SNS COLLEGE OF TECHNOLOGY

Kurumbapalayam (Po), Coimbatore – 641 035

An Autonomous Institution

Accredited by NAAC – UGC with 'A++' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME: 23ITO201- Software Testing (OPEN ELECTIVE)

III YEAR / VI SEMESTER

1

Unit 2 - TEST CASE DESIGN STRATEGIES

Topic : TEST CASE DESIGN STRATEGIES – BLOCK BOX TESTING

SOFTWARE TESTING SERIES

Black Box Testing

Complete Guide

Learn the essentials of validating software functionality without seeing the code. A deep dive into techniques, types, and best practices.

- Definition & Goals
- Functional Types
- Testing Techniques
- Real-world Examples

DEFINITION

What Is Black Box Testing?

A software testing method where the functionality of an application is tested **without knowing** its internal code structure, implementation details, or internal paths.

Hidden Internal Logic

The internal workings are a "black box" to the tester. You don't see how it works, only that it works.

End-User Perspective

Focuses on inputs and expected outputs, simulating how a real user interacts with the software.

The "Black Box" Concept

"We check if the output matches the requirements for a given input, ignoring the internal mechanism."

CORE CONCEPT

Input → Black Box → Output

The fundamental principle: Validating behavior by observing outputs for controlled inputs, while the system's internal logic remains hidden.

4

REAL WORLD EXAMPLE: LOGIN

OBJECTIVES

Main Goals of Black Box Testing

Why do we do it? To ensure software works exactly as intended from the outside in.

Functional Compliance

Verify that the software meets all specified functional requirements and business rules.

Output Accuracy

Ensure correct results are produced for both valid and invalid input sets.

Robustness

Confirm the system handles errors gracefully and doesn't crash on bad data.

User Perspective

Validate that the behavior matches end-user expectations and usability standards.

SCOPE OF TESTING

What Does It Test?

Focusing on the external behavior across four key dimensions

Functional Behavior

Core Business Logic

- Verifying correct outputs for specific inputs (Happy Path).
- Validation of business rules and calculations.
- Data processing integrity (CRUD operations).
- Integration between different modules or APIs.

UI & Usability

Look & Feel

- Navigation flow and layout consistency.
- Form field validation messages and clarity.
- Accessibility compliance (contrast, tab order).
- Content accuracy (labels, tooltips, instructions).

6

Error Handling

Robustness

- System behavior under invalid or missing inputs.
- Graceful recovery from failure states/crashes.

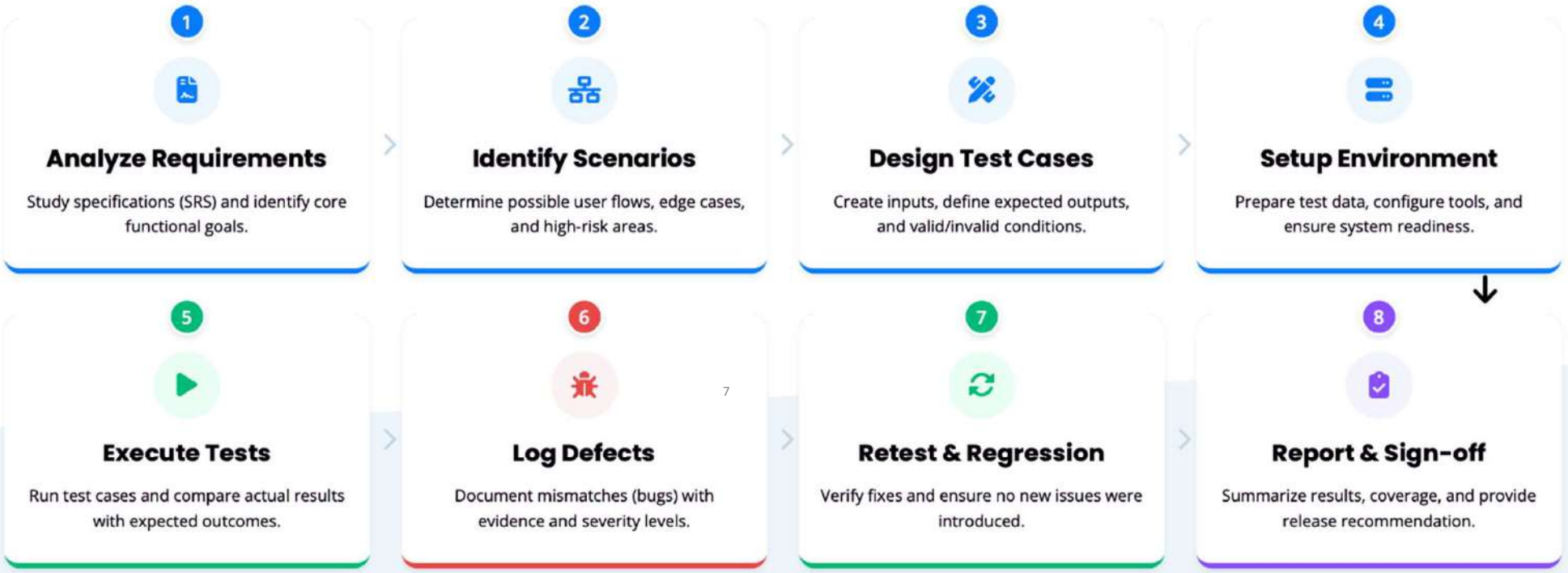
Perf & Compatibility

Environment

- Responsiveness across different screen sizes.
- Browser and OS compatibility rendering.

Black Box Testing Process

A systematic approach to validating software functionality



Types of Black Box Testing

The methodology is broad, but most tests fall into three primary categories depending on the testing objective.

Functional Testing

Validates that specific features and functions of the software operate in conformance with the requirement specifications. Focuses on "Does it work?"

SANITY | INTEGRATION | SYSTEM | ACCEPTANCE

Regression Testing

Ensures that recent code changes, updates, or bug fixes haven't adversely affected existing features. Focuses on "Did we break anything?"

8

RETESTING | FULL SUITE | SELECTIVE | AUTOMATED

Non-Functional

Tests aspects of the application not related to specific behaviors, such as reliability, efficiency, and usability. Focuses on "How well does it work?"

PERFORMANCE | USABILITY | SECURITY | COMPAT

Functional Testing

Verifies that each feature of the application works in conformance with requirement specifications.

PRIMARY GOAL

"Does it do what it's supposed to do?"

Manual & Automated

COMMON SCENARIOS

Login & Auth
Verify credential checks, access rights, and session timeouts.

Form Validation
Check required fields, email formats, and character limits.

Search & Filter
Ensure results match queries and filters narrow data correctly.

Checkout Flow
Validate cart updates, calculations, and payment processing.

KEY FOCUS AREAS

Inputs & Outputs
Correct result for given data

Business Rules
Logic specific to domain

UI Interaction
Navigation & Controls

API Responses
Status codes & payload

Regression Testing

The practice of re-running tests to ensure that code changes haven't adversely affected existing features.

PRIMARY GOAL

"Did we break anything that was working?"

Automated + Targeted Manual

WHEN TO PERFORM

After Bug Fixes
Verify the fix works and ensure no new defects were introduced.

New Features
Check that adding new modules doesn't conflict with old ones.

Code Merges
Validate system integrity when branches are merged to main.

Before Release
Final safety net cycle before deployment to production.

STRATEGY & TIPS

10

Automate
Use scripts for repetitive, stable tests

Prioritize Risk
Test critical business flows first

Impact Analysis
Focus on areas affected by code

Sanity Check
Smoke test before full suite

Non-Functional Testing

Verifies the quality attributes of the system—focusing on "how well" it operates rather than just "what" it does.

CORE QUESTION

"Is it fast, usable, and reliable?"

Metrics & Attributes

TESTING AREAS

Performance
Check response time, throughput, and system scalability under load.

Usability
Evaluate user flow clarity, efficiency, and accessibility standards.

Compatibility
Verify behavior across browsers, OS versions, devices, and locales.

Black Box Context
Tests rely on observing external behavior & outputs, not code profiling.

KEY METRICS

Response Time	WCAG Score	Environments	Throughput
Latency & Speed	Accessibility Level	Cross-browser/OS	Requests per sec

BENEFITS

Key Advantages

Why Black Box Testing is a critical component of a robust QA strategy.

No Coding Required

Testers don't need deep programming knowledge or access to source code. This allows non-technical QA specialists and domain experts to validate software effectively.

Real User Perspective

Tests mirror actual end-user behavior. This "outsider" view helps identify usability issues, confusing workflows, and gaps that developers might miss.

Reusable Test Cases

12

Test scenarios are based on functional specs, not code. This makes them highly reusable for regression testing even if the underlying implementation changes.

Effective Across Layers

Applicable at multiple levels including Unit, Integration, System, and Acceptance testing. It validates the complete system stack from UI to API responses.

CHALLENGES

Limitations & Challenges

While effective for functionality, Black Box testing has blind spots that teams must manage.

Hidden Code Issues

Since the internal structure is invisible, logic errors like unreachable code, infinite loops, or inefficient algorithms often go undetected unless they cause an explicit output failure.

Unknown Test Coverage

It is difficult to design exhaustive test cases without seeing the code. Testers may miss testing specific code branches or complex conditional paths, leading to gaps in coverage.

Missed Deep Defects

Subtle bugs that lie deep within the implementation logic—but don't immediately crash the UI—might be overlooked compared to White Box testing which analyzes the source.

Potential Redundancy

Without coordination, Black Box tests may unknowingly duplicate scenarios already covered by developers in Unit Tests, leading to inefficient use of QA time and resources.

Black Box vs. White Box Testing

Two complementary approaches with distinct perspectives.

Black Box		White Box
Hidden / Unknown	CODE VISIBILITY (VS)	Visible / Known
Functional Behavior External	PRIMARY FOCUS	Internal Logic & Structure Internal
QA Testers / Users	PERFORMED BY	Developers / SDETs
Requirements / Specs	BASIS	Source Code / Design
End-user perspective	PERSPECTIVE	Developer perspective

14

Best Practice: Hybrid Approach

Combine both methods (often called Gray Box Testing) to ensure high-quality software that is both structurally sound and functionally correct.

Activity

Activity Part 4: Error Guessing

✍ **Task**

List **possible errors** based on experience.

Examples:

•Negative marks entered

•Age entered as text

•Double payment

•Page refreshed during payment

👉 **Student Output:**

•At least **5 error-based test cases**

Core Black Box Techniques

Essential methods for designing effective test cases without code visibility.

01

Equivalence Partitioning

Divides input data into valid and invalid partitions. If one value in a partition works, others in the same group are assumed to work too.

02

Boundary Value Analysis

Focuses on values at the extreme ends (boundaries) of input ranges. Tests min, min-1, max, max+1, as errors often lurk at edges.

03

Decision Table Testing

Uses a table to map different combinations of input conditions to expected actions. Ideal for complex business rules with logical dependencies.

04

State Transition Testing

Validates how the system transitions from one state to another triggered by events. Critical for systems with finite states (e.g., ATM, Login).

05

Error Guessing

An experience-based technique where testers anticipate common mistakes or error-prone situations based on intuition and past knowledge.

06

Use Case Testing

Designs test cases based on actual user scenarios (use cases). Ensures the system satisfies functional requirements from start to finish.

MCQ

TEST CASE DESIGN STRATEGIES – MCQs

1. Test case design strategies are used to:

A. Increase code size

B. Reduce testing time without losing coverage

C. Remove all bugs

D. Avoid documentation

Answer: B

2. Which test case design technique divides input data into valid and invalid groups?

A. Boundary Value Analysis

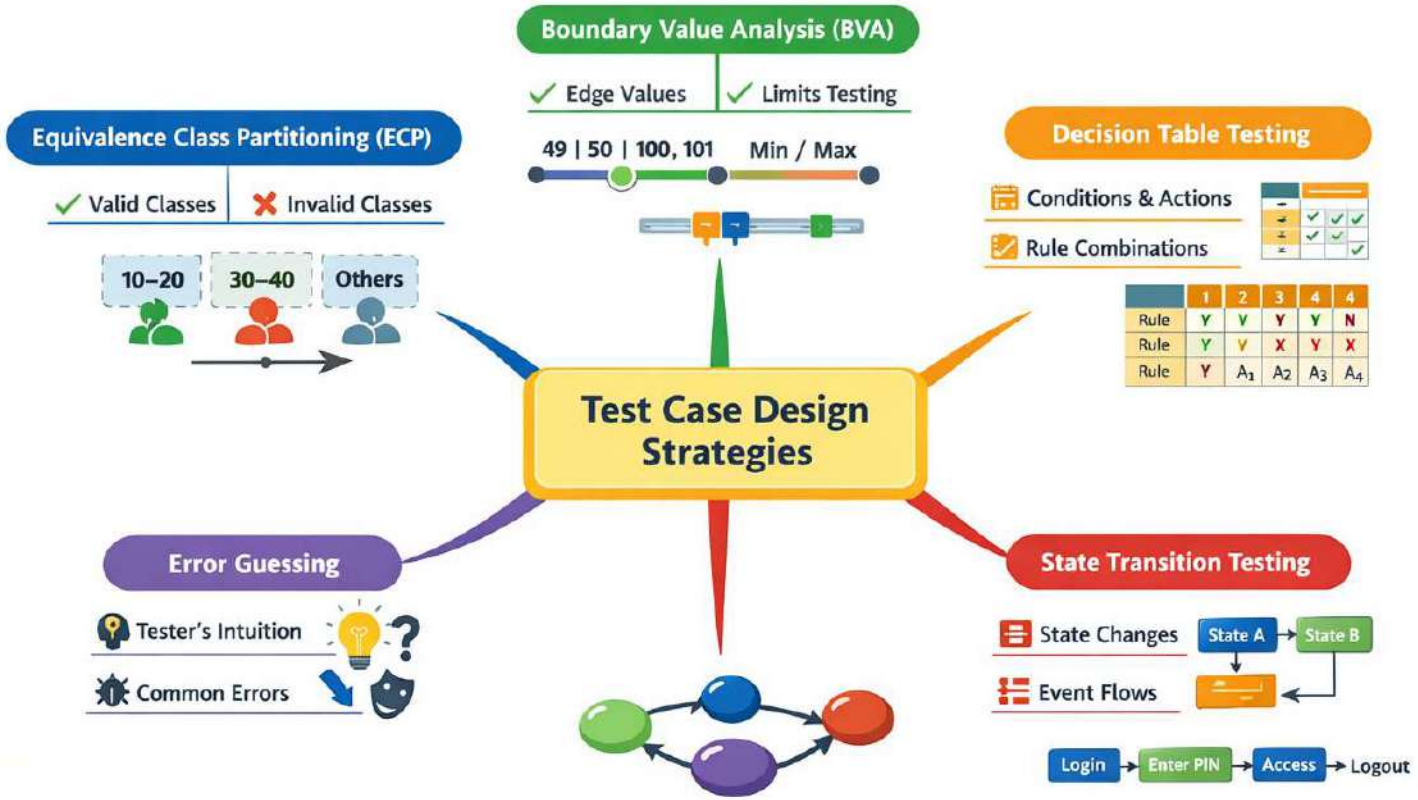
B. Decision Table Testing

C. Equivalence Class Partitioning

D. State Transition Testing

Answer: C

MINDMAP


Boundary Value Analysis (BVA)
Edge Values
Limits Testing
49 | 50 | 100, 101
Min / Max
Equivalence Class Partitioning (ECP)
Valid Classes
Invalid Classes
10–20
30–40
Others
Decision Table Testing
Conditions & Actions
Rule Combinations
Test Case Design Strategies
Error Guessing
Tester's Intuition
Common Errors
State Transition Testing
State Changes
State A
State B
Event Flows
Login
Enter PIN
Access
Logout

TEXT BOOKS	
1.	Srinivasan Desikan and Gopalaswamy Ramesh, ―Software Testing – Principles and Practices, Pearson Education, 2006.
REFERENCES	
1.	Ilene Burnstein, ―Practical Software Testing, Springer International Edition, 2003
2.	Edward Kit, Software Testing in the Real World – Improving the Process‖, Pearson Education, 1995
3.	Boris Beizer, Software Testing Techniques – 2nd Edition, Van Nostrand Reinhold, New York, 1990.
4.	Aditya P. Mathur, ―Foundations of Software Testing _ Fundamental Algorithms and Techniques, Dorling Kindersley (India) Pvt. Ltd., Pearson Education, 2008.

THANK YOU!