

**Department of Computer Engineering**

**Academic Term: First Term 2023-24 Class: T.E**

**/Computer Sem – V / Software Engineering**

|                             |  |
|-----------------------------|--|
| <b>Practical No:</b>        | <b>9</b>   |
| <b>Title:</b>               | <b>Designing Test Cases for Performing White Box Testing</b> |
| <b>Date of Performance:</b> | <b>03/10/2023</b>  |
| <b>Roll No:</b>             | 9615   |
| <b>Team Members:</b>        | Omkar,Soham,Emmanuel   |

**Rubrics for Evaluation:**

| <b>Sr. No</b> | <b>Performance Indicator</b>         | <b>Excellent</b> | <b>Good</b>           | <b>Below Average</b> | <b>Total Score</b> |
|---------------|--------------------------------------|------------------|-----------------------|----------------------|--------------------|
| 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**

## 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.

Dr. B. S. Daga Fr. CRCE, Mumbai

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.

Materials and Resources:

Project brief and details for the sample software project

Whiteboard or projector for explaining White Box Testing techniques

Test case templates for documentation

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. The 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.

App.js

```
A render() {  
  return (  
    <>  
B    <Navbar />  
C    <Suspense fallback={<div>Loading...</div>}>  
D      <Routes>  
E        <Route path="/" element={<News category="top" />} />  
F        <Route path="/business" element={<News category="business" />} />  
G        <Route  
          path="/entertainment"  
          element={<News category="entertainment" />}  
        />  
H        <Route path="/science" element={<News category="science" />} />  
I        <Route path="/sports" element={<News category="sports" />} />  
J      <Route path="/health" element={<News category="health" />} />  
K      <Route  
        path="/technology"  
        element={<News category="technology" />}  
      />  
    </Routes>  
  </Suspense>  
  <Footer />  
  </>  
  );  
L } //end of render
```

@B c if suspense fallback  
else D

@D E if path="/"  
F if path="/business"  
G if path="/entertainment"  
H if path="/science"  
I if path="/sports"  
J if path="/health"  
K if path="/technology"

