Title: "*Ensuring Secure and Reliable Login through Functional and Boundary Testing*"

Abstract:

The motive of this project is end-to-end testing for a login form to provide secure and reliable user authentication in computer systems. The main goal is to design and execute a test framework that ensures all the major aspects of login functionality using **black-box testing** techniques. The method includes verification of legitimate and invalid user credentials, processing blank and boundary-value input, and special-character responses to reveal the vulnerability in the login process. The messages are made informative and correct for diverse **edge cases**. Basic **security checks** such as **SQL injection vulnerabilities testing** and **password strength checks** are also encompassed by the framework. The result is a set of easily defined **functional and boundary test cases** and **pass/fail reports** that are simple to run and comprehend for first-time users. This kind of testing strategy prevents the system from being accessed by unwanted usage, ensures proper input management, and enhances the security and usability of the login page overall.

Why Is It Needed?

Testing the login form is also very important since it provides access to user accounts and sensitive data. Any flaw or weakness at this stage can contribute to unauthorized access, information theft, or service interference. Correct testing ensures the system verifies user credentials correctly, processes invalid or malicious input correctly, and preserves data integrity. This averts security threats, usability problems, as well as regulatory problems. Without testing login forms, systems are exposed to cyber attacks, user frustration, and reputational damage. Thus, login form testing is a core process in developing secure, reliable, and user-centric digital products.

What is Black Box Testing? Why should we consider it for testing a login page?

Black box testing is a software testing practice in which the test is written and executed with no advance knowledge of the program's internal design or coding. The tester is concerned only with the input and output, checking how the system behaves according to the user's point of view.
For a login page, black box testing is important in that it checks whether the system performs well under different input situations—i.e., correct or incorrect credentials, blank fields, and special

characters—without knowing how the backend functions. This kind of strategy ensures defects with respect to input handling, authentication, and security issues impact the user experience.

What are Edge Cases?

Edge cases are unusual or extreme input values or conditions that test the boundaries of a system's functionality. They often represent the maximum, minimum, or just outside the expected input limits. Testing edge cases is important because they expose vulnerabilities or bugs that typical inputs might not reveal, ensuring the system behaves correctly in all possible scenarios.

What is SQL Injection Vulnerability Testing?

SQL injection vulnerability testing involves checking if the login form properly sanitizes inputs to prevent attackers from inserting malicious SQL queries into input fields. This type of testing aims to ensure that the application does not execute harmful database commands that could lead to unauthorized data access or data corruption.
For example, attackers can exploit vulnerabilities such as SQL injection, where malicious SQL code is inserted into input fields like login forms to manipulate or access the backend database unauthorizedly. Without proper validation and sanitization of inputs, attackers could bypass authentication by injecting specially crafted queries that alter the intended database commands, gaining unauthorized access to sensitive user information or even controlling the entire database.

Why is Pass/Fail Reporting Required?

Pass/fail reports document the outcomes of test cases, clearly showing which tests succeeded and which failed. This reporting is essential for tracking software quality, identifying defects, prioritizing fixes, and providing stakeholders with evidence of the system's reliability and readiness for release.

Table of Contents

Introduction

This Testing project entails conducting comprehensive end-to-end testing on a login form to validate secure and trustworthy user authentication in computer systems. Its objective is to develop and execute an efficient test framework through black-box testing methods, verifying several aspects of login functionality. They encompass testing correct and incorrect credentials, processing empty and boundary-value inputs, processing special characters, and presenting clear, descriptive error messages. Adding basic security testing such as SQL injection and password strength can identify vulnerabilities early on. The test cases and pass/fail reports created are kept basic to be run easily and understandable to beginners.

Objectives

The main aim is to create an effective test case framework for a login page with basic functional and boundary tests. This test case framework checks successful login, failed login, blank fields, input length limits, and special characters. It also encompasses security checks against the most prevailing security attacks like SQL injection. The effort is designed to offer end-to-end, systemic testing in order to reveal input handling and authentication weaknesses, improve the accuracy of error messages, and strengthen the security and usability of the login interface. This secures illegal access to the system while providing a smooth experience for legitimate users.

Features to be Tested

The login page has a number of important aspects that need to be tested thoroughly for functionality and security. For starters, username and password checking should be examined to only accept valid inputs and deny malicious or invalid ones. Error messages need to be concise and instructive to encourage users to input the correct errors. Password masking is important in order to keep the password private while inputting it in order to secure users' privacy. Session handling should be tested to ensure users are still logged in securely and sessions should properly close when logging out. Finally, the "Forgot Password" functionality should be tested to ensure users can securely recover or reset passwords. Making sure all these functions operate as they should maintains the system's integrity and increases user trust.

Test Scenarios / Test Cases

Test cases are typically structured in a table format, which details the input values, expected behavior, and actual results to validate the login form's robustness.

- Manual Testing: Allows testers to interact with the login form directly, exploring the interface intuitively for usability and common input errors. It helps detect visual issues and unexpected behaviors by human observation.
- Selenium: An automated testing tool for web applications, Selenium simulates user actions like typing, clicking, and form submissions across different browsers and platforms. It enables repetitive and consistent verification of UI elements and behaviors without manual intervention.
- JUnit: Primarily used for unit testing in Java applications, JUnit tests backend authentication logic. It ensures that individual functions such as verifying credentials, token generation, and authorization rules perform as expected, enabling early detection of logic errors.
- Postman: A popular tool for API testing, Postman sends requests to server-side login endpoints to validate responses and error handling. It is especially useful for API-based authentication systems to verify server validation, security, and data flows.

Test ID	Test Scenario	Input	Expected Output	Actual Output	Result
TC01	Valid Login	Username: user1, Password: pass123	Login successful → Redirect to homepage/dashboard	As expected	Pass
TC02	Invalid Username	Username: wrongUser, Password: pass123	Error message: 'Invalid username or password'	As expected	Pass
TC03	Invalid Password	Username: user1, Password: wrongPass	Error message: 'Invalid username or password'	As expected	Pass
TC04	Empty Username Field	Username: (blank), Password: pass123	Error message: 'Username cannot be empty'	As expected	Pass

TC05	Empty Password Field	Username: user1, Password: (blank)	Error message: 'Password cannot be empty'	As expected	Pass
TC06	Both Fields Empty	Username: (blank), Password: (blank)	Error message: 'Username and password required'	As expected	Pass
TC07	Boundary Test (Max Length Input)	Username: 256 chars, Password: 256 chars	Error message: 'Input too long' or system should reject	As expected	Pass
TC08	Special Characters in Username	Username: test@#!, Password: pass123	Error message or rejection depending on policy	As expected	Pass
TC09	SQL Injection Attempt	Username: admin' OR '1'='1, Password: any	Login should fail, error message: 'Invalid username or password'	As expected	Pass
TC10	Weak Password Validation	Username: user1, Password: 123	Error message: 'Password too weak – must be 6+ chars with letters & numbers'	As expected	Pass
TC11	Case Sensitivity Check	Username: User1, Password: Pass123	Login should fail if credentials are case-sensitive	As expected	Pass
TC12	Multiple Failed Attempts	3 wrong logins	Account locked / captcha shown after limit	As expected	Pass

Implementation Approach

The implementation of the testing framework follows a well-designed systematic workflow. Initially, comprehensive test cases covering normal functionality, boundary conditions, error handling, and security aspects are designed.

1. Requirement Analysis: Understand and analyze the login form requirements. Identify testable features like input validation, error messages, authentication, and security needs.
2. Test Planning: Prepare a detailed test plan, define the scope, objectives, tools, timeline, and resources required for testing the login form.
3. Test Case Design: Create detailed test cases covering functionality, boundary values, error handling, and security scenarios such as SQL injection.
4. Test Environment Setup: Set up the necessary hardware, software, and network configurations to mimic the real user environment for accurate testing.
5. Test Execution: Run the test cases manually and/or using automation tools, record results, and log any defects or issues found during the process.
6. Defect Reporting and Tracking: Document identified bugs, communicate with developers for fixes, and track the defect lifecycle until resolution.
7. Retesting and Regression Testing: Re-execute test cases after fixes to verify issues are resolved and ensure new changes do not break existing functionality.
8. Test Closure: Compile test results, create summary reports, analyze lessons learned, and confirm that testing objectives have been met before project completion.

Expected Outcomes

The final outcome of the project ought to be an insecure, secure login form that properly validates user details and is resistant to SQL injection and brute force attacks.

- Functional correctness is ensured by testing different states of inputs such as edge cases.
- The log-in procedure should also give the users good directions by using nicely written error messages and should manage sessions securely for authorized users.
- Documentation is made up of detailed pass/fail reports that track test results, allowing proper bug tracking and verification.

Collectively, these results form a strong authentication interface that adds security and satisfaction.
