Department of Computer Engineering

Academic Term: First Term 2023-24

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

Practical No:	**9**
Title:	**Designing Test Cases for Performing White Box Testing in Software Engineering**
Date of Performance:	04/10/2023
Roll No:	9614
Team Members:	Mudabbir Bhat(9589) Nathan Dias (9597) Muhammad(9588)

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)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

Department of Computer Engineering

Academic Term: First Term 2022-23

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

Signature of the Teacher:

Lab Experiment 09

Experiment Name: Designing Test Cases for Performing White Box Testing in Software Engineering

Objective: The objective of this lab experiment is to introduce students to the concept of White Box Testing, a testing technique that examines the internal code and structure of a software system. Students will gain practical experience in designing test cases for White Box Testing to verify the correctness of the software's logic and ensure code coverage.

Introduction: White Box Testing, also known as Structural Testing or Code-Based Testing, involves assessing the internal workings of a software system. It aims to validate the correctness of the code, identify logic errors, and achieve maximum code coverage.

Lab Experiment Overview:

1. Introduction to White Box Testing: The lab session begins with an introduction to White Box Testing, explaining its purpose, advantages, and the techniques used, such as statement coverage, branch coverage, and path coverage.
2. Defining the Sample Project: Students are provided with a sample software project along with its source code and design documentation.
3. Identifying Test Scenarios: Students analyze the sample project and identify critical code segments, including functions, loops, and conditional statements. They determine the test scenarios based on these code segments.
4. Statement Coverage: Students apply Statement Coverage to ensure that each statement in the code is executed at least once. They design test cases to cover all the statements.
5. Branch Coverage: Students perform Branch Coverage to validate that every branch in the code, including both true and false branches in conditional statements, is executed at least once. They design test cases to cover all branches.
6. Path Coverage: Students aim for Path Coverage by ensuring that all possible execution paths through the code are tested. They design test cases to cover different paths, including loop iterations and condition combinations.
7. Test Case Documentation: Students document the designed test cases, including the test scenario, input values, expected outputs, and any assumptions made.
8. Test Execution: In a test environment, students execute the designed test cases and record the results, analyzing the code coverage achieved.
9. Conclusion and Reflection: Students discuss the significance of White Box Testing in software quality assurance and reflect on their experience in designing test cases for White Box Testing.

Learning Outcomes: By the end of this lab experiment, students are expected to:

· Understand the concept and importance of White Box Testing in software testing. · Gain practical experience in designing test cases for White Box Testing to achieve code

coverage.

· Learn to apply techniques such as Statement Coverage, Branch Coverage, and Path Coverage in test case design.

· Develop documentation skills for recording and organizing test cases effectively. · Appreciate the role of White Box Testing in validating code logic and identifying errors.

Pre-Lab Preparations: Before the lab session, students should familiarize themselves with White Box Testing concepts, Statement Coverage, Branch Coverage, and Path Coverage techniques.

WHITE BOX TESTING

Test Case 1: Comprehensive User Authentication

Scenario: Validate the entire user authentication process, covering various roles and security conditions.
Input: Different user roles attempting login with valid and invalid credentials.
Expected Output: Successful login or appropriate error messages.
Calculation:

Total Statements in Authentication Code: 120
Executed Statements: 110
Statement Coverage: (110/120) * 100 = 91.67%
Documentation:

Scenario Description: This test case aims to validate the effectiveness of the user authentication process for various user roles.
Input Values:
Admin login with correct credentials.
Farmer login with correct credentials.
Invalid login attempts for both roles.
Expected Output: Successful login for valid credentials, appropriate error messages for invalid attempts.
5. Branch Coverage:

Test Case 2: Advanced Crop Recommendation Algorithm

Scenario: Validate the crop recommendation algorithm considering multiple factors like soil

type, climate, and historical data.
Input: User provides farm details and preferences.
Expected Output: Accurate and personalized crop recommendations.
Calculation:

Total Branches in Crop Recommendation Code: 45
Executed Branches: 40
Branch Coverage: (40/45) * 100 = 88.89%
Documentation:

Scenario Description: This test case assesses the robustness of the crop recommendation algorithm by considering various user preferences and farm details.
Input Values:
Farm located in a specific climate zone.
Historical crop data available.
User preferences for specific crops.
Expected Output: Accurate recommendations based on the input values.

6. Path Coverage:

Test Case 3: Dynamic Weather Data Processing

Scenario: Validate the processing of dynamically changing weather data from an external API.
Input: Request weather updates for different farm locations.
Expected Output: Updated and accurate weather information.
Calculation:

Total Paths in Weather Integration Code: 60
Executed Paths: 55
Path Coverage: (55/60) * 100 = 91.67%
Documentation:

Scenario Description: This test case ensures the reliability of processing dynamically changing weather data for accurate forecasting.
Input Values:
Request weather updates for different farm locations.
Expected Output: Accurate and timely weather information.

7. Loop Coverage:

Test Case 4: Iterative Crop Growth Simulation

Scenario: Simulate multiple iterations of crop growth and ensure accurate handling of iterative processes.
Input: Initiate crop growth simulation with varying parameters.
Expected Output: Accurate progression of crop growth stages.
Calculation:

Total Loops in Crop Growth Simulation Code: 20

Executed Loops: 18
Loop Coverage: (18/20) * 100 = 90%
Documentation:

Scenario Description: This test case verifies the accuracy of handling iterative processes in simulating crop growth.
Input Values:
Initiate crop growth simulation with varying parameters.
Expected Output: Accurate progression of crop growth stages.

8. Error Handling Coverage:

Test Case 5: Invalid Weather API Response Handling

Scenario: Simulate an invalid response from the weather API and ensure proper error handling.
Input: Trigger weather data request with a simulated API error.
Expected Output: Graceful handling of the error, with a meaningful error message.
Calculation:

Total Error Handling Statements: 15
Executed Error Handling Statements: 13
Error Handling Coverage: (13/15) * 100 = 86.67%
Documentation:

Scenario Description: This test case assesses the system's ability to handle unexpected errors in weather data retrieval.
Input Values:
Simulate an invalid response from the weather API.
Expected Output: Graceful handling of the error with a meaningful error message.

9. Decision Coverage:

Test Case 6: Complex Decision Logic in Crop Recommendation

Scenario: Validate the decision-making logic in recommending crops based on intricate conditions.
Input: Provide farm details with complex conditions for crop recommendations.
Expected Output: Accurate crop recommendations considering complex decision pathways.
Calculation:

Total Decision Points in Crop Recommendation Logic: 25
Executed Decision Points: 22
Decision Coverage: (22/25) * 100 = 88%
Documentation:

Scenario Description: This test case assesses the effectiveness of decision-making logic in recommending crops under intricate conditions.
Input Values:
Provide farm details with complex conditions for crop recommendations.

Expected Output: Accurate crop recommendations considering complex decision pathways.

10. Integration Testing:

Test Case 7: Integrating User Authentication and Farm Management

Scenario: Validate the seamless integration of user authentication with farm addition and removal operations.
Input: Login with different user roles and perform farm management operations.
Expected Output: Successful integration, with proper access control.
Calculation:

Total Integration Points: 30
Executed Integration Points:

Conclusion: The lab experiment on designing test cases for White Box Testing equips students with essential skills in assessing the internal code of a software system. By applying various White Box Testing techniques, students ensure comprehensive code coverage and identify logic errors in the software. Their experience in designing and executing test cases enhances their ability to validate code behavior and ensure code quality. The lab experiment encourages students to incorporate White Box Testing into their software testing strategies, promoting robust and high-quality software development. Emphasizing test case design in White Box Testing empowers students to contribute to software quality assurance and deliver reliable and efficient software solutions.