Department of Computer Engineering

Academic Term: First Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	8
Title:	Black Box Testing
Date of Performance:	
Roll No:	9607
Team Members:	Sanika Patankar, Lisa Gonsalves, Eden Evelyn Charles

Rubrics for Evaluation:

Sr. No.	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not On Time)	
2	Theory Understanding (02)	02 (Correct)	NA	01 (Tried)	
3	Content Quality (03)	01 (All used)	02 (Partial)	03 (Rarely allowed)	
4	Post Lab Questions (04)	04 (Done Well)	03 (Partially Correct)	02 (Submitted)	

Signature of the Teacher:

Lisa Gonsalves
9607
TE COMPS B

SE EXP 8: Black Box Testing

TEST CASE 1:

Test Case ID		SP_001	**Test Case Description**		Testing the Shake feature					
Created By		Sanika	**Reviewed By**		Lisa		**Version**		1.0	
QA Tester's Log		Review the app in version 1.0								
Tester's Name		Eden	**Date Tested**		April 22, 2023		**Test Case (Pass/Fail/Not Executed)**		Pass	
S #	**Prerequisites:**				**S #**	**Test Data**				
1	User should have the FEMEI app downloaded				1	Email id = crce.9563.ce@gmail.com				
2	User shouldhave signed in				2	Password = Sanika12				
3	User should have filled in the requeried data including emergency contact				3	Emergency Contact Number: 87******98				
4					4					
Test Scenario	Verifying the smooth working of the Shake feature									
Step #	**Step Details**		**Expected Results**		**Actual Results**			**Pass / Fail / Not executed / Suspended**		
1	Navigate to FEMEI Application		Site should open		As Expected			Pass		
2	Enter Userid & Password		Credential can be entered		As Expected			Pass		
3	Click Submit		Cutomer is logged in		As Expected			Pass		
4	Click on the Emergency button		Emergency Page should open		As Expected			Pass		
5	Shake the phone to connect a call to the emergency contact		Call Should go to the emergency contact		As Expected			Pass		

TEST CASE 2:

Test Case ID		LG_001	**Test Case Description**		Testing the Prediction of the next cycle feature					
Created By		Lisa	**Reviewed By**		Sanika		**Version**		1.0	
QA Tester's Log		Review the app in version 1.0								
Tester's Name		Eden	**Date Tested**		April 21, 2023		**Test Case (Pass/Fail/Not Executed)**		Pass	
S #	**Prerequisites:**				**S #**	**Test Data**				
1	User should have the FEMEI app downloaded				1	Email id = crce.9563.ce@gmail.com				
2	User shouldhave signed in				2	Password = Sanika12				
3	User should have filled in the requeried data including the start of the current cycle				3					
4					4					
Test Scenario	Verifying the prediction feature									
Step #	**Step Details**		**Expected Results**		**Actual Results**			**Pass / Fail / Not executed / Suspended**		
1	Navigate to FEMEI Application		Site should open		As Expected			Pass		
2	Enter Userid & Password		Credential can be entered		As Expected			Pass		
3	Click Submit		Cutomer is logged in		As Expected			Pass		
4	Click on the Prediction button		Prediction Page should open		As Expected			Pass		
5	Answer the questions asked		Should display a page with 12 questions to be answered		As Expected			Pass		
6	View the predictions		Display the results of the next cycle		As Expected			Pass		

POSTLABS:

a. Create a set of black box test cases based on a given set of functional requirements, ensuring adequate coverage of different scenarios and boundary conditions.

Creating a black-box test cases for a simple login functionality based on the following functional Requirement: Functional Requirement Example: A web application should allow users to log in using a username and password. Upon successful login, the user should be redirected to their dashboard. If login is unsuccessful, the user should see an error message. Here's how we can create black-box test cases for this requirement:

Test Case: Valid Login
Input: Username = "user123," Password = "pass123"
Expected Output: Redirect to the dashboard.

Test Case: Invalid Username
Input: Username = "invaliduser," Password = "pass123"
Expected Output: Error message: "Invalid username or password."

Test Case: Invalid Password
Input: Username = "user123," Password = "wrongpass"
Expected Output: Error message: "Invalid username or password."

Test Case: Empty Username and Password
Input: Username = "" (empty), Password = "" (empty)
Expected Output: Error message: "Please enter a username and password."

Test Case: SQL Injection Attempt
Input: Username = "user123," Password = "pass123' OR '1'='1"
Expected Output: Error message: "Invalid username or password."

Test Case: Password with Special Characters
Input: Username = "user123," Password = "p@ssw0rd"
Expected Output: Redirect to the dashboard (testing password with special characters).

Test Case: Account Locked After Three Failed Attempts
Input: Try incorrect login three times with any username and password.
Expected Output: After the third incorrect attempt, display an error message and lock the account for a specific duration.

Test Case: Boundary Condition (Long Username and Password)
Input: Username with 100 characters, Password with 100 characters
Expected Output: Either a successful login or an error message (testing boundary condition with maximum input length).

These test cases cover a range of scenarios, including valid and invalid logins, handling of special characters and security concerns, account locking, and boundary conditions. Each test case aims to verify whether the login functionality works as expected, according to the given requirements, without considering the internal code of the application

b. Evaluate the effectiveness of black box testing in uncovering defects and validating the software's functionality, comparing it with other testing techniques.
Black box testing is a valuable software testing technique for identifying defects and validating software functionality. Its effectiveness varies depending on the context and specific testing requirements. Here's an evaluation of its effectiveness and a comparison with other testing techniques:

Effectiveness of Black Box Testing:

1. **Uncovering Defects**: Black box testing is effective at finding defects related to incorrect functionality, boundary conditions, and input/output issues. It's particularly useful for identifying integration problems.
2. **Validation of Functionality**: This technique is well-suited for verifying that software aligns with external requirements and user expectations without requiring knowledge of the internal code.
3. **Independence from Implementation Details**: Testers don't need to know the internal code, making it accessible to non-developers and free from potential biases.
4. **User-Centric Approach**: Black box testing focuses on the end-user's perspective, ensuring it matches user experiences.
5. **Systematic Test Case Design**: Various techniques enable systematic test case design for comprehensive test coverage.

Comparison with Other Testing Techniques:

1. **White Box Testing**: White box testing examines the internal code and is excellent for code-level defect identification but less effective for user perspective evaluation. Combining both techniques can provide comprehensive coverage.
2. **Gray Box Testing**: Gray box testing combines elements of both black and white box testing, offering targeted test case design with partial knowledge of the internal code.
3. **Regression Testing**: Black box testing is effective for regression testing, ensuring that the software's external behaviour remains intact when new changes are introduced.

In summary, black box testing is highly effective for defect identification and functionality validation, especially from the user's viewpoint and when internal code knowledge isn't required. It's a valuable part of a comprehensive testing strategy and can be enhanced by combining it with other testing techniques, depending on the specific project needs.

c. Assess the challenges and limitations of black box testing in ensuring complete test coverage and discuss strategies to overcome them.
Black box testing is valuable but has challenges and limitations when it comes to achieving complete test coverage. Test coverage refers to how well a testing process exercises a software system. Here are the challenges and strategies to overcome them:

Challenges:

1. **Limited Knowledge of Internal Code**: Testers lack access to the internal code, making it difficult to cover all code paths and edge cases.
2. **Inadequate Test Scenarios**: Identifying all possible test scenarios and boundary conditions is challenging without code knowledge.
3. **Overlooking Integration Issues**: Black box testing may miss integration problems that require insight into internal workings.
4. **Inefficiency in Exploratory Testing**: Black box testing is less efficient for exploratory testing that requires extensive exploration of the application.

Strategies to Overcome Limitations:

1. **Collaboration with Developers**: Foster collaboration between testers and developers to gain insights into the software's architecture and potential concerns.
2. **Requirements Analysis**: Thoroughly analyze requirements and design test cases based on them.
3. **Boundary Value Analysis and Equivalence Partitioning**: Use these techniques to systematically design test cases that cover various inputs.
4. **Risk-Based Testing**: Prioritise testing based on risk assessment, focusing on critical areas.
5. **Use of Testing Tools**: Leverage automated testing tools to simulate different inputs and perform a large number of test cases efficiently.
6. **Pair Testing**: Testers work closely with developers to describe test scenarios and gain insights.
7. **Comprehensive Test Plan**: Create a well-documented test plan with clear objectives, scenarios, and exit criteria.
8. **Exploratory Testing as Supplement**: Use exploratory testing alongside black box testing to uncover unexpected issues.
9. **Continuous Feedback and Learning**: Document lessons learned and apply them to future testing efforts.
10. **External Audit and Third-party Review**: Consider external reviews to gain an independent perspective.

In conclusion, while black box testing has limitations in achieving complete test coverage, a combination of strategies can help mitigate these challenges and ensure effective testing. The key is to strike a balance between structured testing and exploratory approaches, making use of available resources and knowledge.