S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 9

Aim: Identify and design the Test Suites for the given problem definition.

Name of Student	
Roll No	
Semester/Year	V/III
Academic Session	2024-25
Date of Performance	
Date of Submission	

AIM: Identify and design the Test Suites for the given problem definition.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

- Learn about different techniques of testing a software
- Design unit test cases to verify the functionality and locate bugs, if any

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement

- Processor: Dual Core
- RAM: 1GB
- Hard Disk Drive: > 80 GB

Software Requirement

- Operating System – Windows

THEORY

Software Testing

Testing software is an important part of the development life cycle of a software. It is an expensive activity. Hence, appropriate testing methods are necessary for ensuring the reliability of a program. According to the ANSI/IEEE 1059 standard, the definition of testing is the process of analysing a software item, to detect the differences between existing and required conditions i.e. defects/errors/bugs and to evaluate the features of the software item.

The purpose of testing is to verify and validate a software and to find the defects present in a software. The purpose of finding those problems is to get them fixed.

- Verification is the checking or we can say the testing of software for consistency and conformance by evaluating the results against pre-specified requirements.
- Validation looks at the systems correctness, i.e. the process of checking that what has been specified is what the user actually wanted.
- Defect is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

Testing Frameworks

Following are the different testing frameworks:

- jUnit - for Java unit test
- Selenium - is a suite of tools for automating web applications for software testing purposes, plugin for Firefox
- HP QC - is the HP Web-based test management tool. It familiarizes with the process of defining releases, specifying requirements, planning tests, executing tests, tracking defects, alerting on changes, and analyzing results. It also shows how to customize project
- IBM Rational - Rational software has a solution to support business sector for designing, implementing and testing software

Need for Software Testing

There are many reasons for why we should test software, such as:

- Software testing identifies the software faults. The removal of faults helps reduce the number of system failures. Reducing failures improves the reliability and the quality of the systems.
- Software testing can also improve the other system qualities such as maintainability, usability, and testability.
- In order to meet the condition that the last few years of the 20th century systems had to be shown to be free from the 'millennium bug'.
- In order to meet the different legal requirements.
- In order to meet industry specific standards such as the Aerospace, Missile and Railway Signalling standards.

Test Cases and Test Suite

A test case describes an input description and an expected output descriptions. Input are of two types: preconditions (circumstances that hold prior to test case execution) and the actual inputs that are identified by some testing methods. The set of test cases is called a test suite. We may have a test suite of all possible test cases.

Types of Software Testing

Testing is done in every stage of software development life cycle, but the testing done at each level of software development is different in nature and has different objectives. There are different types of testing, such as stress testing, volume testing, configuration testing, compatibility testing, recovery testing, maintenance testing, documentation testing, and usability testing. Software testing are mainly of following types [1]

1. Unit Testing
2. Integration Testing
3. System Testing

1. Unit Testing

Unit testing is done at the lowest level. It tests the basic unit of software, that is the smallest testable piece of software. The individual component or unit of a program are tested in unit testing. Unit testing are of two types.

- **Black box testing**:
  This is also known as functional testing, where the test cases are designed based on input output values only. There are many types of Black Box Testing but following are the prominent ones.

- **Equivalence class partitioning:** In this approach, the domain of input values to a program is divided into a set of equivalence classes. e.g., Consider a software program that computes whether an integer number is even or not that is in the range of 0 to 10. Determine the equivalence class test suite. There are three equivalence classes for this program. - The set of negative integers - The integers in the range 0 to 10 - The integer larger than 10

- **Boundary value analysis:** In this approach, while designing the test cases, the values at boundaries of different equivalence classes are taken into consideration. e.g. In the above given example as in equivalence class partitioning, a boundary values based test suite is { 0, -1, 10, 11 }

- **White box testing:**
  It is also known as structural testing. In this testing, test cases are designed on the basis of examination of the code. This testing is performed based on the knowledge of how the system is implemented. It includes analysing data flow, control flow, information flow, coding practices, exception and error handling within the system, to test the intended and unintended software behaviour. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities. This testing requires access to the source code. Though white box testing can be performed any time in the life cycle after the code is developed, but it is a good practice to perform white box testing during the unit testing phase.

Integration Testing

Integration testing is performed when two or more tested units are combined into a larger structure. The main objective of this testing is to check whether the different modules of a program interface with each other properly or not. This testing is mainly of two types:

- Top-down approach
- Bottom-up approach

In bottom-up approach, each subsystem is tested separately and then the full system is tested. But the top-down integration testing starts with the main routine and one or two subordinate routines in the system. After the top-level 'skeleton' has been tested, the immediately subroutines of the 'skeleton' are combined with it and tested.

System Testing

System testing tends to affirm the end-to-end quality of the entire system. System testing is often based on the functional / requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability are also checked. There are three types of system testing

- Alpha testing is done by the developers who develop the software. This testing is also done by the client or an outsider with the presence of developer or we can say tester.
- Beta testing is done by very few numbers of end users before the delivery, where the change requests are fixed, if the user gives any feedback or reports any type of defect.
- User Acceptance testing is also another level of the system testing process where the system is tested for acceptability. This test evaluates the system's compliance with the client requirements and assess whether it is acceptable for software delivery

An error correction may introduce new errors. Therefore, after every round of error-fixing, another testing is carried out, i.e., called regression testing. Regression testing does not belong to either unit testing, integration testing, or system testing, instead, it is a separate dimension to these three forms of testing.

Regression Testing

The purpose of regression testing is to ensure that bug fixes and new functionality introduced in a software do not adversely affect the unmodified parts of the program [2]. Regression testing is an important activity at both testing and maintenance phases. When a piece of software is modified, it is necessary to ensure that the quality of the software is preserved. To this end, regression testing is to retest the software using the test cases selected from the original test suite.

Example

```
#include <stdio.h>

int
main()
{
    int n, res;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n >= 1 && n <= 100)
    {
        res = n * n;
        printf("\n Square of %d is %d\n", n, res);
    }
    else if (n<= 0 || n > 100)
        printf("Beyond the range");

    return 0;
}
```

Output

```
Inputs              Outputs
I1 : -2        O1 :  Beyond the range
I2 :  0        O2 :  Beyond the range
I3 :  1        O3 :  Square of 1 is 1
I4 : 100       O4 :  Square of 100 is 10000
I5 : 101    O5 :  Beyond the range
I6 : 4      O6 :  Square of 4 is 16
I7 : 62        O7 :  Square of 62 is 3844
```

Test Cases

```
T1 : {I1 ,O1}
T2 : {I2 ,O2}
T3 : {I3, O3}
T4 : {I4, O4}
T5 : {I5, O5}
T6 : {I6, O6}
T7 : {I7, O7}
```

Case Study

1 : A Library Information System for SE VLabs Institute

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.
As already discussed under the theory section, test case preparation could begin right after requirements identification stage. It is desirable (and advisable) to create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s). A simplified form of the RTM is shown in table 1 (the numbers shown in this table are arbitrary; not specific to LIS).
Table 1: A simplified mapping from requirements to test cases

Requirement #	Test Case #
R1	TC1
R2	TC2, TC3, TC4
R3	TC5
R4	TC6

Table 1 states which test case should help us to verify that a specified feature has been implemented and working correctly. For instance, if test case # TC6 fails, that would indicate requirement # R4 has not fully realized yet. Note that it is possible that a particular requirement might need multiple test cases to verify whether it has been implemented correctly.

To be specific to our problem, let us see how we can design test cases to verify the "User Login" feature. The simplest scenario is when both user name and password have been typed in correctly. The outcome will be that the user could then avail all features of LIS. However, there could be multiple unsuccessful conditions:

- User ID is wrong
- Password is wrong
- User ID & password are wrong
- Wrong password given twice consecutively
- Wrong password given thrice consecutively
- Wrong password given thrice consecutively, and security question answered correctly
- Wrong password given thrice consecutively, and security question answered incorrectly

We would create test case for the above stated login scenarios. These test cases together would constitute a test suite to verify the concerned requirement. Table 2 shows the details of this test suite.

Table 2: A test suite to verify the "User Login" feature

#	TS1
Title	Verify "User Login" functionality
Description	To test the different scenarios that might arise while an user is trying to login

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC 1	Verify that user already registered with the LIS is able to login with correct user ID and password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is logged in	1. Type in employee ID as *149405* 2. Type in password *this_is_password* 3. Click on the 'Login' button	"Home" page for the user is displayed
TC 2	Verify that an unregistered user of LIS is unable to login		Employee ID *149405xx* is not a registered user of LIS	User is not logged in	1. Type in employee ID as *149405xx* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC 3	Verify that user already registered with the LIS is unable to login with incorrect		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message

#	TS1
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
	password					
TC 4	Verify that user already registered with the LIS is unable to login with incorrect password given twice consecutively	TC3	This test case is executed after execution of TC3 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever2* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC 5	Verify that user already registered with the LIS is unable to login with incorrect password given thrice consecutively	TC4	This test case is executed after execution of TC4 before executing any other test case	User is not logged in	*1.* Type in employee ID as *149405* *2.* Type in password *whatever3* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message; the security question and input box for the answer are displayed
TC 6	Verify that a registered user can login after three consecutive failures by correctly answering the security question	TC5	This test case is executed after execution of TC6 before executing any other test case. Answer to the security question is *my_answer*.	Email sent containing new password. The email is expected to be received within 2 minute.	1. Type in the answer as *my_answer* 2. Click on the 'Email Password' button	Login dialog is displayed; an email containing the new password is received
TC 7	Verify that a registered user's account is blocked after three		Execute the test cases TC3, TC4, and TC5 once again (in order) before executing this test case	User account has been blocked	1. Type in the answer as *not_my_answer* 2. Click on the 'Email	The message *"Your account has been blocked! Please contact the administrator."* appears

#	TS1
Title	Verify "User Login" functionality
Description	To test the different scenarios that might arise while an user is trying to login

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
	consecutive failures and answering the security question incorrectly				Password' button	

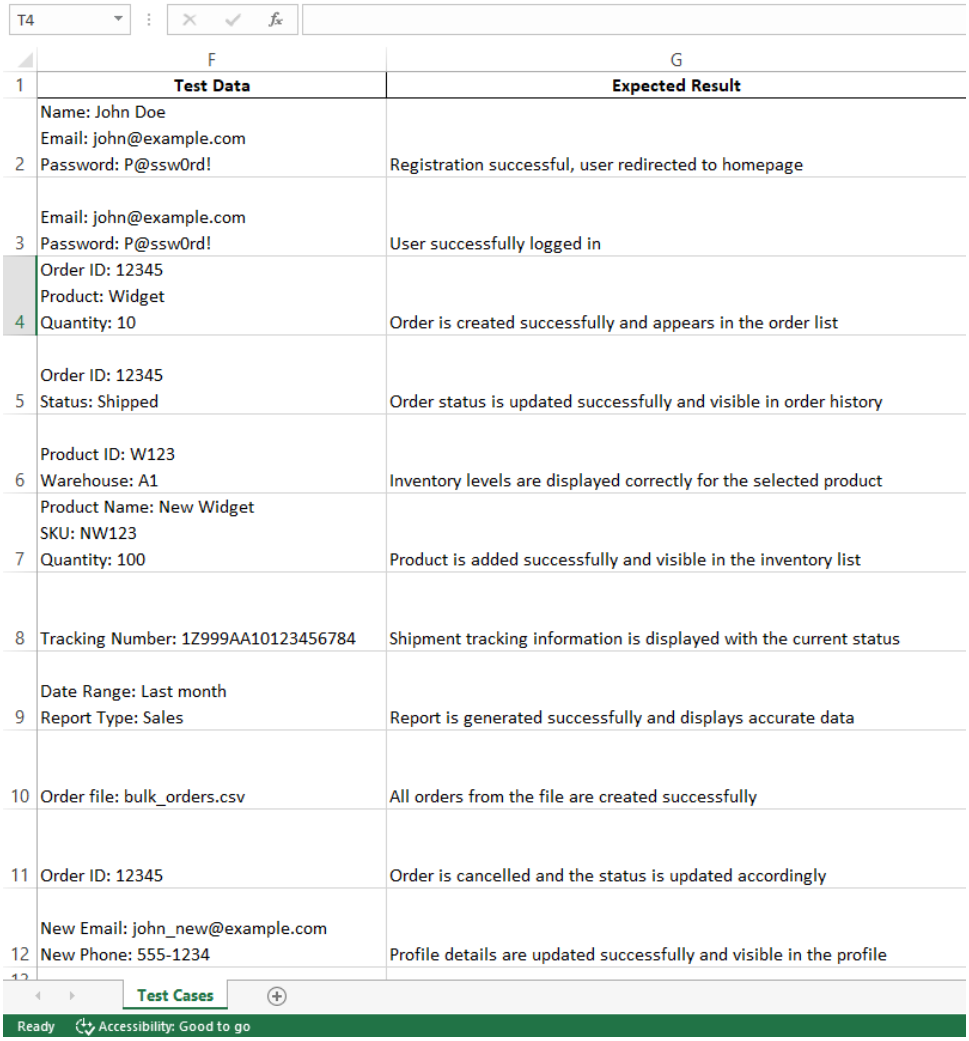
OBSERVATION:

T4

	A	B	C	D	E
1	Test Case ID	Test Case Name	Description	Preconditions	Test Steps
2	TC 1	User Registration	Verify that a user can register with valid credentials	User not logged in	1. Open registration page 2. Enter valid user details 3. Click Register
3	TC 2	User Login	Verify that a registered user can log in with valid credentials	User has registered	1. Open login page 2. Enter valid credentials 3. Click Login
4	TC 3	Create New Order	Verify that a user can create a new order with all required information	User is logged in	1. Navigate to 'Create Order' page 2. Enter order details 3. Submit order
5	TC 4	Update Order Status	Test if an admin can update the order status to different stages	Admin is logged in	1. Navigate to 'Order Management' page 2. Select an order 3. Update status to 'Shipped'
6	TC 5	Inventory Check	Verify that inventory levels can be checked for a given product	Inventory data is available	1. Navigate to 'Inventory' page 2. Search for a product 3. Check stock level
7	TC 6	Add New Product	Test that an admin can add a new product to the inventory	Admin is logged in	1. Navigate to 'Add Product' page 2. Enter product details 3. Click 'Add'
8	TC 7	Shipment Tracking	Ensure that a shipment can be tracked using a valid tracking number	Valid tracking number is available	1. Navigate to 'Track Shipment' page 2. Enter tracking number 3. Click 'Track'
9	TC 8	Generate Reports	Verify that the system can generate accurate reports based on sales data	Admin is logged in	1. Navigate to 'Reports' page 2. Select report criteria 3. Click 'Generate Report'
10	TC 10	Bulk Order Creation	Ensure that a user can create multiple orders in a single action	User is logged in	1. Navigate to 'Bulk Order' page 2. Upload order file 3. Submit the order batch
11	TC 11	Order Cancellation	Verify that a user can cancel an order within the allowed time frame	Order is placed and within cancellation period	1. Navigate to 'Order History' page 2. Select an order 3. Click 'Cancel'
12	TC 12	User Profile Update	Test that a user can update their profile information	User is logged in	1. Navigate to 'Profile' page 2. Update profile details 3. Click 'Save'

Test Cases

Ready Accessibility: Good to go



	F	G
1	**Test Data**	**Expected Result**
2	Name: John Doe Email: john@example.com Password: P@ssw0rd!	Registration successful, user redirected to homepage
3	Email: john@example.com Password: P@ssw0rd!	User successfully logged in
4	Order ID: 12345 Product: Widget Quantity: 10	Order is created successfully and appears in the order list
5	Order ID: 12345 Status: Shipped	Order status is updated successfully and visible in order history
6	Product ID: W123 Warehouse: A1	Inventory levels are displayed correctly for the selected product
7	Product Name: New Widget SKU: NW123 Quantity: 100	Product is added successfully and visible in the inventory list
8	Tracking Number: 1Z999AA10123456784	Shipment tracking information is displayed with the current status
9	Date Range: Last month Report Type: Sales	Report is generated successfully and displays accurate data
10	Order file: bulk_orders.csv	All orders from the file are created successfully
11	Order ID: 12345	Order is cancelled and the status is updated accordingly
12	New Email: john_new@example.com New Phone: 555-1234	Profile details are updated successfully and visible in the profile

Test Cases

Ready Accessibility: Good to go

CONCLUSION:

DISCUSSION QUESTIONS

1) Define Software Testing & explain the importance of Software Testing.

2) List the different Software Testing Frameworks.

3) List the different types of Testing and explain any 3.

4) Distinguish between Black Box and White Box Testing.

Sr.No.	Black Box Testing	White Box Testing
1)		
2)		
3)		
4)		
5)		

5) List the different types of Black Box Testing.

6) Define Test Data and explain its role in software testing**.**

REFERENCES:

- http://vlabs.iitkgp.ernet.in/se/10
- https://en.wikipedia.org/wiki/Software_test_documentation
- http://www.doqs.com/pdf/DOQSTestPlanning.pdf