Department of
Computer Science and Engineering

Title: Test cases generation to evaluate the performance of the given project

Integrated Design Project II

CSE 406

Green University of Bangladesh

1 Objective(s)

- Understand different software testing tools and their features
- Manage the project from beginning to end
- Define, formulate and analyze a problem
- To learn how to write software testing documents, and communicate with engineers invidious forms.
- To gain the techniques and skills on how to use modern software testing tools to support software testing projects
- Execute how to run test script wizard and Execute how to do performance testing using testing tools including Selenium and JMeter respectively

2 Introduction

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

Test cases are typically written by members of the quality assurance (QA) team or the testing team and can be used as step-by-step instructions for each system test. Testing begins once the development team has finished a system feature or set of features. A sequence or collection of test cases is called a test suite.

A test case document includes test steps, test data, preconditions and the post conditions that verify requirements.

2.1 Why test cases are important

Test cases define what must be done to test a system, including the steps executed in the system, the input data values that are entered into the system and the results that are expected throughout test case execution. Using test cases allows developers and testers to discover errors that may have occurred during development or defects that were missed during ad hoc tests. The benefits of an effective test case include:

- Guaranteed good test coverage.
- Reduced maintenance and software support costs.
- Reusable test cases.
- Confirmation that the software satisfies end-user requirements.
- Improved quality of software and user experience.
- Higher quality products lead to more satisfied customers.
- More satisfied customers will increase company profits.

2.2 Example of test case format

Test cases must be designed to fully reflect the software application features and functionality under evaluation. QA engineers should write test cases so only one thing is tested at a time. The language used to write a test case should be simple and easy to understand, active instead of passive, and exact and consistent when naming elements.

The components of a test case include:

- **Test name.** A title that describes the functionality or feature that the test is verifying.
- **Test ID.** Typically a numeric or alphanumeric identifier that QA engineers and testers use to group test cases into test suites.
- **Objective.** Also called the description, this important component describes what the test intends to verify in one to two sentences.
- **References.** Links to user stories, design specifications or requirements that the test is expected to verify.
- **Prerequisites.** Any conditions that are necessary for the tester or QA engineer to perform the test.
- **Test setup.** This component identifies what the test case needs to run correctly, such as app version, operation system, date and time requirements and security specifications.
- **Test steps.** Detailed descriptions of the sequential actions that must be taken to complete the test.
- **Expected results.** An outline of how the system should respond to each test step.

Before writing a test case, QA engineers and testing team members should first determine the scope and purpose of the test. This includes understanding the system features and user requirements as well as identifying the testable requirements.

Next, testers should define how the software testing activities are performed. This process starts by identifying effective test case scenarios – or functionality that can be tested. In order to identify test case scenarios, testers must understand the functional requirements of the system.

3 Test case writing best practices

An effective test case design will be:

- Accurate, or specific about the purpose.
- Economical, meaning no unnecessary steps or words are used.
- Traceable, meaning requirements can be traced.
- Repeatable, meaning the document can be used to perform the test numerous times.
- Reusable, meaning the document can be reused to successfully perform the test again in the future.

4 Types of test cases

- **Functionality test cases.** This is a type of black box testing that can reveal if an app's interface works with the rest of the system and its users by identifying whether the functions that the software is expected to perform are a success or failure. Functionality test cases are based on system specifications or user stories, allowing tests to be performed without accessing the internal structures of the software. This test case is usually written by the QA team.

- **Performance test cases.** These test cases can help validate response times and confirm the overall effectiveness of the system. Performance test cases include a very strict set of success criteria and can be used to understand how the system will operate in the real world. Performance test cases are typically written by the testing team, but they are often automated because one system can demand hundreds of thousands of performance tests.

- **Unit test cases.** Unit testing involves analyzing individual units or components of the software to confirm each unit performs as expected. A unit is the smallest testable element of software. It often takes a few inputs to produce a single output.

- **User interface test cases.** This type of test case can verify that specific element of the graphical user interface (GUI) look and perform as expected. UI test cases can reveal errors in elements that the user interacts with, such as grammar and spelling errors, broken links and cosmetic inconsistencies. UI tests often require cross-browser functionality to ensure an app performs consistently across different browsers. These test cases are usually written by the testing team with some help from the design team.

- **Security test cases.** These test cases are used to confirm that the system restricts actions and permissions when necessary to protect data. Security tests cases often focus on authentication and encryption and frequently use security-based tests, such as penetration testing. The security team is responsible for writing these test cases – if one exists in the organization.

- **Integration test cases.** An integration test case is written to determine how the different software modules interact with each other. The main purpose of this test case is to confirm that the interfaces between different modules work correctly. Integration test cases are typically written by the testing team, with input provided by the development team

- **Database test cases.** This type of test case aims to examine what is happening internally, helping testers understand where the data is going in the system. Testing teams frequently use SQL queries to write database test cases.

- **Usability test cases.** A usability test case can be used to reveal how users naturally approach and use an application. Instead of providing step-by-step details, a usability test case will provide the tester with a high-level scenario or task to complete. These test cases are typically written by the design and testing teams and should be performed before user acceptance testing.

- **User acceptance test cases.** These test cases focus on analyzing the user acceptance testing environment. They are broad enough to cover the entire system and their purpose is to verify if the application is acceptable to the user. User acceptance test cases are prepared by the testing team or product manager and then used by the end user or client. These tests are often the last step before the system goes to production.

- **Regression testing.** This test confirms recent code or program changes have not affected existing system features. Regression testing involves selecting all or some of the executed test cases and running them again to confirm the software's existing functionalities still perform appropriately.

4.1 Test script vs. test case vs. test scenario

The key differences between a test case and a test scenario include:

- A test case provides a set of actions performed to verify that specific software features are performing correctly. A test scenario is any feature that can be tested.
- A test case is beneficial in exhaustive testing – a software testing approach that involves testing every possible data combination.
- A test scenario is more agile and focuses on the end-to-end functionality of the software.
- A test case looks at what to test and how to test it while a test scenario only identifies what to test.
- A test case requires more resources and time for test execution than a test scenario.
- A test case includes information such as test steps, expected results and data while a test scenario only includes the functionality to be tested.

A test script is a line-by-line description of all the actions and data needed to properly perform a test. The script includes detailed explanations of the steps needed to achieve a specific goal within the program, as well as descriptions of the results that are expected for each step.

On the other hand, a test case describes the idea that is to be tested; it does not detail the exact steps to be taken. Therefore, test scripts are more detailed testing documents than test cases, but test cases are more detailed than test scenarios.

5 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.

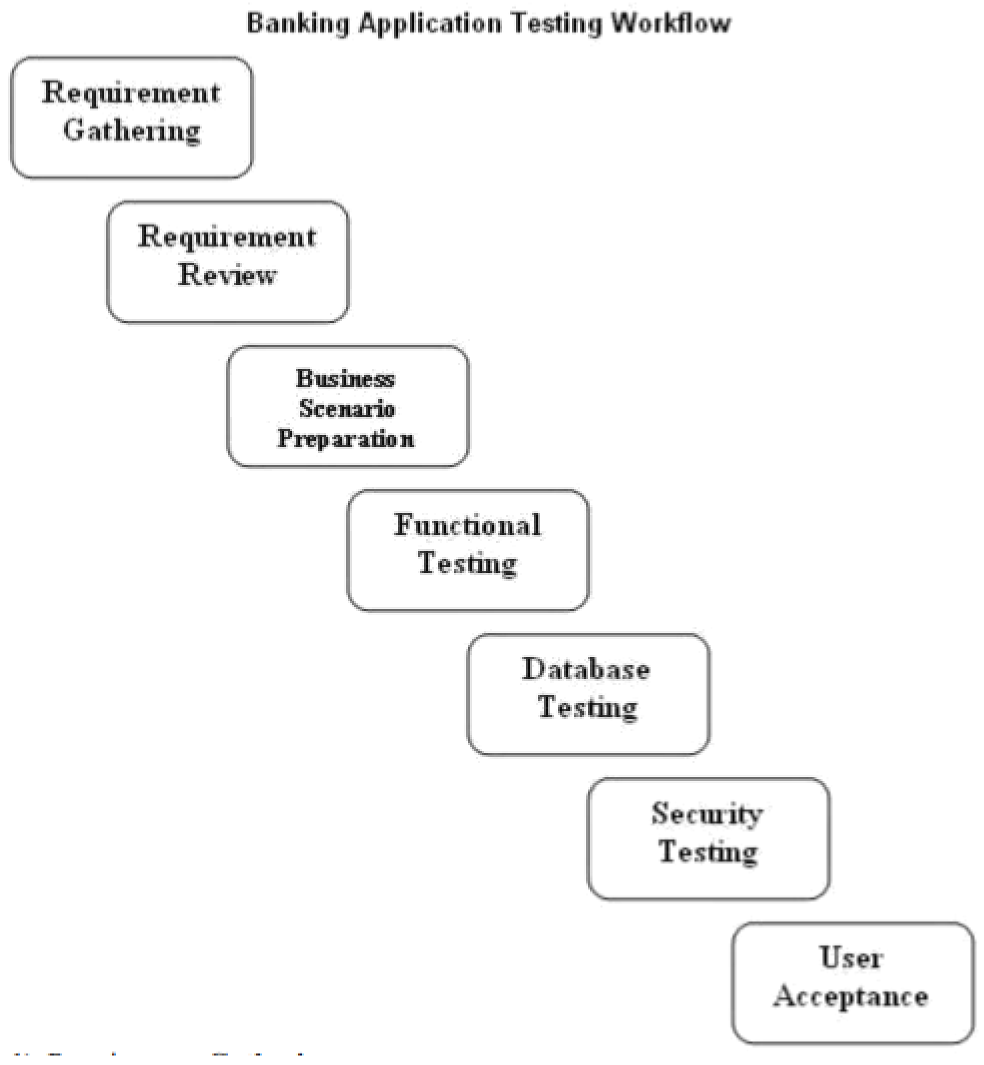

Figure 1

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

5.1 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

5.2 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 types
- Rules Testing
- Testing Stored Procedures and Functions
- Testing Triggers
- Data Integrity

6 Some possible Test cases for Bank System

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

7 Discussion & Conclusion

Based on the focused objective(s), to understand about test cases. Various type of test cases and their application and use in proposed project. Student can think test cares and can test their system accordingly.

8 Lab Task (Please implement yourself and show the output to the instructor)

1. Prepare various test cases based on your proposed project
2. Examine your Input and output test cases.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.