SOFTWARE TESTING LIFE CYCLE

Table of Contents

1 Software Testing Life Cycle: 4
1.1 Requirement Analysis Phase: 4
1.2 Test Planning Phase: 5
1.3 Test Design Phase: 5
1.4 Test Environment Setup Phase: 5
1.5 Test Execution Phase: 5
1.6 Defect Reporting and Tracking Phase: 5
1.7 Regression Testing Phase: 5
1.8 Release Readiness Phase: 6
1.9 Deployment and Post-Deployment Phase: 6
2 Test Cases in STLC: 6
2.1 Test Scenario Identification: 6
2.2 Test Case Design: 6
2.3 Test Case Components: 6
2.4 Positive and Negative Test Cases: 7
2.5 Boundary Value Test Cases: 7
2.6 Equivalence Partitioning: 7
2.7 Data-Driven Test Cases: 7
2.8 Test Case Execution: 7
2.9 Defect Reporting: 8
2.10 Regression Testing: 8
2.11 Maintenance and Updates: 8
3 Example test case for Face book 8
3.1 User Registration: 8
3.1.1 Test Case 1: Verify User Registration with Valid Data 8
3.1.2 Test Case 2: Verify User Registration with Invalid Email 8
3.2 Login and Authentication: 9
3.2.1 Test Case 3: Verify Successful Login 9
3.2.2 Test Case 4: Verify Failed Login with Incorrect Password 9
3.3 Profile Management: 9

3.3.1 Test Case 5: Verify Profile Picture Upload 9
3.3.2 Test Case 6: Verify User Status Update 9
3.4 Friend Requests: 9
3.4.1 Test Case 7: Verify Sending a Friend Request 9
3.4.2 Test Case 8: Verify Accepting a Friend Request 9
3.5 Privacy and Security: 9
3.5.1 Test Case 9: Verify Changing Password 9
3.5.2 Test Case 10: Verify Privacy Settings 10
3.6 Notifications: 10
3.6.1 Test Case 11: Verify Notification for New Friend Request 10
3.6.2 Test Case 12: Verify Notification for Message 10
4 Conclusion 11

List of Figures

Figure 1: STLC Diagram 4

1 Software Testing Life Cycle:

Software Testing Life Cycle (STLC) is a systematic and structured approach to testing software applications. It outlines the different phases and activities involved in testing a software product. The STLC process ensures that testing is carried out in a planned and controlled manner, leading to a higher-quality end product. Here's an introduction to the key phases in the Software Testing Life Cycle.

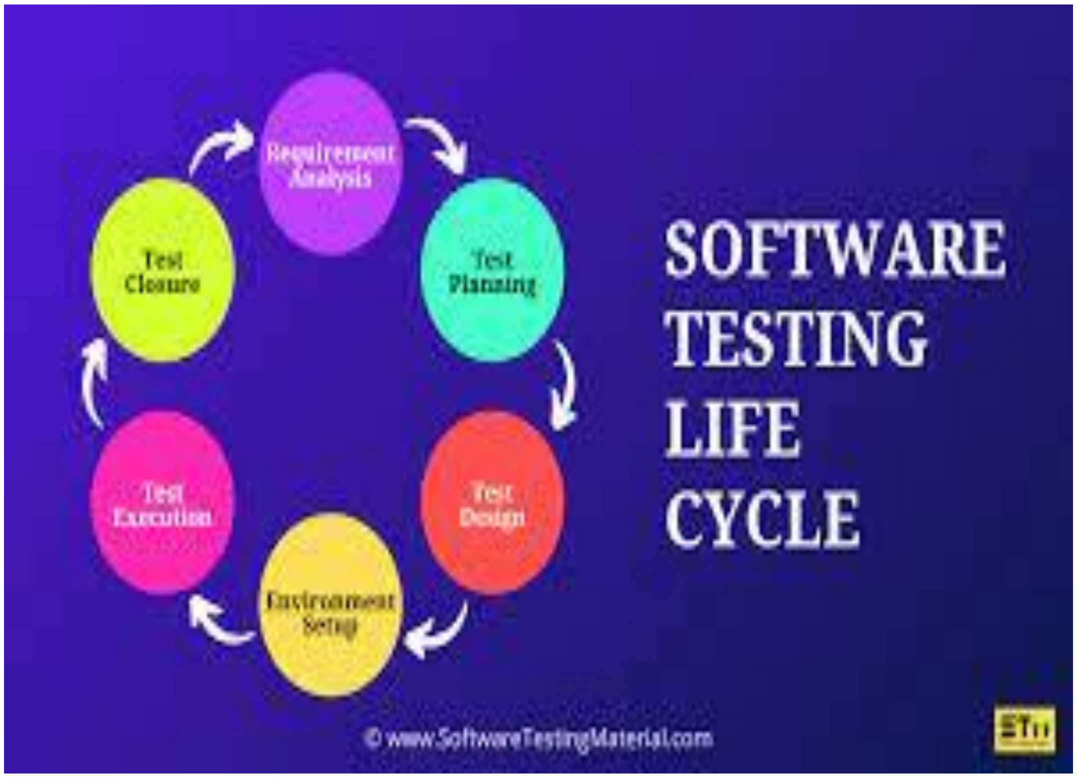

Figure 1: STLC Diagram

1.1 Requirement Analysis Phase:

- In this phase, the testing team reviews and analyzes the requirements documents to gain a thorough understanding of what needs to be tested.
- Testers identify testable features, dependencies, and potential risks.

1.2 Test Planning Phase:

- A test plan is created which outlines the overall testing strategy, objectives, scope, resources, and schedule.
- Testers identify the types of tests to be conducted, including functional, non-functional, and specific testing techniques.

1.3 Test Design Phase:

- In this phase, test cases and test scripts are created based on the requirements and design documents.
- Testers develop both positive and negative test scenarios, specifying inputs, expected outputs, and execution steps.

1.4 Test Environment Setup Phase:

- The testing environment, which includes hardware, software, network configurations, and test data, is established.
- This phase ensures that the testing environment accurately simulates the production environment.

1.5 Test Execution Phase:

- Testers execute the test cases using the prepared test environment.
- They record actual results and compare them with expected results to identify discrepancies (defects).

1.6 Defect Reporting and Tracking Phase:

- When discrepancies are found, they are logged as defects in a defect tracking system.
- Each defect is assigned a severity level and prioritized based on its impact on the system.

1.7 Regression Testing Phase:

- After defects are fixed, regression tests are executed to ensure that the changes did not introduce new issues.
- This phase ensures that existing functionalities remain intact after modifications.

1.8 Release Readiness Phase:

- The testing team assesses whether the software is ready for release.
- They evaluate if the testing objectives have been met and if the quality is acceptable for the intended audience.

1.9 Deployment and Post-Deployment Phase:

- Once the software passes all tests and meets the release criteria, it is deployed in the production environment.
- Testers may also conduct post-deployment testing to verify that the software performs as expected in the live environment.

2 Test Cases in STLC:

Test cases are a critical component of the Software Testing Life Cycle (STLC). They provide detailed instructions for how to verify and validate the functionality of a software application. Test cases are written based on the requirements and test scenarios identified during earlier phases of the STLC. Here's a brief overview of test cases in the STLC:

2.1 Test Scenario Identification:

- Before writing test cases, test scenarios are identified based on the software's functional and non-functional requirements. A test scenario is a high-level description of a specific functionality or aspect of the software that needs to be tested.

2.2 Test Case Design:

- Test case design is the process of creating individual test cases for each identified test scenario. Test cases should be designed to verify specific aspects of the software, including inputs, expected outputs, and conditions.

2.3 Test Case Components:

- A test case typically consists of the following components:
    - **Test Case ID:** A unique identifier for the test case.

- **Test Case Description:** A clear and concise description of what the test case is meant to achieve.
- **Test Steps:** Detailed steps to follow in order to execute the test case.
- **Input Data:** Specific data or conditions required to execute the test case.
- **Expected Result:** The anticipated outcome when the test case is executed successfully.
- **Preconditions:** Any conditions or requirements that must be met before executing the test case.
- **Post-conditions:** The state of the system after the test case has been executed.

2.4 Positive and Negative Test Cases:

- Test cases should cover both positive and negative scenarios. Positive test cases verify that the software behaves as expected under normal conditions, while negative test cases validate its response to unexpected or erroneous inputs.

2.5 Boundary Value Test Cases:

- Boundary value test cases check how the software behaves at the edges or boundaries of input data ranges. These are critical for finding issues related to data limits and constraints.

2.6 Equivalence Partitioning:

- Test cases can be designed based on the concept of equivalence partitioning, where input data is divided into logical groups or partitions to ensure that each group is tested adequately.

2.7 Data-Driven Test Cases:

- Data-driven test cases involve testing the same functionality with different sets of data to ensure that the software performs consistently across various inputs.

2.8 Test Case Execution:

- Once test cases are written, they are executed during the test execution phase of the STLC. Testers follow the test steps, input the data, and compare the actual results with the expected results.

2.9 Defect Reporting:

- If a test case fails, a defect is reported. The defect report should include detailed information about the test case, the steps followed, the actual result, and the expected result.

2.10 Regression Testing:

- Test cases are often used for regression testing, ensuring that previously tested functionality still works after new code changes are introduced.

2.11 Maintenance and Updates:

- Test cases may need to be updated and maintained as the software evolves, requirements change, or defects are fixed.

Well-documented and comprehensive test cases are essential for effective testing in the STLC. They help ensure that the software is thoroughly tested, defects are identified, and the software meets the specified quality standards.

3 Example test case for Face book

Test cases for a complex application like Facebook can be extensive and are typically designed to cover a wide range of functionality, including user registration, profile management, social interactions, security, and more. Below, I'll provide a few example test cases for Facebook to give you an idea of what they might entail. Note that these are simplified examples, and in a real-world scenario, test cases would be more detailed and comprehensive.

3.1 User Registration:

3.1.1 Test Case 1: Verify User Registration with Valid Data

- Input: Valid email, password, name, and birthdate.
- Expected Result: The user is successfully registered, and a confirmation email is sent.

3.1.2 Test Case 2: Verify User Registration with Invalid Email

- Input: Invalid email format.
- Expected Result: The system should display an error message, and registration should not proceed.

3.2 Login and Authentication:

3.2.1 Test Case 3: Verify Successful Login

- Input: Valid email and password.
- Expected Result: The user is logged in, and the homepage is displayed.

3.2.2 Test Case 4: Verify Failed Login with Incorrect Password

- Input: Valid email and an incorrect password.
- Expected Result: The system should display an error message, and login should fail.

3.3 Profile Management:

3.3.1 Test Case 5: Verify Profile Picture Upload

- Input: Upload a profile picture.
- Expected Result: The profile picture is uploaded successfully and displayed on the user's profile.

3.3.2 Test Case 6: Verify User Status Update

- Input: Post a status update.
- Expected Result: The status update is visible to the user's friends and followers.

3.4 Friend Requests:

3.4.1 Test Case 7: Verify Sending a Friend Request

- Input: Send a friend request to another user.
- Expected Result: The other user receives the request and can choose to accept or decline it.

3.4.2 Test Case 8: Verify Accepting a Friend Request

- Input: Accept a friend request.
- Expected Result: The two users are now connected as friends, and their profiles reflect this.

3.5 Privacy and Security:

3.5.1 Test Case 9: Verify Changing Password

- Input: Change the account password.

- Expected Result: The password is updated successfully, and the user can log in with the new password.

3.5.2 Test Case 10: Verify Privacy Settings

- Input: Configure privacy settings to restrict who can see the user's posts.
- Expected Result: The selected privacy settings are applied, and only the specified audience can view the user's posts.

3.6 Notifications:

3.6.1 Test Case 11: Verify Notification for New Friend Request

- Input: Receive a friend request.
- Expected Result: The user receives a notification alerting them to the new friend request.

3.6.2 Test Case 12: Verify Notification for Message

- Input: Receive a new message.
- Expected Result: The user receives a notification about the unread message.

These are just a few examples of test cases for Facebook. In a real testing scenario, there would be many more test cases to cover various functionalities, device compatibility, and different scenarios. Test cases need to be designed meticulously to ensure that Facebook operates smoothly and securely for its users.

4 Conclusion

In conclusion, the software testing life cycle is an integral part of software development that ensures software quality, reduces defects, and helps meet user requirements and expectations. It is a dynamic and iterative process that evolves with changing project demands and technologies, ultimately contributing to the delivery of reliable and high-quality software applications.