available Commands" as shown below.

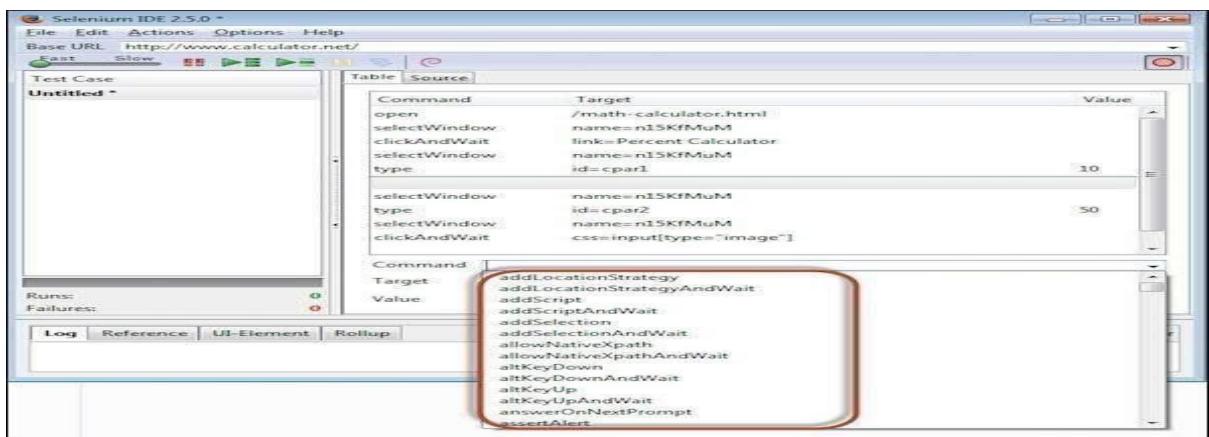


We can also insert a command by performing a "Right-Click" and choosing "Insert New Command".



After inserting a new command, click 'Command' dropdown and select appropriate verification point from the available list of commands as shown below.

Given below are the mostly used verification commands that help us check if a particular step has passed or failed.



- verifyElementPresent
- assertElementPresent
- verifyElementNotPresent
- assertElementNotPresent
- verifyText
- assertText
- verifyAttribute
- assertAttribute
- verifyChecked
- assertChecked

- verifyAlert
- assertAlert
- verifyTitle
- assertTitle

SynchronizationPoints

During script execution, the application might respond based on server load, hence it is required for the application and script to be in sync. Given below are few commands that we can use to ensure that the script and application are in sync.

- waitForAlertNotPresent
- waitForAlertPresent
- waitForElementPresent
- waitForElementNotPresent
- waitForTextPresent
- waitForTextNotPresent
- waitForPageToLoad
- waitForFrameToLoad

Selenium Pattern Matching

- This section deals with how to work with regular expressions using IDE.

Like locators, patterns are a type of parameter frequently used by Selenium. It allows users to describe patterns with the help of special characters. Many a time, the text that we would like to verify are dynamic; in that case, pattern matching is very useful.

Pattern matching is used with all the verification point commands - verifyTextPresent, verifyTitle, verifyAlert, assertConfirmation, verifyText, and verifyPrompt.

There are three ways to define a pattern –

- globbing
- regular expressions, and
- exact patterns.

Globbing

Most techies who have used file matching patterns in Linux or Windows while searching for a certain file type like *.doc or *.jpg. would be familiar with term "globbing"

Globbing in Selenium supports only three special characters: *, ?, and [].

- * – matches any number of characters.

DEPARTMENT OF CSE

- ? – matches a single character.
- [] – called a character class, lets you match any single character found within the brackets.
[0-9] matches any digit.

To specify a glob in a Selenium command, prefix the pattern with the keyword 'glob:'. For example, if you would like to search for the texts "tax year 2013" or "tax year 2014", then you can use the glob "tax year *" as shown below.

However the usage of "glob:" is optional while specifying a text pattern because globbing patterns are the default in Selenium.

Command	Target	Value
clickAndWait	link = search	
verifyTextPresent	glob: tax year *	

Exact Patterns

Patterns with the prefix 'exact:' will match the given text as it is. Let us say, the user wants an exact match with the value string, i.e., without the glob operator doing its work, one can use the 'exact' pattern as shown below. In this example the operator '*' will work as a normal character rather than a pattern-matching wildcard character.

Command	Target	Value
clickAndWait	link = search	
verifyValue	exact: *.doc	

Regular Expressions

Regular expressions are the most useful among the pattern matching techniques available. Selenium supports the complete set of regular expression patterns that Javascript supports. Hence the users are no longer limited by *, ? and [] globbing patterns.

To use RegEx patterns, we need to prefix with either "regexp:" or "regexpi:". The prefix "regexpi" is case-insensitive. The glob: and the exact: patterns are the subsets of the Regular Expression patterns. Everything that is done with glob: or exact: can be accomplished with the help of RegExp.

Example

For example, the following will test if an input field with the id 'name' contains the string 'tax year', 'Tax Year', or 'tax Year'.

Command	Target	Value
clickAndWait	link = search	
verifyValue	id = name	regexp:[Tt]ax ([Yy]ear)

Selenium User Extensions

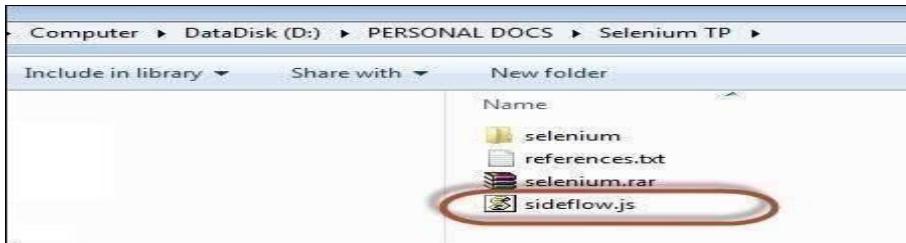
The Java script that allows users to customize or add new functionality.

It is easy to extend Selenium IDE by adding customized actions, assertions, and locator-strategies. It is done with the help of JavaScript by adding methods to the Selenium object prototype. On startup, Selenium will automatically look through the methods on these prototypes, using name patterns to recognize which ones are actions, assertions, and locators.

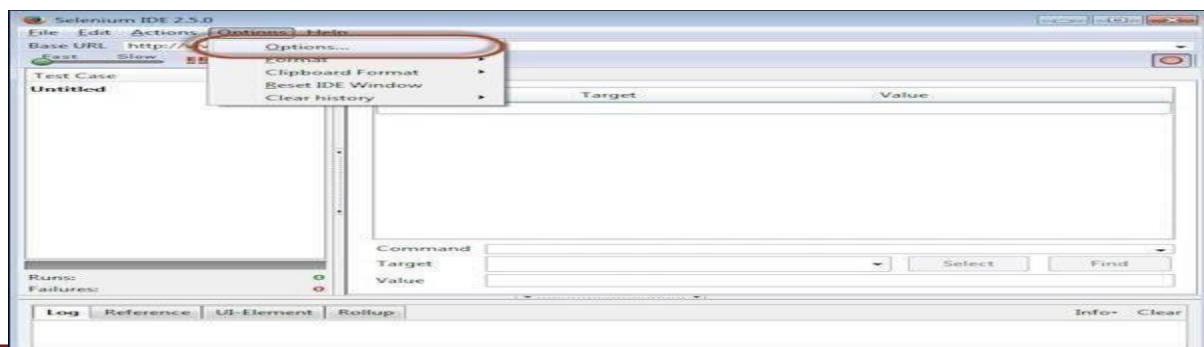
Let us add a 'while' Loop in Selenium IDE with the help of JavaScript.

Step 1 – To add the js file, first navigate

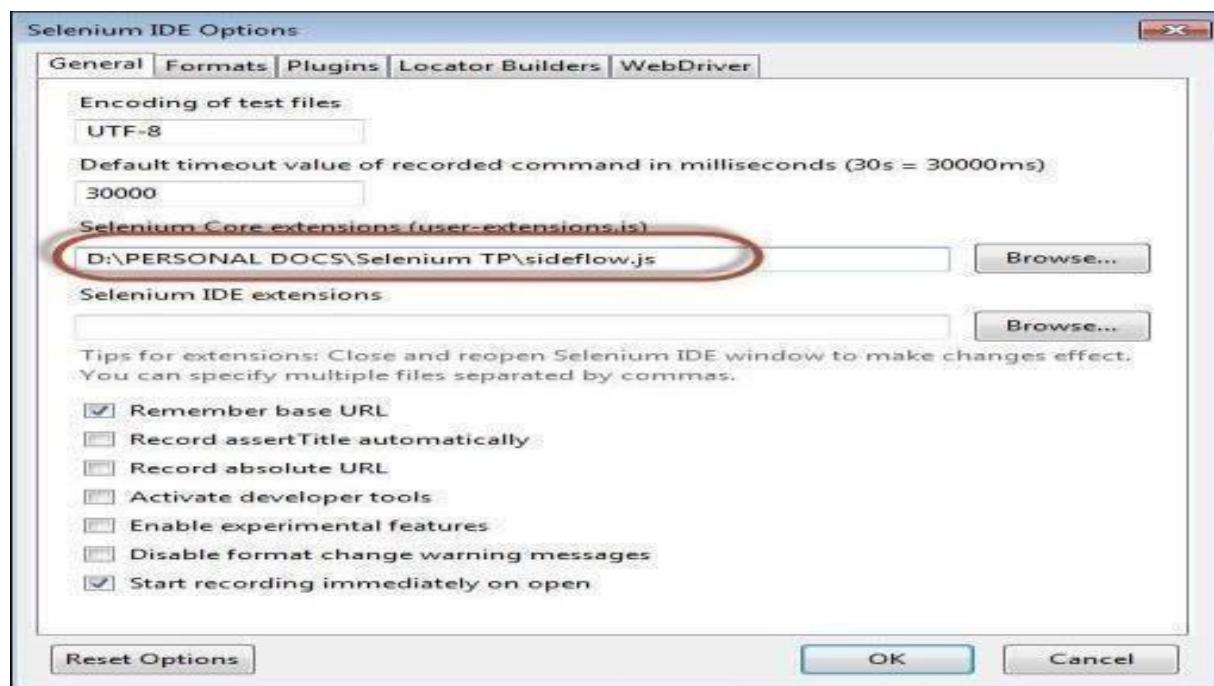
to <https://github.com/darrenderidder/sideflow/blob/master/sideflow.js> and copy the script and placesave it as 'sideflow.js' in your local folder as shown below.



Step 2 – Now launch 'Selenium IDE' and navigate to "Options" >> "Options" as shown below.



Step 3 – Click the 'Browse' button under 'Selenium Core Extensions' area and point to the js file that we have saved in Step 1.



Step 4 – Restart Selenium IDE.

Step 5 – Now you will have access to a few more commands such as "Label", "While" etc.

Step 6 – Now we will be able to create a While loop within Selenium IDE and it will execute as shown below.

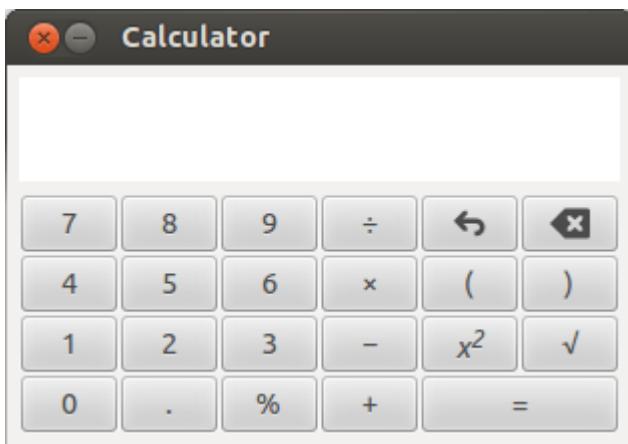
Viva questions:

1. What is Selenium?
2. What are the different Selenium components?
3. What are the testing types that can be supported by Selenium?
4. What are the limitations of Selenium?
5. What is Selenese?

EXPERIMENT: 10

NAME OF THE EXPERIMENT: Test case for calculator in windows application

I am taking WINDOWS calculator as an example for my test case. You can use the belowcalculator or use the one that comes with your operating system.



Basic Calculator Test Cases

Basic Operational Tests

Write the test cases based on the following functions and scenarios.

- Check the calculator if it starts by on button. If it is software based calculator then check if it starts via specific means like from searching for calculator in search bar and then executing application. Or by accessing menu item in the Windows.
- Check if the calculator window maximizes to certain window size.
- Check the if the calculator closes when the close button is pressed or if the exit menu is clicked from file
 - > exit option.
- Check if the help document is accessed from Help > Documentation.
- Check if the calculator allows copy and paste functionality.
- Check if the calculator has any specific preferences.
- Check if all the numbers are working (0 to 9)
- Check if the arithmetic keys (+, -, *, %, /) are working.

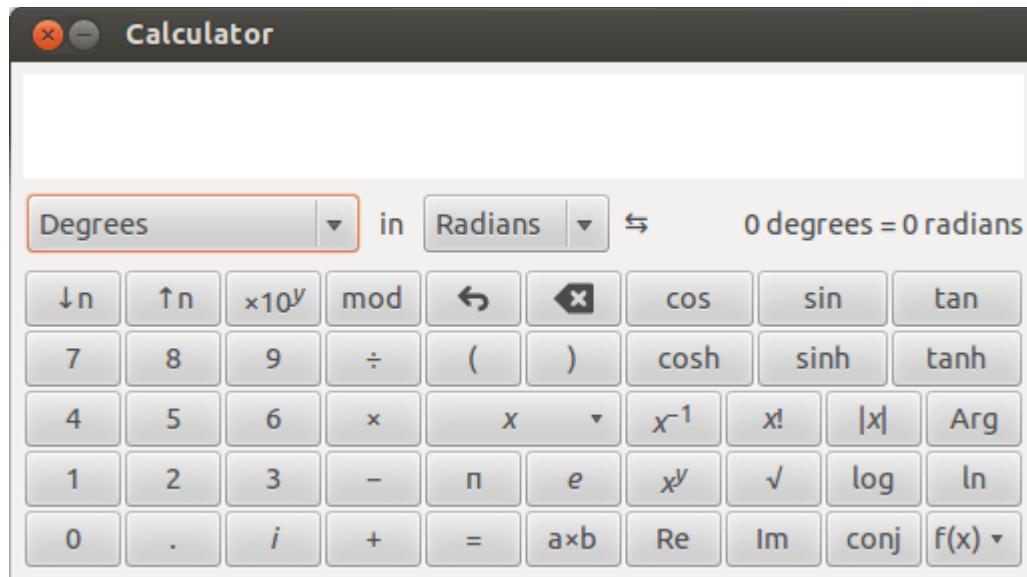
- Check if the clear key is working.
- Check if the brackets keys are working.
- Check if the sum or equal key is working.
- Check if the square and square root key is working.

Functionality Test Cases

- Check the addition of two integer numbers.
- Check the addition of two negative numbers.
- Check the addition of one positive and one negative number.
- Check the subtraction of two integer numbers.
- Check the subtraction of two negative numbers.
- Check the subtraction of one negative and one positive number.
- Check the multiplication of two integer numbers.
- Check the multiplication of two negative numbers.
- Check the multiplication of one negative and one positive number.
- Check the division of two integer numbers.
- Check the division of two negative numbers.
- Check the division of one positive number and one integer number.
- Check the division of a number by zero.
- Check the division of a number by negative number.
- Check the division of zero by any number.
- Check if the functionality using BODMAS/BIDMAS works as expected.

Advanced Tests on Scientific Calculator

If your calculator has advanced features as shown in the screenshot.



Advanced Calculator Test Cases

You can add few more tests in the scientific calculator.

- Check if the sin, cos, tan and cot is operational using the keys.
- Check if the x^{-1} , $x!$, $|x|$, x^y and $f(x)$ is operational and works as expected.
- Check if the log key is operational and works as expected.
- Check if the natural logarithm key is operational and works as expected.
- Check if the factorial key is working as expected.
- Check if the real and imaginary component keys are working as expected.
- Check if the complex conjugate keys are working as expected.

Conversion Function Tests

Some of the advanced scientific calculator has the converter option. It does the conversion of angle, length, weight, area, volume, duration, currency, temperature. Make sure you write the testcases for the same.

Financial Calculator Tests

The additional keys for the financial calculator will be as shown in the image. Some calculator has the mode for enabling these keys.

EXPERIMENT: 11

NAME OF THE EXPERIMENT: Study of Any Bug Tracking Tool
(Bugzilla)

STUDY OF BUGZILLA,BUGBIT AND BUG TRACKING TOOL:

Bugzilla is a Bug Tracking System that can efficiently keep track of outstanding bugs in a product. Multiple users can access this database and query, add and manage these bugs. Bugzilla essentially comes to the rescue of a group of people working together on a product as it enables them to view current bugs and make contributions to resolve issues. Its basic repository nature works out better than the mailing list concept and an organized database is always easier to work with.

Advantage of Using Bugzilla:

1. Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Redhat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS, Bugzilla provides a powerful, easy to use solution to configuration management and replication problems.

2. Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Ultimately, Bugzilla puts the power in user's hands to improve value to business while providing a usable framework for natural attention to detail and knowledge store to flourish.

The bugzilla utility basically allows to do the following:

1. Add a bug into the database

2. Review existing bug reports
3. Manage the content

Bugzilla is organized in the form of bug reports that give all the information needed about a particular bug. A bug report would consist of the following fields.

1. Product->Component
2. Assigned to
3. Status (New, Assigned, Fixed etc)
4. Summary
5. Bug priority
6. Bug severity (blocker, trivial etc)
7. Bug reporter

Using Bugzilla:

Bugzilla usage involves the following activities Setting Parameters and Default Preferences

1. Creating a New User
2. Impersonating a User
3. Adding Products
4. Adding Product Components
5. Modifying Default Field Values
6. Creating a New Bug
7. Viewing Bug Reports

Setting Parameters and Default Preferences:

When we start using Bugzilla, we'll need to set a small number of parameters and preferences. At a minimum, we should change the following items, to suit our particular need:

1. Set the maintainer
2. Set the mail_delivery_method
3. Set bug change policies
4. Set the display order of bug

reports To set parameters and default
preferences:

1. Click Parameters at the bottom of the page.
2. Under Required Settings, add an email address in the maintainer field.
3. Click Save Changes.

DEPARTMENT OF CSE

4. In the left side Index list, click Email.
5. Select from the list of mail transports to match the transport we're using. If evaluating a click2try application, select test. If using SMTP, set any of the other SMTP options for your environment. Click Save Changes.
6. In the left side Index list, click Bug Change Policies.
7. Select On for comment on create, which will force anyone who enters a new bug to enter a comment, to describe the bug. Click Save Changes.

8. Click Default Preferences at the bottom of the page.
9. Select the display order from the drop-down list next to the When viewing a bug, show comments in this order field. Click Submit Changes.

CREATING A NEW USER

Before entering bugs, make sure we add some new users. We can enter users very easily, with a minimum of information. Bugzilla uses the email address as the user ID, because users are frequently notified when a bug is entered, either because they entered the bug, because the bug is assigned to them, or because they've chosen to track bugs in a certain project. To create a new user:

1. Click **users**.
2. Click **adds** a new user.
3. Enter the **login name**, in the form of an email address.
4. Enter the **real name**, a password, and then click **add**.
5. Select the **group access options**. We'll probably want to enable the following options in the row titled user is a member of these groups:
 6. Can confirm
 7. Edit bugs
 8. Edit components
 9. Click **update** when done with setting options.

Impersonating a User:

Impersonating a user is possible, though rare, that we may need to file or manage a bug in an area that is the responsibility of another user when that user is not available. Perhaps the user is on vacation, or is temporarily assigned to another project. We can impersonate the user to create or manage bugs that belong to that user.

Adding Products

We'll add a product in Bugzilla for every product we are developing. To start with, when we

first login to Bugzilla, we'll find a test product called **TestProduct**. We should delete this and create a new product.

To add a product:

1. At the bottom of the page, click **Products**.
2. In the **TestProduct** listing, click **Delete**.
3. Click **Yes, Delete**.
4. Now click **Add a product**.
5. Enter a product name, such as Widget Design Kit.
6. Enter a description.
7. Click **Add**. A message appears that you'll need to add at least one component

Adding Product Components

Products are comprised of components. Software products, in particular, are typically made up of many functional components, which in turn are made up of program elements, like classes and functions. It's not unusual in a software development team environment for different individuals to be responsible for the bugs that are reported against a given component. Even if there are other programmers working on that component, it's not uncommon for one person, either a project lead or manager, to be the gatekeeper for bugs. Often, they will review the bugs as they are reported, in order to redirect them to the appropriate developer or even another team, to review the priority and severity supplied by the reporter, and sometimes to reject bugs as duplicates or enhancement requests, for example.

To add a component:

1. Click the link **add at least one component** in the message that appears after creating a new product.
2. Enter the **Component** name.
3. Enter a **Description**.
4. Enter a **default assignee**. Use one of the users we've created. Remember to enter

the as signee inthe form of an email address.

5. Click **Add**.
6. To add more components, click the name of product in the message that reads edit othercomponents of product <**product name**>.

Modifying Default Field Values

Once we begin to enter new bugs, we'll see a number of drop down lists containing default values. Some of these may work just fine for our product. Others may not. We can modify the valuesof these fields,adding new values and deleting old ones. Let's take a look at the OS category.

To modify default field values:

1. At the bottom of the page, in the **Edit** section, click **Field Values**.
2. Click the link, in this case **OS**, for the field we want to edit. The OS field contains a list of operatingsystem names. We are going to add browsers to this list. In reality, we might create a custom field instead, but for the sake of this example, just add them to the OS list.
3. Click **Add a value**. In the **Value** field, enter IE7.Click **Add**.
4. Click **Add a value** again.
5. In the **Value** field, enter Firefox 3.
6. Click **Add**.
7. Where it reads **add other values for the op_sys field**, click **op_sys**.
8. This redisplays the table. We should now see the two new entries at the top ofthe table. Thesevalues will also appear in the OS drop down list when we create a new bug.

Creating a New Bug:

Creating bugs is a big part of what Bugzilla does best. To create a new bug:

1. In the top menu, click **New**.
2. If we've defined more than one component, choose the component from the component list.
 3. Select a **Severity** and a **Priority**. **Severity** is self explanatory, but **Priority** is generally assumed to be the lower the number, the higher the priority. So, a **P1** is the highest priority

4. Click the **OS** dropdown list to see the options, including the new browser names we entered.
5. Select one of the options. Enter a summary and a description. We can add any other information of choice, but it is not required by the system, although we may determine that our bug reporting policy requires certain information.
 - i. Click **Commit**. Bugzilla adds our bug report to the database and displays the detail page for that bug.

Viewing Bug Reports

Eventually, we'll end up with thousands of bugs listed in the system. There are several ways to view the bugs. The easiest is to click the My Bugs link at the bottom of the page. Because we've only got one bug reported, we'll use the standard Search function.

To find a bug:

1. Click **Reports**.
2. Click the **Search** link on the page, not the one in the top menu. This opens a page titled Find a Specific Bug.
3. Select the **Status**.
4. Select the **Product**.
5. Enter a word that might be in the title of the bug.
6. Click **Search**. If any bugs meet the criteria that we have entered, Bugzilla displays them in a list summary.
7. Click the **ID** number link to view the full bug report.

Modifying Bug Reports

Suppose we want to change the status of the bug. We've reviewed it and have determined that it belongs to one of the users we have created earlier.

To modify a bug report:

1. Scroll down the full bug description and enter a comment in the **Additional Comments** field.
2. Select Reassign bug to and replace the default user ID with one of the other user IDs you created. It must be in the format of an email address.

VIVA QUESTIONS:

1. Define bug tracking system?
2. Compare hardware and software bug tracking system?
3. What are the uses of Bugzilla?
4. What are the different setting parameters and default preferences?
5. Explain how to design test cases using bug bit?

EXPERIMENT: 12

NAME OF THE EXPERIMENT: Study of any open source testing tool (TestLink)

Testlink is an open source test management tool. It enables creation and organization of test cases and helps manage into test plan.

Login to TestLink

Step 1 : Open the Testlink home-page and enter the login details

1. Enter the userID – admin
2. Enter the password
3. Click on the login tab

Creating a Test Project

Step 1: In the main window click on Test Project Management, it will open another window

Step 2: Click on tab "create" to create a new project.

Test Project Management (251 Test Projects)

Search will be done on NAME in LIKE %value| Search/Filter | Reset Filter | Create

Show 25 entries

Name	Description
mre	This project is to test MRE login functionality.
HR44Transcoding	Creating a Test Project: HR44Transcoding
MTERP	MTERP
projecttime	
JIRA-TEST	

Step 3: Enter all the required fields in the window like category for test project, name of the project, prefix, description, etc. After filling all necessary details, click on tab "Create" at the end of the window.

Test Project Management (251 Test Projects) | TEST_demo_Project

Name * Guru 99 -
Prefix (used for Test case ID) * GU
Description The project is to check Guru 99 home-page login functionality

Enhanced features Requirements, Testing Priority, Test Automation (API keys), Inventory

Issue Tracker Integration Issue Tracker

Availability Active, Public

Create Cancel

This will create your project "Guru99" successfully.

Guru 99 | Search/Filter | Reset Filter | Create

Show 25 entries

Name	Description	Prefix	Issue Tracker	Requirement Feature	Active	Public	delete
Guru 99 -	The project is to check Guru 99 home-page login functionality	GU					X

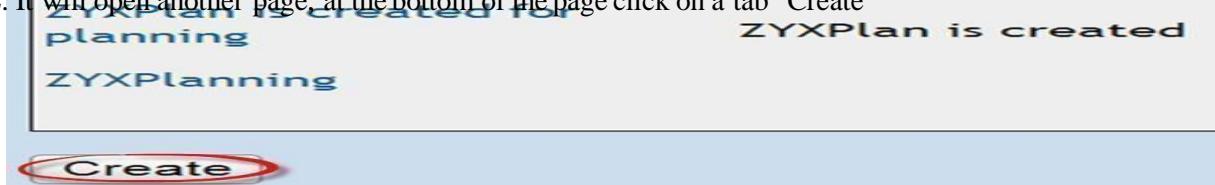
Creating a Test Plan

Test plan holds the complete information like scope of Software testing, milestone, test suites and test cases. Once you have created a Test Project, next step is to create Test plan.

Step 1: From the home-page, click on Test Plan Management from home-page

To create Test Plan, select Test Plan Management from home page

Step 2: It will open another page, at the bottom of the page click on a tab "Create"



Step 3: Fill out all the necessary information like name, description, create from existing test plan, etc. in the open window, and click on "create tab".

Fill out all the necessary details and click on "create" tab

Step 4: Guru 99 Test Plan is created successfully

Guru 99 Test plan is created successfully

Build Creation

Build is a specific release of software

Step 1: Click on Builds/Releases under Test Plan from the home page

Test Plan

- [Test Plan Management](#)
- [Builds / Releases](#)
- [Milestones](#)

Step 2: In the next window, fill all necessary details for software release and click on create to save your release

From home-page select Build/Release for creating specific release for your software

1. Enter the title name
2. Enter the description about the software release
3. Mark the check-box for status- Active
4. Mark the check-box for status- Open
5. Choose the date of release
6. Click on create button

Once you have a release the software it, will appear like this

Title	Description
Guru 99- sample build	This is a sample build for Guru 99 web application
Test Build	Just for testing
samplebuild	this is sample build

Creating Test suite

Test suite is a collection of test cases which may be testing or validating the same component. Following steps will explain how to create test suite for your project.

Test Specification

Step 1: Click on test specification option from the home page.

Test Specification 

Search Test Cases
Assign Keywords
Test Cases created per User

To create test suite for Guru 99 project, select Test Specification from home page



Step 2: On the right-hand side of the panel, click on the setting icon operation.

. It will display a series of test

Step 3: Click on the "create" tab for the test suite

Test Suite Operations

Click on "Create" tab

Step 4: Fill-up all the details for test-suite and click on save it tab.

1. Enter the test suite name
2. Enter the details about your test suite
3. Click on save button to save the details of test-suite

Test Suite Name
Guru99 Login 

Details

The test suite contains test cases related to test Guru 99 login functionality. 

Keywords

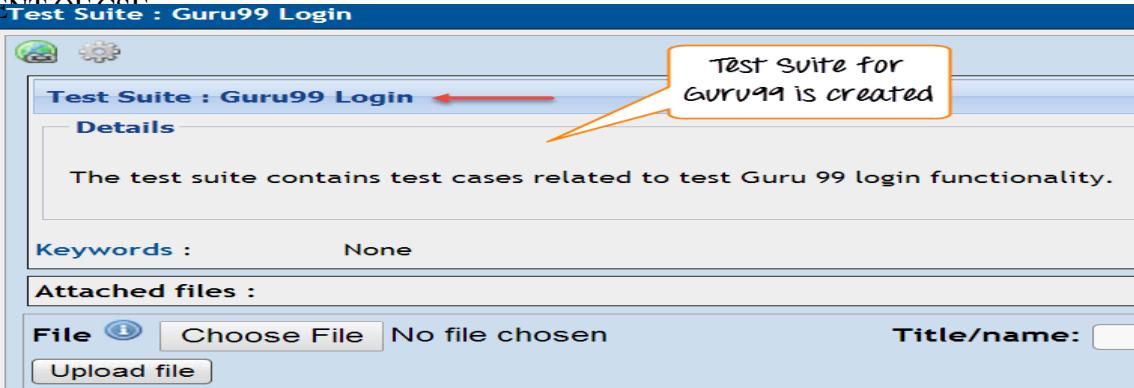
Available

2G Removal
2G-Only RAT Support
3G RAT Support
Components
DOM
DVM
hfujdhfkujdyrfjdhkfud
HSDPA 

Save Cancel 

Fill up all the required details for test-suite and save it

You can see test suite for Guru 99 is created

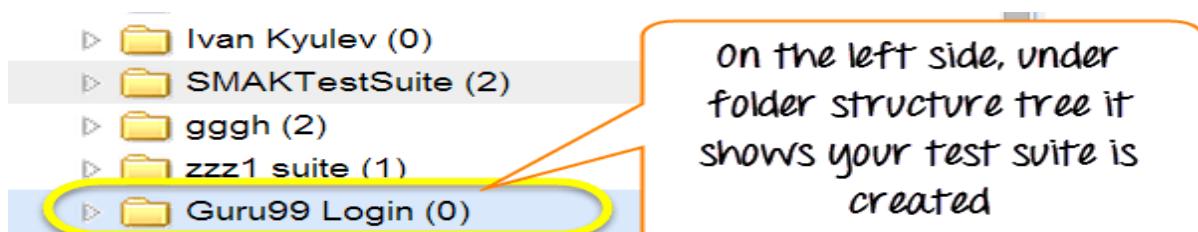


Your test suite appears on the left side of the panel under folder structure tree

Creating a Testcase

Testcase holds a sequence of test steps to test a specific scenario with expected result. Below steps will explain how to create a test-case along with test steps.

Step 1: Click on the test suite folder on the left side of the panel under folder tree structure



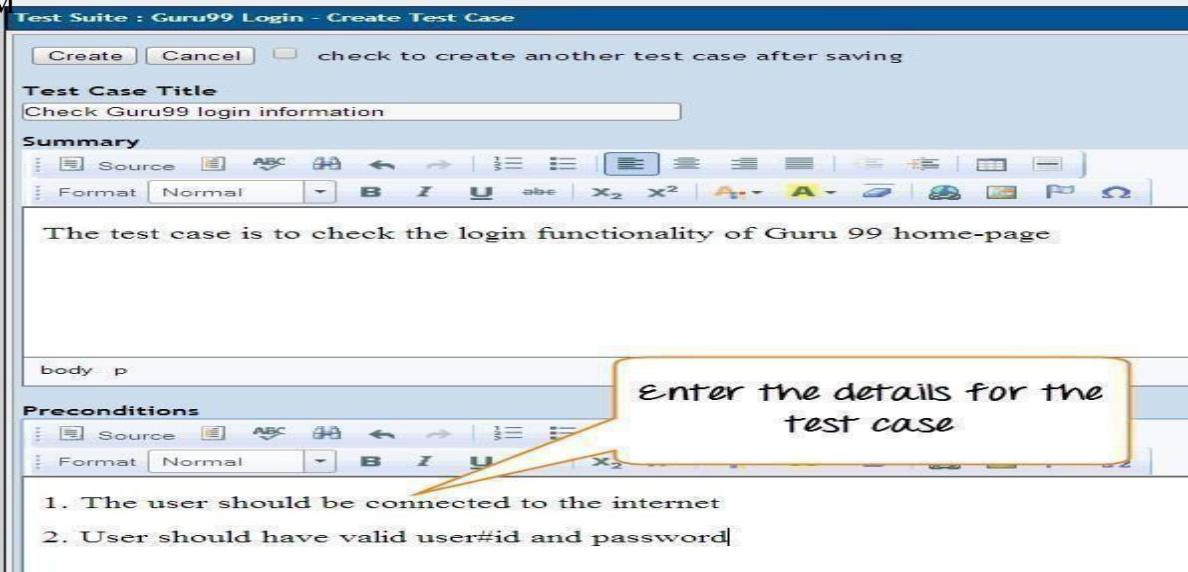
Step 2: Click on the setting icon in the right side panel. List of test case operations will be displayed on the right side panel



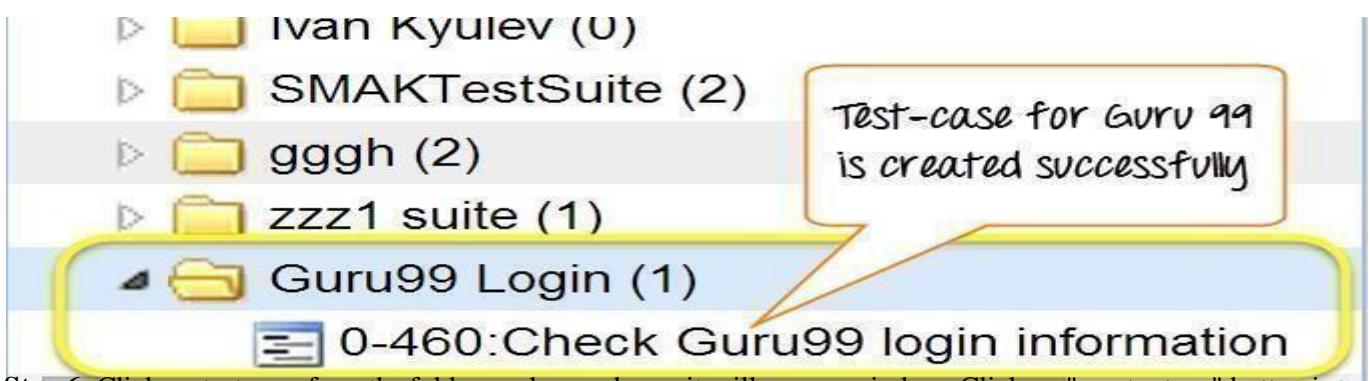
Step 3: New window will open, to create test cases click on create button in test-case operations



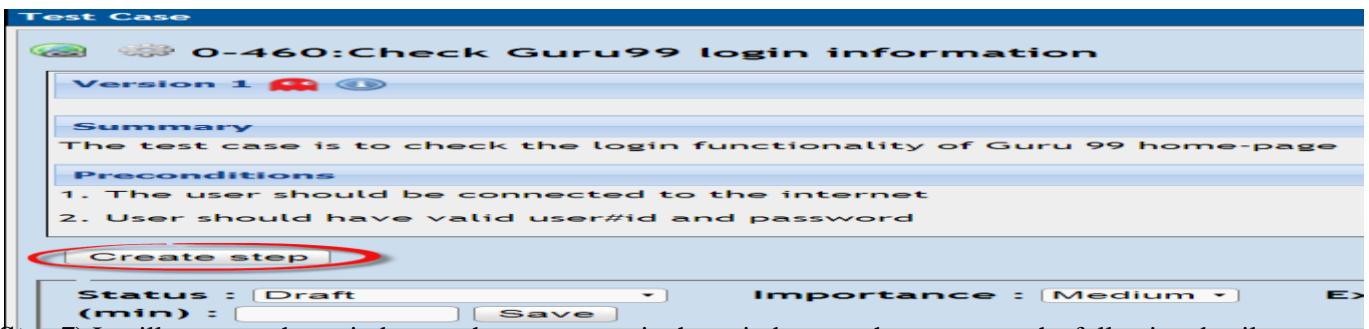
Step 4: Enter the details in the test case specification page



Step 5: After entering the details, click on "create" button to save the details. The test-case for Guru99 is created successfully

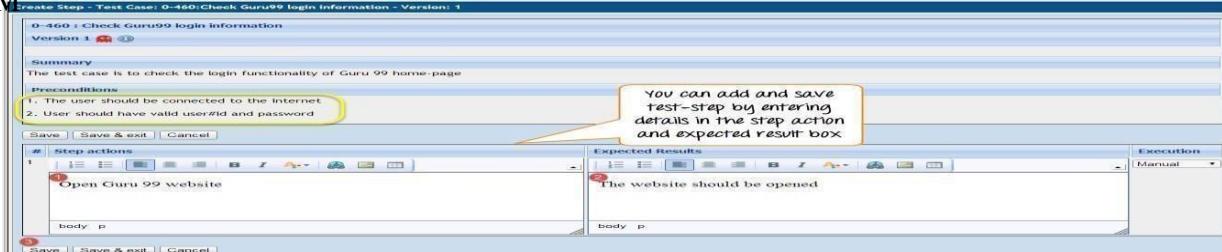


Step 6: Click on test-case from the folder as shown above, it will open a window. Click on "create steps" button in test case. It will open a test case step editor



Step 7: It will open another window on the same page, in that window you have to enter the following details

1. Enter the step-action for your test case
2. Enter the details about the step action
3. Click save it and add another step action OR click save and exit tab if there is no more test step to add



Step 8) Once you save and exit the test step, it will appear like this

This screenshot shows the same test case interface after saving. The "Step actions" and "Expected Results" table now contains four rows of test steps. The first three rows are highlighted with a yellow box. A yellow callout arrow points from the text "The test step will appear like this on the window after save and exit" to this highlighted area. The fourth row is not highlighted. The columns are labeled "Step actions", "Expected Results", and "Execution". The execution status for all steps is "Manual" with a red "X" icon.

#	Step actions	Expected Results	Execution
1	Open Guru 99 website	The website should be opened	Manual
2	Enter username in the username textbox	Text-box should accept the entered data	Manual
3	Enter password in the password text-box	Text-box should accept the entered data	Manual
4	Click on "sign in" button	Login should success and navigate to the home page	Manual

Assigning test case to test plan

For test case to get execute, it should be assign to test plan. Here we will see how we can assign a test-case to test plan.

Step 1) Click on the setting icon on the test panel. It will show the list of operations.

Step 2) Click on "Add to Test Plans"

This screenshot shows the "Test Case" interface for the same test case. At the top, there's a toolbar with buttons for "Edit", "Delete", "Move / Copy", "New sibling", "Export", "Print view", "New version", "Deactivate this version", and "Add to Test Plans". The "Add to Test Plans" button is circled in red. A yellow callout box points to this button with the text: "From the given list of operations, select 'Add to Test Plans'". Below the toolbar, there's a summary section with the same test case details as before. The "Preconditions" section is also present.

Step 3) New window will open, search your project "Guru99"

1. Mark the check box against your test plan
2. Click on add button

1 Mark the check box and then click on Add button

2 Showing 1 to 1 of 1 entries (Total 1)

Add Cancel

This will add your test case to your Test Plan.

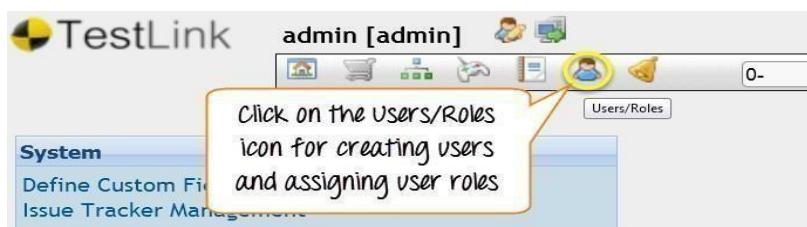
Creating Users and Assigning Roles in TestLink

Testlink provides User management and authorization features.

Below is list of default roles in Testlink and their rights -

Role	Test Cases	Test Metrics
Guest	View	View
Tester	Execute	View
Senior Tester	Edit & Execute	View
Leader & Admin	Edit & Execute	Edit & Execute

Step 1: From the Testlinks home-page, click on users/roles icon from the navigation bar



Step 2: Click Create

Step 3: Fill out all the users details and click the "Save" button

User Management - Account Settings

New User	View Users	View roles	Assign Test Project roles	Assign Test Plan roles																		
User details <table border="1"> <tr><td>Login</td><td>Jamesguru</td></tr> <tr><td>First Name</td><td>Jamesguru</td></tr> <tr><td>Last Name</td><td>Bryan</td></tr> <tr><td>Password</td><td>*****</td></tr> <tr><td>Email</td><td>jamesguru@gmail.com</td></tr> <tr><td>Role</td><td>Senior Test Engineer</td></tr> <tr><td>Locale</td><td>English (wide/UK)</td></tr> <tr><td>Authentication method</td><td>Default(DB)</td></tr> <tr><td>Active</td><td><input checked="" type="checkbox"/></td></tr> </table> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> Fill out all the details about the user and click on save button </div>					Login	Jamesguru	First Name	Jamesguru	Last Name	Bryan	Password	*****	Email	jamesguru@gmail.com	Role	Senior Test Engineer	Locale	English (wide/UK)	Authentication method	Default(DB)	Active	<input checked="" type="checkbox"/>
Login	Jamesguru																					
First Name	Jamesguru																					
Last Name	Bryan																					
Password	*****																					
Email	jamesguru@gmail.com																					
Role	Senior Test Engineer																					
Locale	English (wide/UK)																					
Authentication method	Default(DB)																					
Active	<input checked="" type="checkbox"/>																					

Here in the list we can see the users have been created

jafgc	João		joaofgc@gmail.com	leader
james	james		abc@163.com	leader
Jamesguru	Jamesguru	Bryan	jamesguru@gmail.com	Senior Test Engineer
jan	jan	sivaprakasam	jan.sivaprakasam@cancer.or...	leader

Step 4: Allotting test project role to the user,

1. Click on "Assign Test Project Roles" tab
2. Choose the project name
3. Select the users role from the drop down

User Management - Assign roles

New User	View Users	View roles	Assign Test Project roles	Assign Test Plan roles
Test Project: Guru 99 - 1 2 3				
Set roles to: Senior Test Engineer 1 2 3				
Show 25 entries				
User: Jamesguru (Jamesguru Bryan) 1 2 3				
Test Project Role (Guru 99 -) 1 2 3				
<inherited> Senior Test Engineer				

We assigned our test project "Guru99" to senior test engineer- Jamesguru

Writing Requirements:

Step 1: From the navigation bar select the "Requirements Link", it opens the Requirement page.



Step 2: From the requirement page, on the right side of the panel click on "create" button



Step 3: A new window will open, enter all the details like

1. Document ID
2. Title name
3. Requirement description
4. And Click "Save" button

For the type, you can choose the option from the drop-down- here we chose "**User Requirement Specification**"

Save Cancel 4

Document ID
01 1

Title
Guru99 home-page login 2

Scope

Format Normal 3

When user enters valid username and password, and click on submit button, it should navigate to Guru99 home-page.

Type User Requirement Specification

Step 4: It should create Requirement specification and displayed on the left side panel under project "Guru99".

on left hand side of the panel,
you can see requirement
specification for our project
"Guru99" is created

Step 5: Select the setting button from requirements specification home-page. It will open another window.

Click on the settings icon, from
your requirement specification page

Step 5: Click "Create" tab under Requirement Operations.

Under Requirement
Operations option, click
on "create" tab

Step 6: Fill out all the specified details and click the "Save" button

1. Enter the document ID
2. Enter the title name
3. Enter the description
4. Enter the status-whether it's in draft, rework, review, not testable, etc. Here we chose valid
5. Enter the type – user interface, non-functional, informational, feature, etc. Here we chose usecase
6. Enter the number of test cases needed
7. Enter "Save" button at the end

Document ID
02

Title
Guru99 home-page login

Scope

Format Normal

Status Valid

Type Use Case

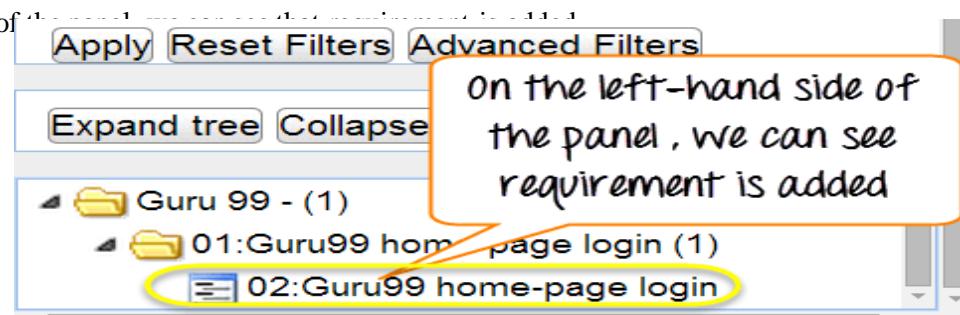
Number of test cases needed 1

Enter all the details and then click on "Save" button

3
When user login with valid user name and password it should navigate to the Guru99 home-page.

Note: To add more requirements you can mark the check-box and click save button

On the left side of



Assigning requirement to test-cases

In Testlink, Requirement can be connected to test cases. It is very crucial feature in order to track test coverage based on requirements. In test reports, you can verify which requirements are not covered and act on them to append in test suites for maximum test coverage

Step 1: From test

#	Step actions	Expected Results
1	Open Guru 99 website	The website should be opened
2	Enter username in the username textbox	Text-box should accept the entered data
3	Enter password in the password text-box	Text-box should accept the entered data
4	Click on "sign in" button	Login should success and navigate to the home page

Create step Resequence Steps

Status : Draft Import

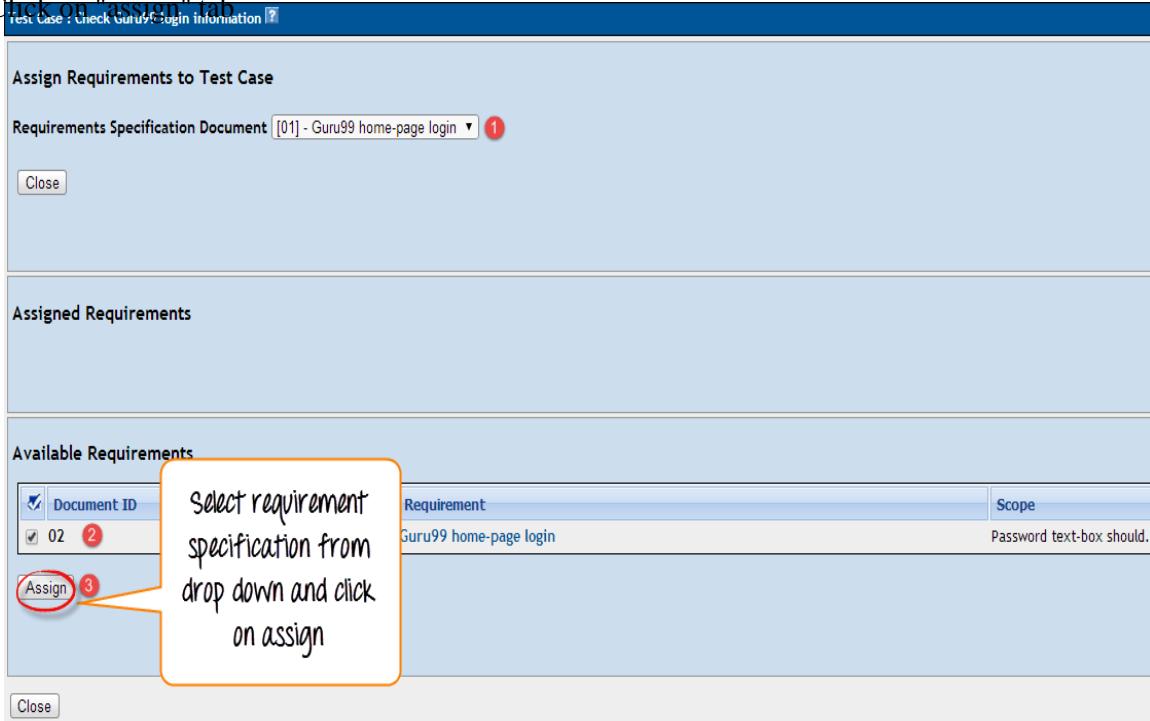
Keywords: None

Requirements : None

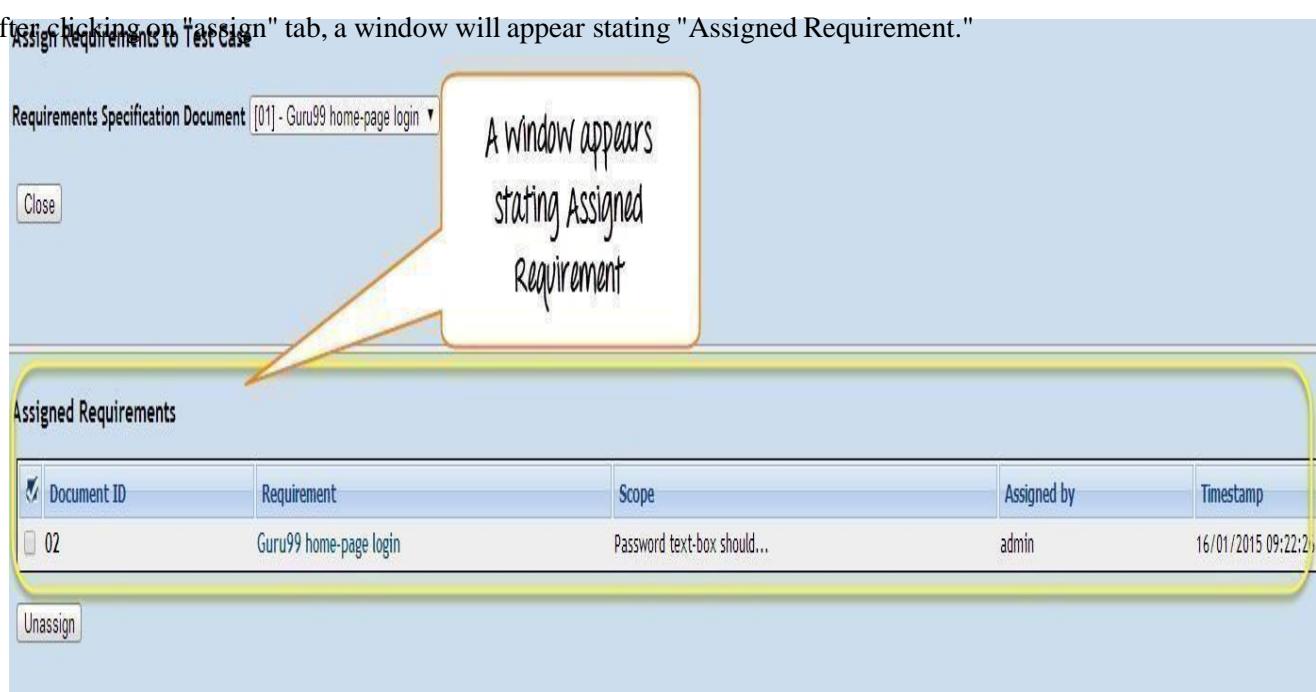
From the test specification section, click on requirements icon

Step 2: To assign requirements specification to test case you have to follow the following steps

1. Scroll the drop down box to select the requirements specification
2. Mark the requirement check box
3. Click on "Assign" tab



After clicking on "Assign" tab, a window will appear stating "Assigned Requirement."



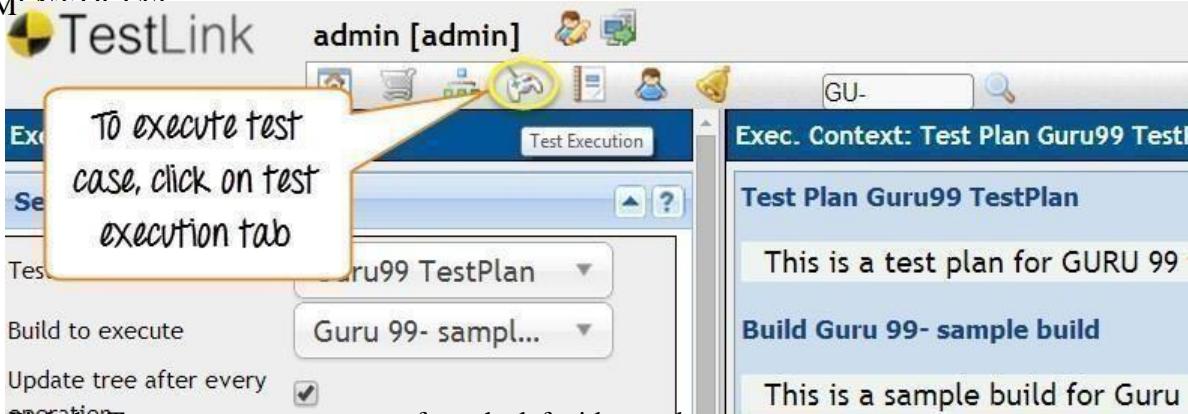
Executing a test case

In TestLink, we can run a test case and change execution status of a test case. Status of a test-case **can be set** to "blocked" "Passed", or "failed". Initially, **it will be in "not run"** status but once you have updated it, it cannot be

DEPARTMENT OF CSE

altered to "not run" status again.

Step 1: From the navigation bar click on the "Test Execution" link. It will direct you to the Test Execution Panel.



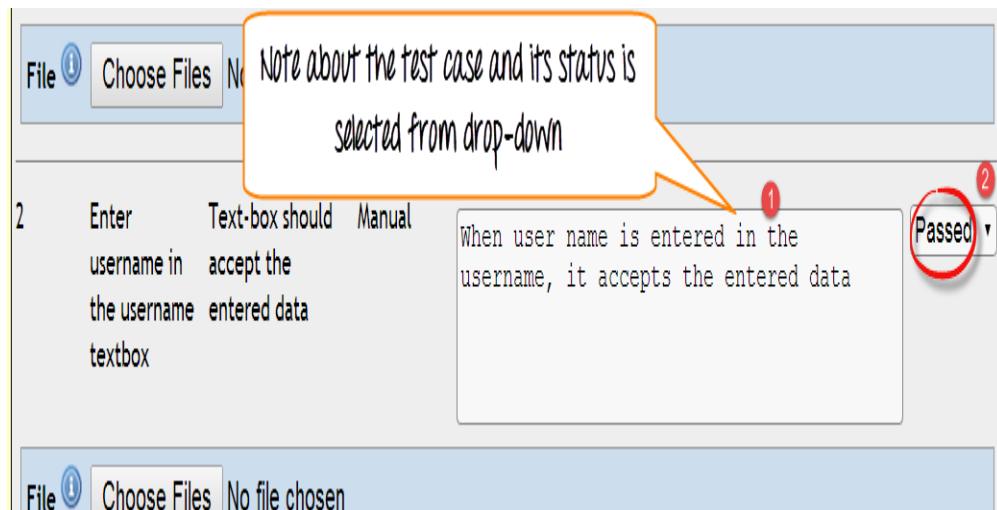
Step 2: Pick the Test case you want to run from the left side panel

Step 3: Once you have selected the test cases, it will open a window.

#	Step actions	Expected Results	Execution	Execution notes	Result
1	Open Guru 99 website	The website should be opened	Manual		
2	Enter username in the username textbox	Text-box should accept the entered data	Manual		
3	Enter password in the password text-box	Text-box should accept the entered data	Manual		
4	Click on "sign in" button	Login should success and navigate to the home page	Manual		

Step 4: Follow the following steps

1. Enter the notes related to test case executed
2. Select its status



Step 5: On the same page, you have to fill similar detail about the execution of test-case. Fill the details, select the status and then click on "save execution".

Test Results on Build Guru 99- sample build

Print Show complete execution history Import Results (XML)

Test Plan Guru99 TestPlan

Build Guru 99- sample build

Test Suite : Guru99 Login/

Latest execution (any build)

Date : 16/01/2015 10:35:06 - Tested by : admin - Build : Guru 99- sample build - Status : Passed

Latest execution (current build) - Build : Guru 99- sample build

Date	Build	Tested by	Status	Exec (min)	Version	Run mode
16/01/2015 10:35:06	Guru 99- sample build	admin	Passed	1		
Notes						
Test executed successfully in Firefox 12, in windows 8 platform						

Generating Test Reports

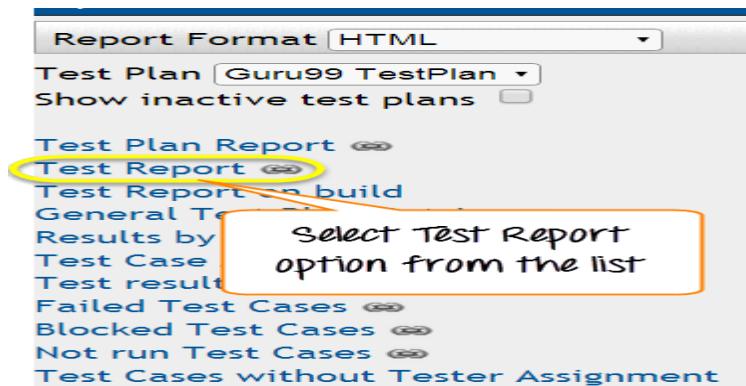
Test link supports various test report formats like

- HTML
- MS Word
- MS excel
- OpenOffice Writer
- OpenOffice calc

Step 1: From the navigation bar, click on Test Reports option

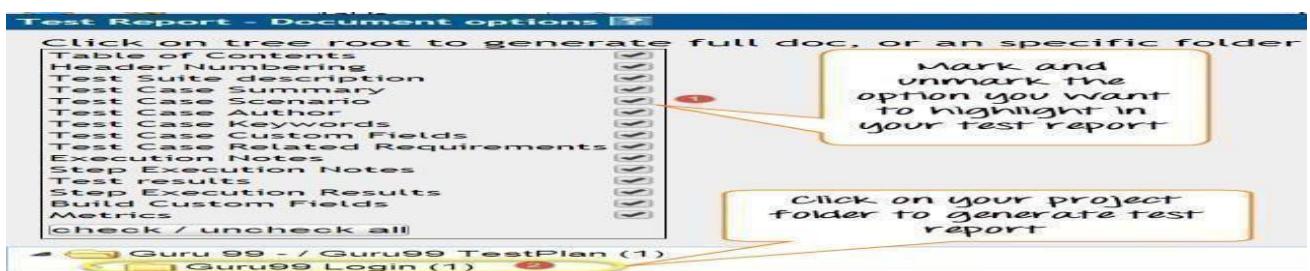


Step 2: From the left side panel, select "Test Report" link



Step 3: To generate a report follow the following steps

1. Mark and unmark the option you want to highlight in your test report
2. click on your project folder



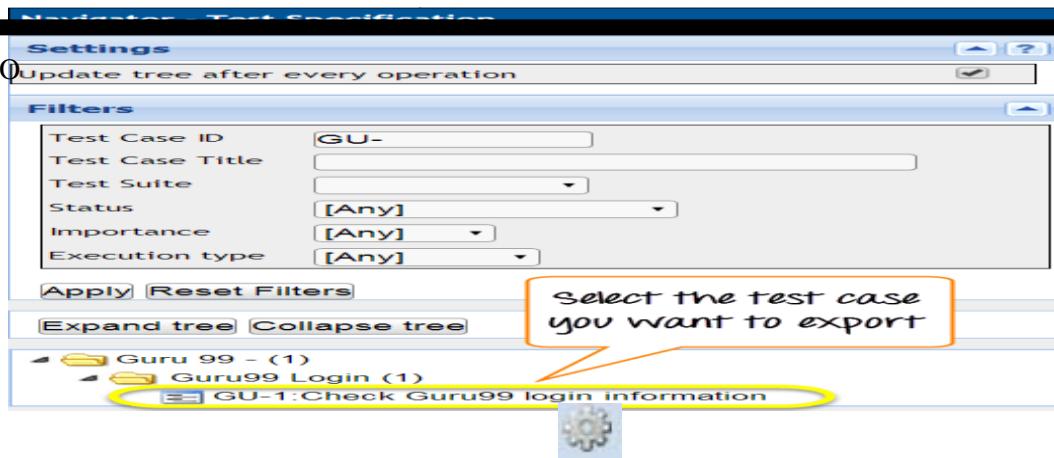
The test report will look like this

1.1. Test Suite : Guru99 Login				
The test suite contains test cases related to test Guru 99 login functionality				
Test Case GU-1: Check Guru99 login Information				
Author: admin Preconditions: 1. The user should be connected to the internet 2. User should have valid user#id and password				
#	Step actions:	Expected Results:	Execution notes:	Execution Status:
1	Open Guru 99 website Enter username in the user name text-box	The website should be opened Text-box should accept the entered data	When user name is entered in the username, it accepts the entered data	Passed
2	Enter password in the password text-box	Text-box should accept the entered data		
3	Click on "sign in" button	Login should success and navigate to the home page		
4	Manual			
Execution type:				

Export Test case/ Test Suite

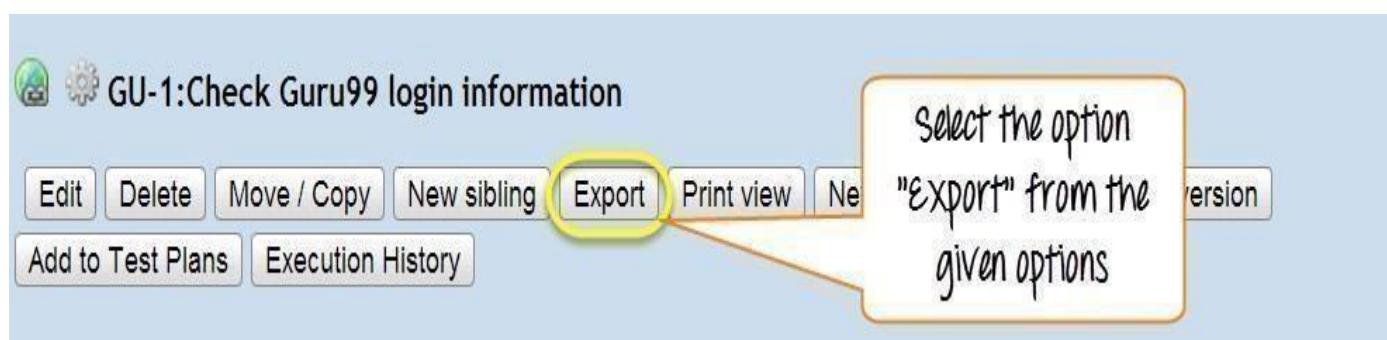
Testlink provides the features to export test projects/test suites in your Testlink and then you can import them into another Testlink project on different server or system. In order to do that you have to follow the following step

Step 1: Choose the test case you want to export in the Test specification page.

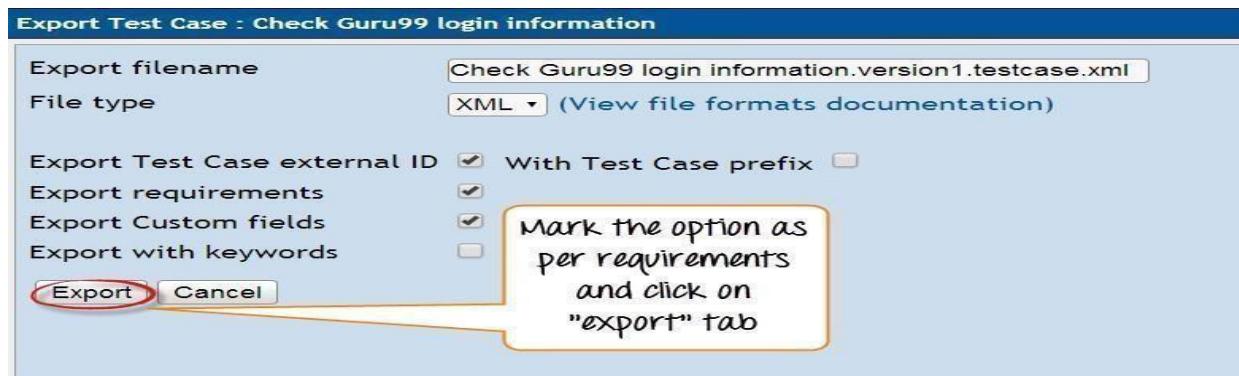


Step 2: Now on the right-hand side of the panel click on the setting icon, it will display all the operations that can be performed on the test case.

Step 3: Click the "export" button



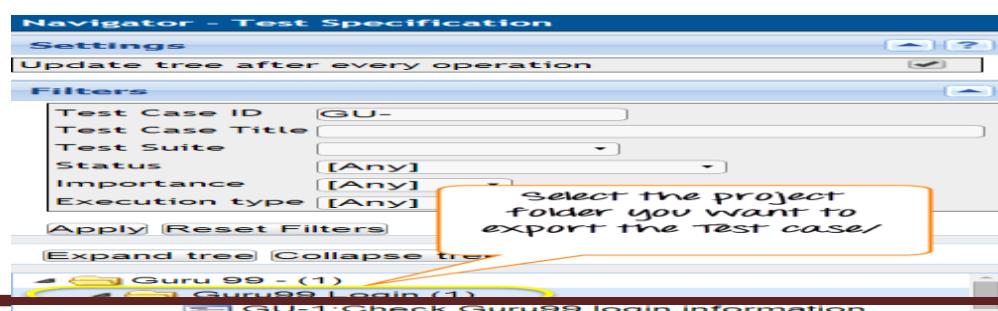
Step 4: It will open another window, mark the option as per requirement and click on the export tab



Following XML is generated

Importing Test case/ Test suite

Step 1: Select the Test suite folder inside which you want to import the test case



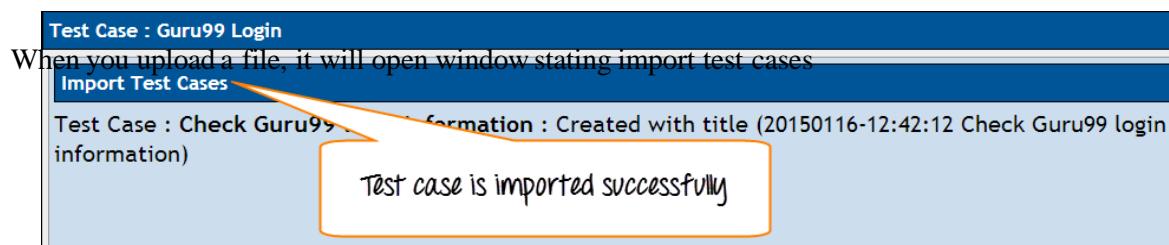
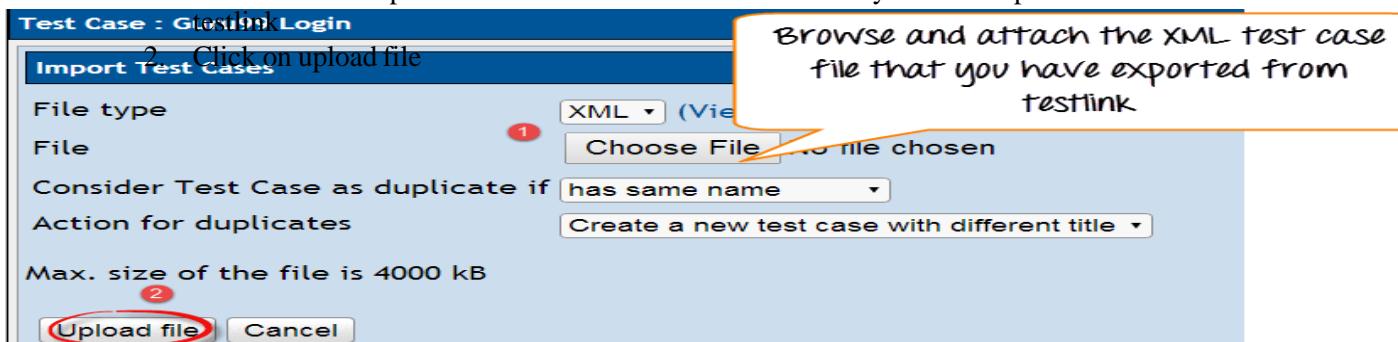
Step 2: Click on the setting icon on the right hand-side of the panel, it will display all the operations that can be executed on the test suite/test case

Step 3: Click on the import button in the test case operations list as



Step 4: Browse and attach the XML test case file that you have exported from test link and click on upload button.

1. Use the browse option to attach the XML test case file that you have exported from



Step 5: Test case will be uploaded and displayed on the right-hand side of the panel

