7. Developing Test Cases

Aim: To develop test cases for testing.

Description:

Software Testing

Testing software is an important part of the development life cycle of a software. According to the ANSI/IEEE 1059 standard, the definition of testing is the process of analyzing 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

Test Cases and Test Suite

A test case describes an input descriptions 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. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Validation Testing**

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 integer - 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 analyzing data flow, control flow, information flow, coding practices, exception and error handling within the system, to test the intended and unintended software behavior. 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.

Validation Testing

There are three types of validation 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 number 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 validation 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

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. 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

Write a program to calculate the square of a number in the range 1-100

```
#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}
```

Test Cases for Book Bank System:

Test case preparation could begin right after requirements identification stage. 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 Book Bank. 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 1 shows the details of this test suite.

Table 1: 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
TC1	Verify that user already registered with the Book Bank (BB) is able to login with correct user ID and password		Employee ID *149405* is a registered user of BB; 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
TC2	Verify that an unregistered user of BB is unable to login		Employee ID *149405xx* is not a registered user of BB	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
TC3	Verify that user already registered with the BB is unable to login with incorrect password		Employee ID *149405* is a registered user of BB; 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
TC4	Verify that user already registered with the BB 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

#	**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
TC5	Verify that user already registered with the BB 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
TC6	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
TC7	Verify that a registered user's account is blocked after three consecutive failures and answering the security question incorrectly		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 Password' button	The message *"Your account has been blocked! Please contact the administrator."* appears