ASSIGNMENT – TASK 9

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

Q) Write a C program to demonstrate the working of the following constructs:

i. do…while

ii. while…do

iii. if …else

iv. switch

v. for Loops.

❖ **To check whether the given number is even or odd.**

Source Code and Test Cases:

i. do…while

```
#include <stdio.h>
int main()
{
   int num;
    printf("Enter a number: ");
   scanf("%d", &num);
   if(num<0)
   {
          printf("Fail ");
          return 0;
  }
  do
  {
         if (num%2 == 0)
        {
            printf("%d is even.\n",
            num);
        }
        else
        {
          printf("%d is odd.\n", num);
```

```
        }
        num=0;
}
while (num!= 0);
return 0;
}
```

Test cases:

1. Positive values within range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
10	10 IS EVEN	10 IS EVEN	PASS
11	11 IS ODD	11 IS ODD	PASS
12	12 IS EVEN	12 IS EVEN	PASS
13	13 IS ODD	13 IS ODD	PASS
14	14 IS EVEN	14 IS EVEN	PASS

2. Negative values in a range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
-10	-10 IS EVEN	-10 IS EVEN	FAIL
-11	-11 IS ODD	-11 IS ODD	FAIL
-12	-12 IS EVEN	-12 IS EVEN	FAIL
-13	-13 IS ODD	-13 IS ODD	FAIL
-14	-14 IS EVEN	-14 IS EVEN	FAIL

3. Out of range values

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
1234567891222222222222	12345678912222222213	234567891222222215	FAIL

ii. while…do

```
#include <stdio.h>
int main()
{
int num;
printf("Enter a number:
");
scanf("%d", &num);
if(num<0)
{
          printf("Fail");
          return 0;
}
while (num!= 0)
{
      if (num % 2 == 0)
      {
         printf("%d is
         even.\n", num);
       }
      else
      {
         printf("%d is odd.\n", num);
      }
      num=0;
}
return 0;

}
```

Test cases:

1. Positive values within range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
10	10 IS EVEN	10 IS EVEN	PASS
11	11 IS ODD	11 IS ODD	PASS
12	12 IS EVEN	12 IS EVEN	PASS
13	13 IS ODD	13 IS ODD	PASS
14	14 IS EVEN	14 IS EVEN	PASS

2. Negative values in a range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
-10	-10 IS EVEN	-10 IS EVEN	FAIL
-11	-11 IS ODD	-11 IS ODD	FAIL
-12	-12 IS EVEN	-12 IS EVEN	FAIL
-13	-13 IS ODD	-13 IS ODD	FAIL
-14	-14 IS EVEN	-14 IS EVEN	FAIL

3. Out of range values

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
1234567891222222222222	12345678912222222213	234567891222222215	FAIL

iii. if …else

```
#include <stdio.h>
int main()
{
    int num;
    printf("Enter a
    number: ");
    scanf("%d", &num);
    if(num<0)
   {
      print(“Fail”);
      return 0;
   }
  for(; ; )
   {
     if (num % 2 ==
     0)
     {
        printf("%d is even.\n", num);
     }
     else
     {
       printf("%d is
      odd.\n", num)
```

```
    }
    num=0;
   if (num == 0)
   {
      break;
   }
}
printf("Program exited.\n");
return 0;
}
```

Test cases:

1. Positive values within range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
10	10 IS EVEN	10 IS EVEN	PASS
11	11 IS ODD	11 IS ODD	PASS
12	12 IS EVEN	12 IS EVEN	PASS
13	13 IS ODD	13 IS ODD	PASS
14	14 IS EVEN	14 IS EVEN	PASS

2. Negative values in a range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
-10	-10 IS EVEN	-10 IS EVEN	FAIL
-11	-11 IS ODD	-11 IS ODD	FAIL
-12	-12 IS EVEN	-12 IS EVEN	FAIL
-13	-13 IS ODD	-13 IS ODD	FAIL
-14	-14 IS EVEN	-14 IS EVEN	FAIL

3. Out of range values

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
1234567891222222222222	12345678912222222213	234567891222222215	FAIL

iv. switch

```
#include <stdio.h>
int main()
{
    int num;
    printf("Enter a
    number:");
    scanf("%d", &num);
    if(num<0)
    {
          printf("Fail");
          return 0;
   }
  switch(num % 2)
  {
      case 0: printf("%d is
                even.\n",num);
                break;
     default: printf("%d is
                odd.\n", num);
  }
 return 0;
}
```

Test cases:

1. Positive values within range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
10	10 IS EVEN	10 IS EVEN	PASS
11	11 IS ODD	11 IS ODD	PASS
12	12 IS EVEN	12 IS EVEN	PASS
13	13 IS ODD	13 IS ODD	PASS
14	14 IS EVEN	14 IS EVEN	PASS

2. Negative values in a range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
-10	-10 IS EVEN	-10 IS EVEN	FAIL
-11	-11 IS ODD	-11 IS ODD	FAIL
-12	-12 IS EVEN	-12 IS EVEN	FAIL
-13	-13 IS ODD	-13 IS ODD	FAIL
-14	-14 IS EVEN	-14 IS EVEN	FAIL

3. Out of range values

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
12345678912222222222222	1234567891222222222213	234567891222222215	FAIL

v. for Loops.

```
#include <stdio.h>
int main()
{
    int num;
    printf("Enter a
    number: ");
    scanf("%d", &num);
    if(num<0)
   {
        printf("Fail");
        return 0;
   }
   for (; ;)
  {
     if (num % 2 == 0)
     {
          printf("%d is even.\n", num);
     }
    else
    {
        printf("%d is odd.\n", num);
```

```
    }
    num=0;
  if (num == 0)
 {
    break;
 }
}
printf("Program exited.\n");
return 0;
}
```

Test cases:

1. Positive values within range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
10	10 IS EVEN	10 IS EVEN	PASS
11	11 IS ODD	11 IS ODD	PASS
12	12 IS EVEN	12 IS EVEN	PASS
13	13 IS ODD	13 IS ODD	PASS
14	14 IS EVEN	14 IS EVEN	PASS

2. Negative values in a range

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
-10	-10 IS EVEN	-10 IS EVEN	FAIL
-11	-11 IS ODD	-11 IS ODD	FAIL
-12	-12 IS EVEN	-12 IS EVEN	FAIL
-13	-13 IS ODD	-13 IS ODD	FAIL
-14	-14 IS EVEN	-14 IS EVEN	FAIL

3. Out of range values

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
1234567891222222222222	12345678912222222213	234567891222222215	FAIL

ASSIGNMENT – TASK

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

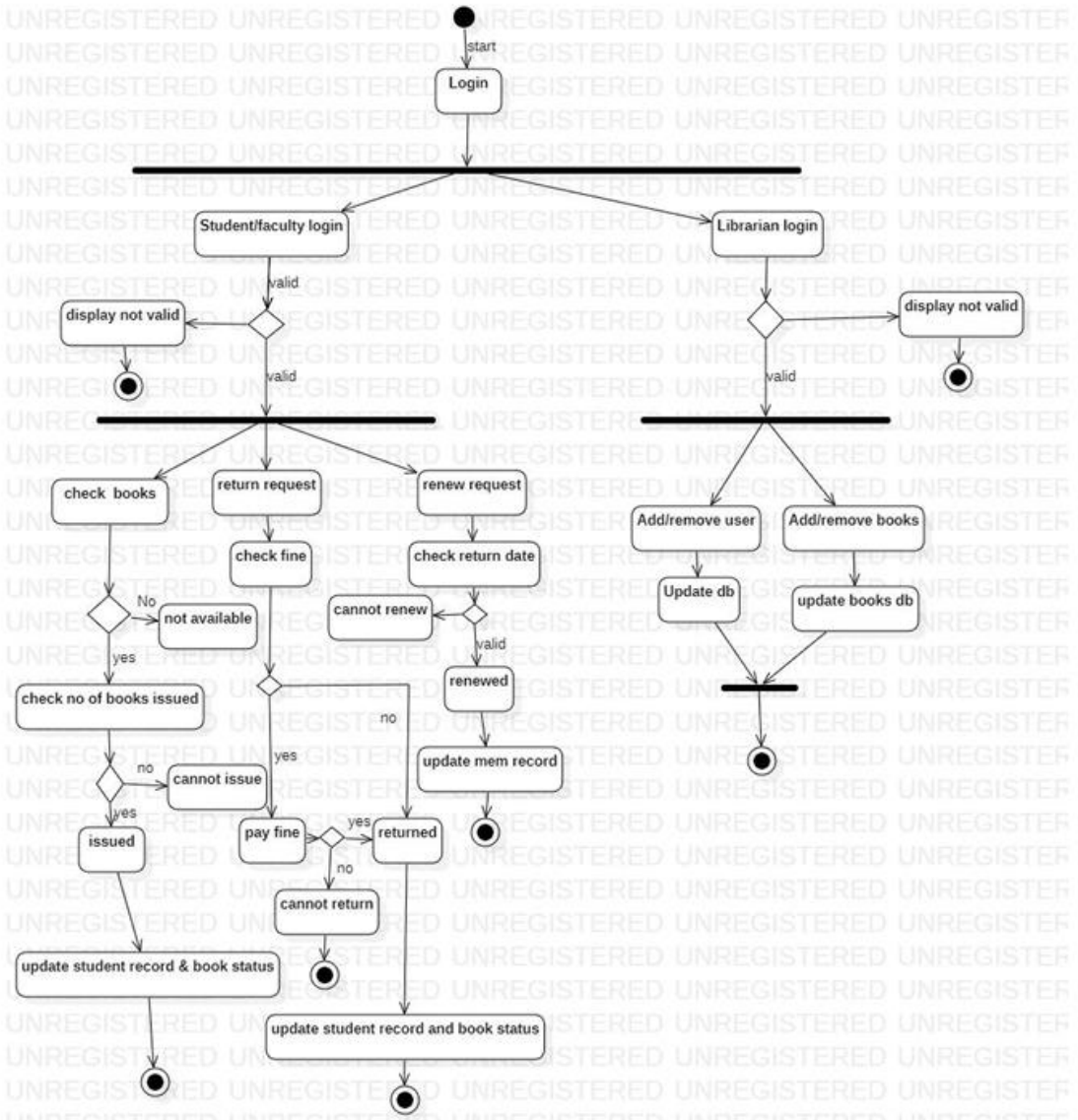

Fig:1: Activity diagram for Unified Library System

prepare test cases based on above take some negative actions based on your knowledge

S No	Input	Expected Output	Actual Output	Result	Problem
1	Student Login	Login successful	Login successful	Pass	-
2	Librarian Login	Login successful	Login Failed	Fail	Invalid credentials
3	Return Request	Request Accepted	Request Accepted	Pass	-
4	Check Fine	No due	Due	Fail	Fine not updates properly
5	Renew request	Request Accepted	Request Declined	Fail	Issue with the server
6	Updated record	Record updated successfully	Record updated successfully	Pass	-
7	Returned book status	Returned	Not returned	Fail	Data not updated
8	Update member record	Record Updated	Record Updated	Pass	-
9	Book Status	Updated	Updated	Pass	-
10	Remove user	User removed	User removed	Pass	-
11	Add user	User added	No user added	Fail	New user not updated
12	Update database	Database updated	Database updated	Pass	-
13	Update books database	Books database updated	Books database not updates	Fail	Issue with database
14	Check books	Available	Available	Pass	-
15	Add book	Book added	Book added	Pass	-
16	Remove book	Book removed	Book not removed	Fail	Book status not updated

17	No. of books issued	100	100	Pass	-
18	Faculty login	Login successful	Login unsuccessful	Fail	Invalid credentials
19	Pay fine	Paid	Paid	Pass	-
20	Update student record	Updated	Updated	Pass	-

ASSIGNMENT – TASK 11

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

EXPERIMENT # 3
SYSTEM SPECIFICATIONS

3.1 OBJECTIVE:

1. Study the system specifications of ATM system and report various bugs in it.
2. Study the system specifications of banking application and report various bugs in it.

3.2 BUGS IN ATM SYSTEM:

- **Machine is accepting ATM card.**
- **Machine is rejecting expired card.**
- **Successful entry of PIN number.**
- **Unsuccessful operation due to enter wrong PIN number 3 times.**
- **Successful selection of language.**
- **Successful selection of account type.**
- **Unsuccessful operation due to invalid account type.**
- **Successful selection of amount to be withdrawn.**
- **Successful withdrawal.**
- **Expected message due to amount is greater than day limit.**
- **Unsuccessful withdraw operation due to lack of money in ATM.**
- **Expected message due to amount to withdraw is greater than possible balance.**
- **Unsuccessful withdraw operation due to click cancel after insert card.**

Operation	Expected Output	Actual Output	Result	Problem
Accepting Card	Accepted	Accepted	Passed	-
Expired Pin Number	Not - Accepted	Not - Accepted	Failed	Pin number expired / Wrong pin number
Successful entry of pin number	Accepted	Accepted	Passed	-
Unsuccessful operation due to enter wrong PIN number 3 times.	Not - Accepted	Not - Accepted	Failed	System might not be locking the card after three unsuccessful attempts, allowing for further attempts.
Successful selection of language.	Accepted	Accepted	Passed	-
Successful selection of account type.	Accepted	Accepted	Passed	-
Unsuccessful operation due to invalid account type.	Not - Accepted	Not - Accepted	Failed	System does not provide a clear error message when an invalid account type is selected.
Successful selection of amount to be withdrawn.	Accepted	Accepted	Passed	-
Successful withdrawal.	Accepted	Accepted	Passed	-
Expected message due to amount is greater than day limit	Accepted	Accepted	Passed	-
Unsuccessful withdraw operation due to lack of money in ATM.	Not - Accepted	Not - Accepted	Failed	System does not provide a clear error message indicating that the ATM is out of cash.
Expected message due to amount to withdraw is greater than possible balance.	Accepted	Accepted	Passed	-
Unsuccessful withdraw operation due to click cancel after insert card.	Not - Accepted	Not - Accepted	Failed	System might not handle the cancel action properly, potentially leaving the user's account vulnerable.

2.2 VIVA QUESTIONS:

1. What are design specifications?

Design specifications are detailed descriptions of how a system should be built, including its architecture, interfaces, components, and behaviors.

2. What are different types of bugs?

There are various types of bugs, including syntax errors, logic errors, runtime errors, and semantic errors.

3. Compare functional and structural testing?

Functional testing evaluates the functionality of a system against its requirements, while structural testing examines the internal structure or code of the system to ensure it functions correctly.

4. Explain dichotomies of testing?

Dichotomies of testing refer to contrasting principles in testing, such as verification vs. validation, black-box vs. white-box testing, and static vs. dynamic testing.

5. What are consequences of bugs?

Consequences of bugs include system failures, security vulnerabilities, financial losses, damage to reputation, and potential harm to users. Therefore, thorough testing and bug fixing are crucial in software development.

ASSIGNMENT – TASK 12

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

EXPERIMENT # 5

TEST PLAN

5.1 OBJECTIVE:

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

TEST PLAN DOCUMENT FOR LIBRARY MANAGEMENT SYSTEM:

The Library Management System is an online application for assisting a librarian imagining 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. This test report is the result for testing in the LMS. It mainly focuses on two problems:

1. What we will test
2. How we will test.

1. GUI TEST

Pass criteria: librarians could use this GUI to interface with the backend

library database without any difficulties

2. DATABASE TEST

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

3. BASIC FUNCTION TEST ADD A STUDENT

1. Each customer/student should have following attributes: Student ID/SSN (unique), Name, Address and Phone number.
2. The retrieved customer information by viewing customer detail should contain the four attributes.

4. UPDATE/DELETE STUDENT

1. The record would be selected using the student ID.
2. Updates can be made on full. Items only: Name, Address, Phone number. The record can be deleted if there are no books issued by user. The updated values would be reflected if the same customer's ID/SSN is called for.

5. CHECK-IN BOOK

1. Librarians can check in a book using its call number.
2. The check-in can be initiated from a previous search operation where user has selected a set of books.
3. The return date would automatically reflect the current system date.
4. Any late fees would be computed as difference between due date and return date at rate of 10 cents a day.

A) Login Form:

Sno	Test Case	Actual Output	Expected Output	Result	Remark
1	Enter valid Name and password	Accepted	Accepted	Pass	Correct details

2	Enter Invalid Name and Password	Not Accepted	Not Accepted	Fail	Invalid details

Book Entry Form:

Sno	Test case	Actual output	Expected Output	Result	Remark
1	Add a book	Accepted	Accepted	Pass	At first user have to fill all fields with proper data, if any Error like entering text data instead of number or entering number instead of text is found then it gives proper message otherwise Adds Record To the Database
2	Delete Book	Accepted	Accepted	Pass	This deletes the details of book by using Accession no.
3	Search Book	Accepted	Accepted	Pass	Modified records are Updated in database.
4	Edit Book	Accepted	Accepted	Pass	Displays the Details of book for entered Accession no. Otherwise gives proper Error message.
5	Exit	Accepted	Accepted	Pass	Exits Current book.

Student Entry Form:

Sno	Test case	Actual output	Expected Output	Result	Remark
1	Add details	Accepted	Accepted	Pass	At first user have to fill all fields with proper data , if any Error like entering text data instead of number or entering number instead of text is found then it
					gives proper message otherwise Adds Record To the Database
2	Delete details	Accepted	Accepted	Pass	This deletes the details of user by using Accession no.
3	Search details	Accepted	Accepted	Pass	Modified user records are Updated in database.
4	Edit details	Accepted	Accepted	Pass	Displays the Details of user for entered Accession no. Otherwise gives proper Error message.
5	Exit	Accepted	Accepted	Pass	Exits from the current user

Book Issue and return Form:

Sno	Test case	Actual output	Expected Output	Result	Remark
1	Issue Book (Valid Data)	Accepted	Accepted	Pass	At first user have to fill all fields with proper data. System validates data and creates a new record in the Issue table.

2	Issue Book (Invalid Book ID)	Not Accepted	Not Accepted	Fail	Invalid book
3	Search Book	Accepted	Accepted	Pass	User enters Book ID. System searches Issue records and displays details of matching issued books.
4	Return Book (Valid Data)	Accepted	Accepted	Pass	System checks if Book ID exists in the Issue table.
5	Exit	Accepted	Accepted	Pass	Exits Current book.

ASSIGNMENT – TASK 9

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

❖ **SOURCE CODE:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[3][3],b[3][3],c[3][3], i,  j, k, m, n, p, q;
    printf("Enter matrix no. of rows & cols:");
    scanf("%d%d",&m,&n);
    printf("Enter 2matrix no.of rows & cols") ;
    scanf("%d%d",&p,&q);
    printf("\n enter the matrix elements");
    for(i=0; i< m; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
  printf("\n a matrix is\n");
  for(i=0;i<m ;i++)
  {
     for(j=0; j<n; j++)
    {
        printf("%d\t", a[i][j]);
     }
```

```
        printf("\n");
       }
     for(i=0;i<p; i++)
     {
         for(j=0; j<q; j++)
        {
           scanf("%d\t", &b[i][j]);
        }
     }
  printf("\n b matrix is\n");
  for(i=0; i<p; i++)
 {
    for(j=0; j<q; j++)
   {
       printf("%d\t", b[i][j]);
   }
 printf("\n");
 }
 for(i=0; i<m; i++)
 {
     for(j=0; j<q; j++)
    {
            c[i][j]=0;
            for(k=0;k<n;k++)
           {
               c[i][j]=c[i][j]+a[i][k]*b[k][j];
           }
    }
for(i=0; i<m; i++)
 {
```

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

INPUT AND OUTPUT:

Input	Actual Output
Matrix1: 1 1 1 1 1 1 1 1 1 Matrix2: 1 1 1 1 1 1 1 1 1	3 3 3 3 3 3 3 3 3

Test cases:

Test case no: 1

Test case name: Equal no. of rows & cols

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

Test case no: 2

Test case name: Cols of 1st matrix is not equal to rows of 2nd matrix.

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols = 2 2 Matrix2 rows & cols = 3 2 Matrix1: 1 1 1 1 Matrix2: 1 1 1 1 1 1	Operation can't be performed	2 2 2 2	fail

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols = 2 2 Matrix2 rows & cols = 2 2 Matrix1: 1234567891 2222222222 2234567891 2222222221 Matrix2: 234567891 22222221533 213242424 56456475457		1234567891 -2072745074 -2060399405 -2072745075 234567891 747385053 213142424 621900609	fail

ASSIGNMENT – TASK 13

NAME: B. Sindhuja
ROLL.NO: 2103A51085
BATCH: 21CSBTB05

EXPERIMENT: 13

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
DEPARTMENT OF CSE
SOFTWARE TESTING METHODOLOGY Page 54 suggestions while typing the email ids based on the existing email Ids in user’s email list.
3. Verify that the user can enter multiple comma-separated email Ids 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 email Ids 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 email Ids 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.

Test Case Results for Gmail Inbox Functionality

S. No	Test Case	Testing	Observation	Result
1	Verify that a newly received email is displayed as highlighted in the Inbox section.	Send a new email to the Gmail account.	Check if the newly received email appears at the top of the Inbox section with a highlighted or distinct visual indicator.	**Pass if the email is highlighted, indicating it's new. Fail if it is not highlighted.**
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).	Send an email with a specific sender, subject, and body content.	Check if the sender's email ID or name, mail subject, and a trimmed version of the mail body (limited to a single line) are displayed correctly.	**Pass if all the details are displayed accurately. Fail if it is not displayed.**
3	Verify that on clicking the newly received email, the user is navigated to email content.	Click on the newly received email.	Verify that clicking on the email navigates the user to the email content.	**Pass if the user is directed to the email content. Fail if it is not directed.**
4	Verify that the email contents are correctly displayed with the desired source formatting.	Send an email with various formatting elements (e.g., text styles, links, images).	Check if the email content displays all formatting elements correctly as intended.	**Pass if the email content renders with the desired source formatting. Fail if it is not formatted.**
5	Verify that any attachments are attached to the email and are downloadable.	Send an email with attachments (e.g., documents, images).	Check if the attachments are visible and downloadable from the email.	**Pass if the attachments are present and downloadable. Fail if it is not present and downloadable.**
6	Verify that the attachments are scanned for viruses before download.	Send an email with an attachment containing a known virus.	Check if the system scans the attachment for viruses before allowing download.	**Pass if the system detects and prevents the download of the infected attachment. Fail if it is not attached.**

7	Verify that all the emails marked as read are not highlighted.	Open and read a newly received email.	Check if the email is no longer highlighted after being marked as read.	**Pass if the email is no longer highlighted after being read.** **Fail if it is not being read.**
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.	Open multiple emails, mark them as read, and observe the email list.	Check if a timestamp indicating the read time is appended to both read and unread emails.	**Pass if the timestamp is appended to all emails.** **Fail if it is not timestamp.**
9	Verify that the count of unread emails is displayed alongside the 'Inbox' text in the left sidebar of Gmail.	Send multiple unread emails to the Gmail account.	Check if the count of unread emails is displayed next to the 'Inbox' label.	**Pass if the count is displayed accurately.** **Fail if it is not displayed accurately.**
10	Verify that the unread email count increases by one on receiving a new email.	Send a new email to the Gmail account.	Check if the unread email count increments by one after receiving the new email.	**Pass if the unread email count increases by one.** **Fail if it not increased by one.**
11	Verify that the unread email count decreases by one on reading an email (marking an email as read).	Open and read a previously unread email.	Check if the unread email count decreases by one after marking the email as read.	**Pass if the unread email count decreases by one.** **Fail if it is not decreased by one.**
12	Verify that email recipients in CC are visible to all users.	Send an email with recipients in the CC field.	Check if the recipients in the CC field are visible to all users.	**Pass if the CC recipients are visible.** **Fail if it not visible.**
13	Verify that email recipients in BCC are not visible to the user.	Send an email with recipients in the BCC field.	Check if the recipients in the BCC field are not visible to the user.	**Pass if the BCC recipients are not visible.** **Fail if it is visible.**

14	Verify that all received emails get piled up in the 'Inbox' section and get deleted in cyclic fashion based on the size availability	Send emails until the Inbox reaches its size limit.	Check if the oldest emails are automatically deleted to make space for new ones when the Inbox reaches its size limit.	**Pass if emails are deleted in a cyclic fashion based on size availability.** **Fail if it is not deleted in cyclic fashion.**
15	Verify that emails can be received from non-Gmail email IDs like Yahoo, Hotmail, etc.	Send emails from non-Gmail email IDs (e.g., Yahoo, Hotmail) to the Gmail account.	Check if emails from non-Gmail email IDs are received and displayed in the Inbox.	**Pass if emails from non-Gmail email IDs are received successfully.** **Fail if it is not received successfully.**

Test Cases for GMail – Compose Mail Functionality

S.NO	Test Case	Testing	Observation	Result
1	Verify that on clicking 'Compose' button, a frame to compose a mail gets displayed.	Click the 'Compose' button.	Check if a frame or window to compose a new email is displayed.	**Pass/Fail**
2	Verify that the user can enter email IDs in 'To', 'cc', and 'bcc' sections and also get suggestions while typing based on existing email IDs in the user's email list.	Enter email IDs in the 'To', 'cc', and 'bcc' fields.	Check if the user receives suggestions while typing email IDs based on existing email IDs in the user's email list.	**Pass/Fail**
3	Verify that the user can enter multiple comma-separated email IDs in 'To', 'cc', and 'bcc' sections.	Enter multiple comma-separated email IDs in the 'To', 'cc', and 'bcc' fields.	Check if the system accepts multiple comma-separated email IDs in each section.	**Pass/Fail**
4	Verify that the user can type the Subject line in the 'Subject' textbox.	Type a subject line in the 'Subject' textbox.	Check if the user can input text in the 'Subject' field.	**Pass/Fail**
5	Verify that the user can type the email in the email-body section.	Type an email in the email-body section.	Check if the user can input text in the email-body section.	**Pass/Fail**
6	Verify that users can format mail using editor-options provided like choosing font-family, font-size, bold-italic-underline, etc.	Use editor options to format the email (e.g., font- family, font-size, bold, italic, underline).	Check if the formatting options are available and working as expected.	**Pass/Fail**
7	Verify that the user can attach a file as an attachment to the email.	Click on the attachment icon/ button and attach a file.	Check if the file is successfully attached to the email.	**Pass/Fail**
8	Verify that the user can add images in the email and select the size for the same.	Insert an image into the email and select its size.	Check if the image is successfully added and if the size can be adjusted.	**Pass/Fail**

9	Verify that after entering email IDs 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.	Enter email IDs, subject line, and mail body, then click 'Send'.	Check if the email is successfully sent to the intended recipients.	**Pass/Fail**
10	Verify that sent mails can be found in 'Sent Mail' section of the sender.	Send an email and check the 'Sent Mail' section.	Check if the sent email is listed in the 'Sent Mail' section.	**Pass/Fail**
11	Verify that mail can be sent to non-Gmail email IDs also.	Enter a non-Gmail email ID in the recipient field and send the email.	Check if the email is successfully delivered to the non-Gmail email ID.	**Pass/Fail**
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.	Send emails until the 'Sent Mail' section reaches its size limit.	Check if the oldest emails are automatically deleted to make space for new ones when the 'Sent Mail' section reaches its size limit.	**Pass/Fail**
13	Verify that the emails composed but not sent remain in the draft section.	Compose an email but do not send it.	Check if the email remains in the draft section.	**Pass/Fail**
14	Verify the maximum number of email recipients that can be entered in 'To', 'cc', and 'bcc' sections.	Enter a large number of email IDs in the recipient fields.	Check if there is a limit to the number of email recipients that can be entered.	**Pass/Fail**
15	Verify the maximum length of text that can be entered in the 'Subject' textbox.	Enter a long subject line.	Check if there is a limit to the length of the subject line.	**Pass/Fail**
16	Verify the content limit of text/images that can be entered and successfully delivered as mail body.	Enter a large amount of text/ images in the email body.	Check if there is a limit to the content size that can be successfully sent.	**Pass/Fail**

17	Verify the maximum size and number of attachments that can be attached with an email.	Attach large files and multiple attachments to an email.	Check if there are limits to the size and number of attachments.	**Pass/Fail**
18	Verify that only the allowed specifications of the attachment can be attached with an email.	Attach files of various formats and sizes.	Check if the system accepts only allowed specifications of attachments.	**Pass/Fail**
19	Verify that if the email is sent without a Subject, a pop-up is generated warning the user about no subject line. Also, verify that on accepting the pop-up message, the user is able to send the email.	Send an email without a subject line.	Check if a pop-up warning about the missing subject line is generated and if the email can still be sent after accepting the warning.	**Pass/Fail**