**EXPERIMENT NUMBER 2**

**Aim:** Analyse black box testing and white box testing

**Objective:** To compare and understand the implementation of both black box and white box testing techniques.

**Testing:** "The process of evaluating a system, product, or service to determine whether it meets specified requirements, is free from defects, and is fit for its intended purpose."

**Example\_1:** A user testing a website's login functionality without knowing how the authentication process works internally.

**Example\_2:** A developer testing a specific function or module of code to ensure it works as intended, using knowledge of the internal implementation.

**Introduction:**

Black box testing and white box testing are two fundamental testing techniques used in software testing. Black box testing focuses on testing the functionality of the software without knowledge of its internal structure, while white box testing examines the internal code and logic of the software.

**Black Box Testing:**

**Black Box Testing** is a software testing methodology that focuses on the **functional requirements** of a system without delving into its internal workings or implementation details. In other words, testers perform Black Box Testing without any prior knowledge of the system’s source code, algorithms, or intricate internal pathways. Instead, they zero in on a simple principle: inputs and outputs. It’s like evaluating a vending machine—you don’t need to know how it dispenses snacks; you just care that it does so when you insert coins and make a selection[1](https://testsigma.com/guides/black-box-testing/)[2](https://pg-p.ctme.caltech.edu/blog/coding/guide-to-black-box-testing).

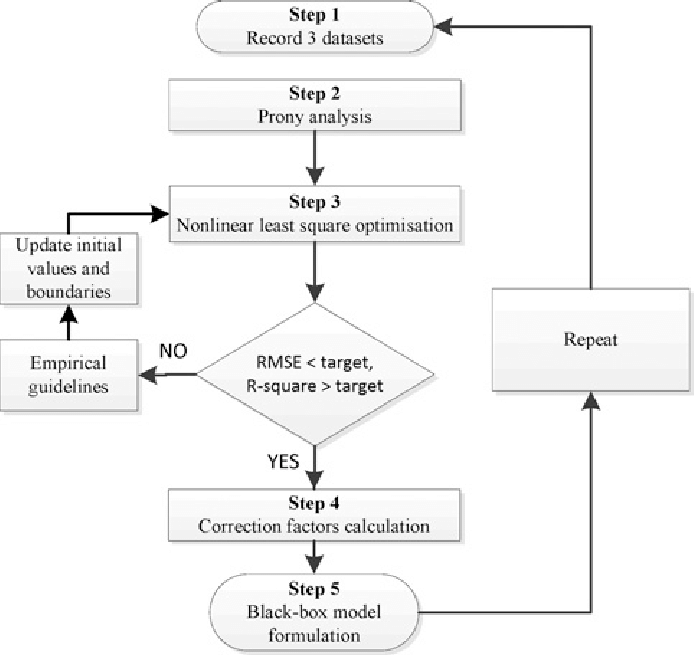
Here are the key features of Black Box Testing:

1. **No Knowledge of Internal Code**: Testers conducting Black Box Testing don’t have access to the software’s internal code. They focus solely on understanding the system’s behaviour based on the inputs provided and the resulting outputs.
2. **Requirements-Centric**: Black Box Testing ensures that a system meets the requirements specified by the customer or user. It’s all about validating whether the software behaves as expected without peeking under the hood.
3. **Independence from Implementation Details**: Unlike its counterpart, White Box Testing (which examines internal code structures), Black Box Testing doesn’t care about how the system achieves its functionality. It’s more concerned with whether it delivers the desired outcomes.

Now, let’s explore a few aspects related to Black Box Testing:

* **Types of Black Box Testing**:
  + **Functional Testing**: This checks whether the software functions correctly according to its specifications.
  + **Non-Functional Testing**: This assesses aspects like performance, usability, and security.
  + **Regression Testing**: Ensures that new changes don’t break existing functionality.
* **What Can Be Identified by Black Box Testing?**:
  + System responses to expected and unexpected user actions
  + Response time
  + Usability issues
  + Reliability issues
* **Black Box Testing Techniques**:
  + **Equivalence Class Testing**: Grouping input values into classes and testing representative values from each class.
  + **Boundary Value Testing**: Focusing on boundary values (e.g., minimum and maximum) to uncover issues.
  + **Decision Table Testing**: Creating decision tables to cover different scenarios.
* **Manual vs. Automated Black Box Testing**:
  + **Manual**: Testers manually execute test cases.
  + **Automated**: Tools automate test case execution, making it faster and more efficient.
* **Tools for Automating Black Box Testing**:
  + **Test sigma**: A cloud-based tool for automated testing.
  + **Catalon**: A popular open-source automation tool.
  + **Test Complete**: A comprehensive testing solution.

**Flowchart :-**



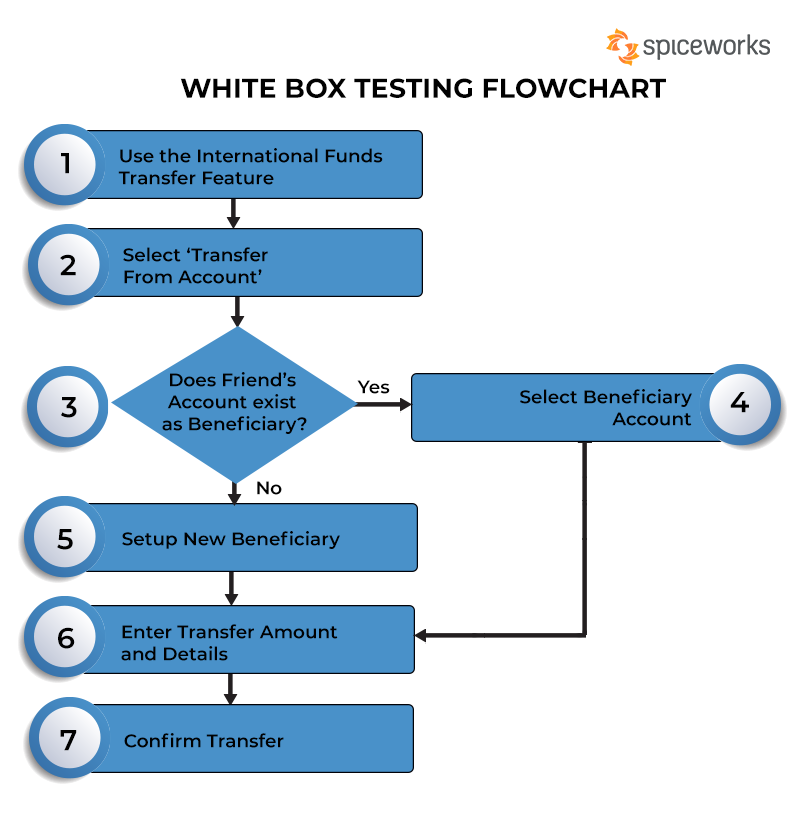
**White Box Testing:**

White Box Testing, also known as structural testing, code-based testing, or glass box testing, is a software testing technique that focuses on the software’s internal logic, structure, and coding. It’s like having X-ray vision for your application—testers get complete access to the source code, design documents, and all the intricate details. They become software detectives, investigating every nook and cranny of the system.

Here’s what makes White Box Testing special:

1. **Complete Visibility**: Testers have the equivalent of a backstage pass—they can see how the software dances behind the scenes. The “house” here is the application, the “blueprint” is the source code, and the “electrical circuits” are the internal workings of the software.
2. **Purpose and Benefits**:
   * **Thoroughness**: White Box Testing ensures complete code coverage. Every part of the software’s internal structure gets scrutinized.
   * **Automation**: Test cases can be easily automated, saving time and resources.
   * **Optimization**: By identifying hidden errors and redundancies, it contributes to code optimization.
   * **Introspection**: It provides an in-depth understanding of the software, which is invaluable for future development and maintenance.
3. **What to Verify in White Box Testing ?**
   * Correctness of code execution paths
   * Boundary conditions
   * Error handling mechanisms
   * Data flow and control flow
4. **Types of White Box Testing**:
   * **Statement Coverage**: Ensures that each line of code is executed at least once.
   * **Branch Coverage**: Checks if all possible branches (if-else, loops) are traversed.
   * **Path Coverage**: Examines all possible execution paths through the code.
5. **Techniques for White Box Testing**:
   * **Static Analysis**: Reviewing the code without executing it.
   * **Dynamic Analysis**: Executing the code and observing its behaviour.
   * **Code Reviews**: Collaborative examination by developers.

**Flowchart :-**



**Comparison of Black Box Testing and White Box Testing:**

|  | **Black Box Testing** | **White Box Testing** |
| --- | --- | --- |
| Focus | Functionality | Internal Code and Logic |
| Knowledge Level | No knowledge of internal structure | Detailed knowledge of internal structure |
| Scope | Functional level | Unit level, Integration level, System level |
| Implementation | Done by testers | Done by developers |
| Testing Level | Functional testing | Unit testing, Integration testing, System testing |
| Testing Initiation | After requirements gathering | After coding |
| Programming | Not required | Required |
| Testing Focus | Input and Output | Code paths and logic flows |
| Applicability | Suitable for functional testing | Suitable for unit testing and integration testing |
| Alternative Names | Behavioural testing | Structural testing, Code-based testing |
| Time Consumption | Less time-consuming | More time-consuming |
| Suitable for Algorithm Testing | No | Yes |
| Approach | Top-down approach | Bottom-up approach |
| Example | Testing a login feature | Testing a sorting algorithm |
| Exhaustiveness | Less exhaustive | More exhaustive |

**Conclusion:** Black box testing and white box testing are two essential testing techniques used in software testing. Black box testing focuses on testing the functionality of the software without knowledge of its internal structure, while white box testing examines the internal code and logic of the software. Understanding the differences between these two techniques is crucial for selecting the appropriate testing strategy for a software project.