Software Testing

Tutorial Week 6

What is Software Testing?

Definition: Software testing is the process of evaluating and verifying that a software product meets the required standards and functions as expected.

Objectives:

- Identify and fix bugs.
- Ensure software reliability and performance.
- Validate user requirements.

Types of Testing

Manual Testing

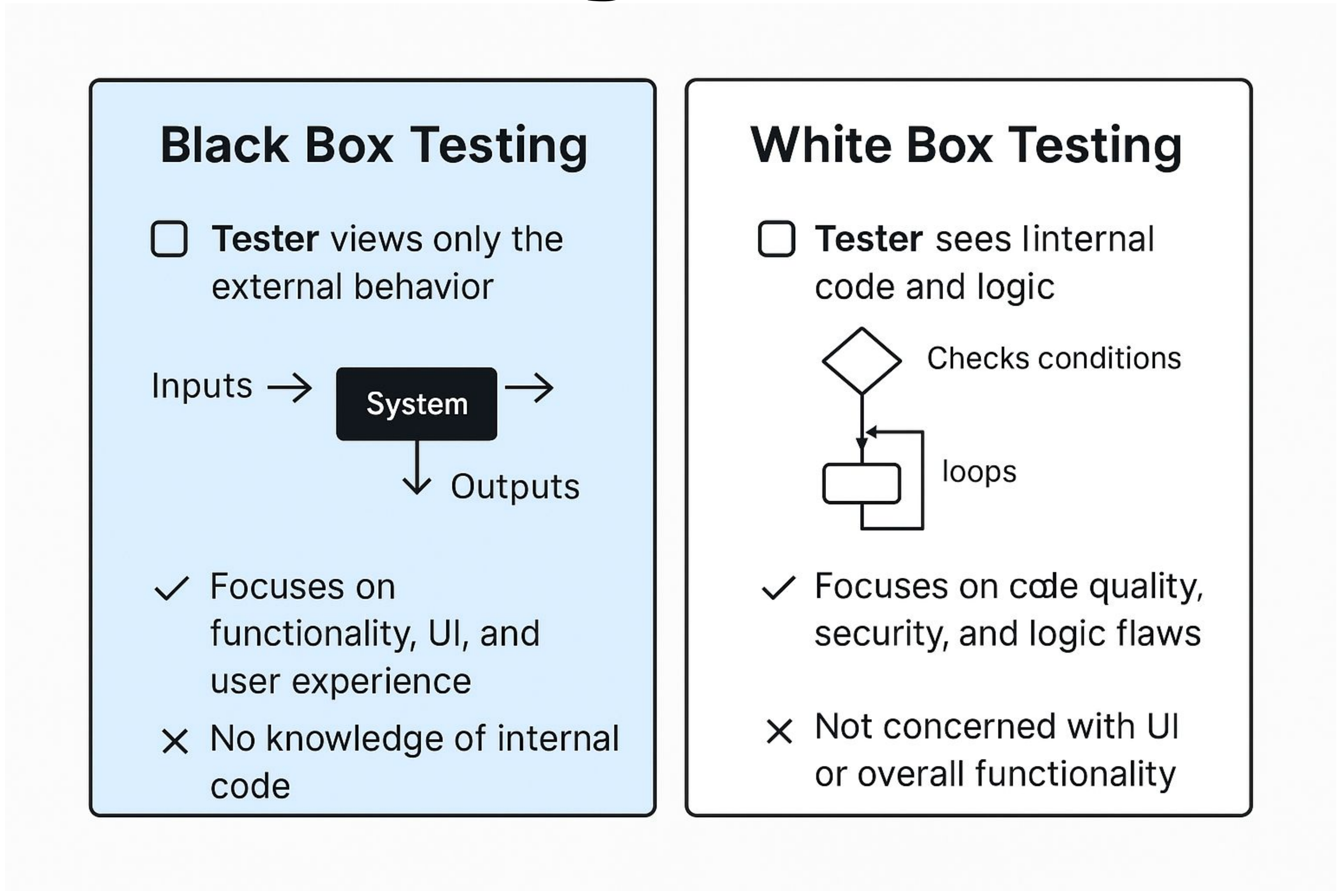
Automation Testing

Other Types of Testing

Functional Testing (Unit, Integration, System, UAT)

Non-Functional Testing (Performance, Security, Usability)

Black Box Testing and White Box Testing



Black Box Testing

☐ **Tester** views only the external behavior

Inputs → System →

↓ Outputs

✓ Focuses on functionality, UI, and user experience

✕ No knowledge of internal code

White Box Testing

☐ **Tester** sees linternal code and logic

Checks conditions

loops

✓ Focuses on code quality, security, and logic flaws

✕ Not concerned with UI or overall functionality

Why is Software Testing Important?

Prevents critical failures.

Improves user experience.

Saves time and resources.

Build customer confidence.

Fun Fact: The first software bug was a real bug—a moth found in a computer in 1947!

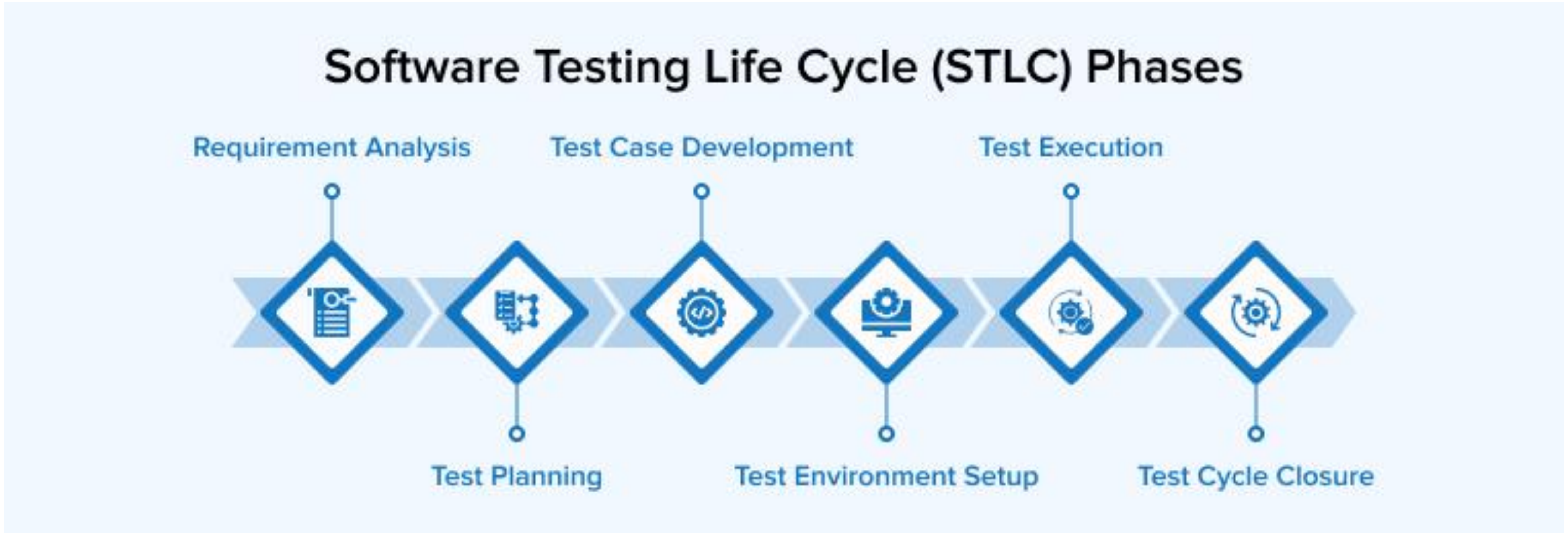
Why Do Bugs Happen?

- Miscommunication or unclear requirements.
- Programming errors.
- Lack of proper testing.

Bug Life Cycle

- Identification
- Assignment
- Fixing
- Testing
- Closure

Software Testing Life Cycle (STLC)



Software Testing Life Cycle (STLC)

- Requirement Analysis :
  - Understand & analyze testing requirements.
  - Identify testable aspects of the application.
- Test Planning:
  - Define scope, test strategy, and schedule.
  - Identify required tools and resources.
- Test Case Development:
  - Design test cases, test scripts, and test data.
  - Peer review and approval.
- Environment Setup:
  - Prepare test environment (hardware, software, network).
  - Ensure configuration aligns with requirements.
- Test Execution:
  - Execute test cases and log defects.
  - Compare actual vs expected results.
- Test Closure:
  - Analyze test results, prepare reports.
  - Document lessons learned for future improvements.

Introduction to Selenium

What is Selenium?
Selenium is a popular open-source framework for automating web application testing.

- Used for automating browser actions.
- Provides APIs for multiple programming languages.

Features:

- Supports multiple programming languages (Java, Python, etc.).
- Compatible with various browsers (Chrome, Firefox, etc.).
- Allows cross-browser testing.

Components of Selenium

Selenium WebDriver: A tool for automating browsers by directly interacting with them.

- Supports multiple programming languages (Java, Python, C#, etc.).
- Suitable for functional and regression testing.
- Works with different browsers like Chrome, Firefox, Edge, and Safari.

Selenium IDE: A record-and-playback tool for creating automation scripts.

- Best for beginners and quick test case creation.
- Available as a browser extension for Chrome and Firefox.
- Generates scripts that can be exported to WebDriver-compatible code.

Selenium Grid: Enables parallel test execution on multiple machines and browsers.

- Useful for distributed testing across different environments.
- Helps in running tests faster by leveraging multiple nodes.
- Works with Selenium WebDriver to execute tests remotely.

Advantages

FASTER TESTING PROCESS.

REDUCES HUMAN ERRORS.

REUSABLE TEST SCRIPTS.

CSS Selectors

Locator Type	Description	Syntax
ID	Locates an element using its unique ID attribute. This is one of the fastest and most reliable locator types.	driver.findElement(By.id("elementID"));
Name	Finds elements by their name attribute. This can be useful when multiple elements share the same name.	driver.findElement(By.name("elementName"));
Class Name	Targets elements based on their class attribute. This is useful for selecting multiple elements that share a class.	driver.findElement(By.className("className"));
Tag Name	Locates elements by their tag name (e.g., input, button, div). This can be used to find all elements of a specific type.	driver.findElement(By.tagName("tagName"));
Link Text	Specifically used to locate anchor (<a>) elements by their visible text. This is helpful for finding links.	driver.findElement(By.linkText("Visible Text"));
Partial Link Text	Similar to Link Text, but allows for partial matching of the anchor text, making it useful for long or dynamic link text.	driver.findElement(By.partialLinkText("Partial Text"));
XPath	A powerful and flexible locator that uses XML path language to navigate through elements and attributes in the DOM.	driver.findElement(By.xpath("//tag[@attribute='value']"));
CSS Selector	Utilizes CSS selectors to find elements. This is often faster than XPath and can be very expressive for complex selections.	driver.findElement(By.cssSelector("cssSelector"));

Steps To Get Started As a QA/Tester

LEARN THE BASICS OF SOFTWARE TESTING.

FAMILIARIZE YOURSELF WITH SELENIUM TOOLS.

PRACTICE WRITING AUTOMATED TEST SCRIPTS.

EXPLORE ADVANCED FEATURES LIKE SELENIUM GRID.

Project Simulation

Activities

Create a Test Case and a Bug Report for the following Scenario

Students will create a Test Case and a Bug Report for the following case:

https://dribbble.com/session/new
Dribbble is a website where designers, illustrators, and other creative professionals showcase their work. It's a platform for sharing and discovering design-related content, including graphics, animations, and user interface designs. Users can upload their portfolios, browse other designers' work, and connect with fellow creatives. It's popular in the design community for inspiration, feedback, and networking.

1. Please create a test case for the following Functionalities.

- Sign up
- Login to the system
- Search box

Sample: <LINK TO THE SAMPLE TEST CASE AND BUG REPORT>

Automation Testing: Install a Selenium library

Setting up the Selenium library for Python.

```
pip install selenium
pip install selenium-x.x.x.-py3-none-any.whl
```

Write Your first Selenium Script

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

# Set up the WebDriver (Make sure you have the correct driver installed)
driver = webdriver.Chrome()
driver.get("https://www.google.com")

# Find the search box and input the query
search_box = driver.find_element(By.NAME, "q")
search_box.send_keys("Selenium Tutorial")
search_box.send_keys(Keys.RETURN)

# Wait for results to load
time.sleep(2)

# Close the browser
#driver.quit()
```

Selenium Task

Use the following credentials to test the following test case using Selenium.
URL: cms.periwin.com
Username: colab@hck.com
Password: Hck321#@!

Algorithm:
Initialize WebDriver: Open a web browser using Selenium.
Navigate to the Login Page: Load the URL of the login page.
Locate Username & Password Fields: Identify input fields using locators (ID, Name, XPath, etc.).
Enter Credentials & Submit:

1. Input username and password.
2. Click the login button.

Verify Login Success or Failure

3. Check if the **dashboard/home page** is loaded (success).
4. If login fails, capture the error message.

Close the Browser

5. End the test session.

Code Link for Selenium Code:

Thank You