Introduction to Software Testing

This section lays the groundwork for the entire course. **Software Testing** is the process of evaluating a software system to ensure it meets specified requirements and to find defects. The syllabus covers the history of testing and why it's so crucial—to identify bugs early, ensure quality, and prevent failures. You'll learn to describe a

defect, which is a deviation from the expected behavior, and understand the **Principles of Testing**, such as testing showing the presence of defects, not their absence. The distinction between

Quality Assurance (QA), which is process-oriented, and **Quality Control (QC)**, which is product-oriented, is also a key part of this module. Other topics include the scope and constraints of testing, the specific

roles of a Software Tester, and overviews of the **Software Development Life Cycle (SDLC)** and **Software Testing Life Cycle (STLC)**.

AGILE Testing, a method that involves testing continuously within an Agile development environment, is also introduced.

Test Planning

Test planning is the strategic phase where you define the testing approach. You'll learn to create a

Test Strategy and Planning document. This involves customizing the test process to fit the project's needs, managing the testing

Budget, and creating a **Schedule** to ensure tests are completed on time. This module also covers

Risk and Configuration Management, which involves identifying potential risks to the project and setting up the test environment.

Design of Testing

This module focuses on the creative part of testing: designing the actual tests. You'll learn how to create

Test Scenarios and detailed **Test Cases**. A

Test Scenario is a high-level idea of what to test (e.g., "test the login functionality"), while a **Test Case** provides the specific steps, input data, and expected results (e.g., "Step 1: Enter valid username; Step 2: Enter valid password; Expected Result: User logs in successfully"). The syllabus also covers how to create

Test Data and explains the difference between a test case and a test scenario. You'll also learn about the

Traceability Matrix, a document that links test cases to requirements to ensure all requirements are tested.

-

Types of Testing

This module describes various types of testing based on their purpose:

- **Regression Testing**: Ensuring that new code changes don't break existing functionality.
- **Smoke Testing**: A quick test to verify if the core functionalities are working after a new build.
- **Database Testing**: Verifying data integrity and the functionality of the database.
- **Performance, Load, and Stress Testing**: These non-functional tests check how the system performs under varying conditions.
- **Compatibility Testing**: Ensuring the software works correctly on different operating systems, browsers, and devices.
- **Security Testing**: Identifying vulnerabilities in the application.
- **Usability Testing**: Evaluating how user-friendly the application is.
- **Internationalization and Localization Testing**: Ensuring the software can be adapted for different languages and regions.

Executing and Managing Defects

This section focuses on the practical execution of tests and handling defects. You'll learn about the

Build and Release process, reading **Release Notes**, and using a **Pre QA Checklist**. The concepts of

Entry and Exit criteria are also covered, which define when to start and stop a testing phase. A key part of this module is

Managing Defects, including **Defect Prevention**, **Defect Discovery**, and the **Defect Life Cycle** (from discovery to closure). You'll also learn about

Severity (the impact of a defect) and **Priority** (how quickly it needs to be fixed). The module includes a hands-on component using a tool like

Bugzilla to identify and log defects.

Team Collaboration and Reporting

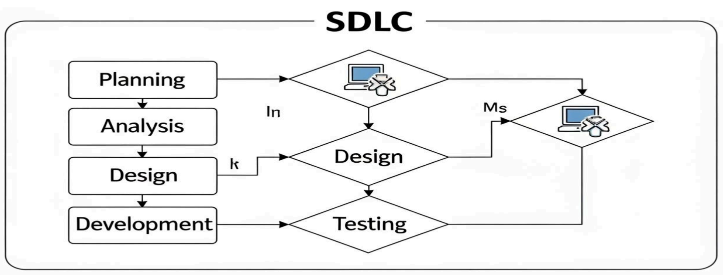
The final topics emphasize communication and metrics. You'll learn to create

Test Status Reports and **Test Closure Reports** to keep stakeholders informed. This module also highlights the importance of collaboration between testers and developers, client interaction, and working in models like

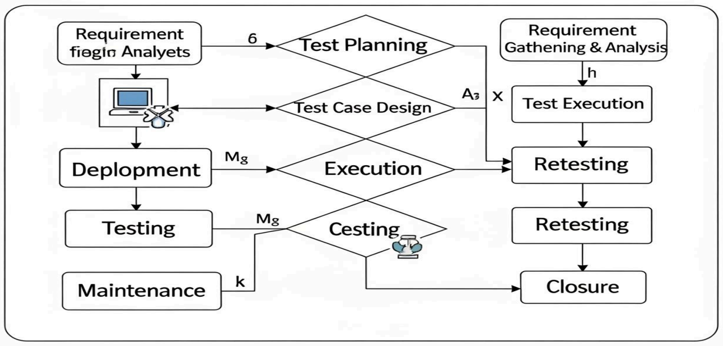
Onshore/Offshore. The last part of the syllabus covers

Measurements and Metrics , including their benefits, life cycle, and different types, to help measure testing effectiveness and progress.

Software Development Life Cycle

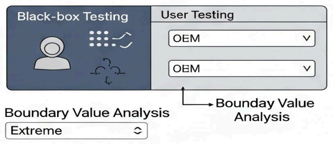


STLC
Software Testing Life Cycle

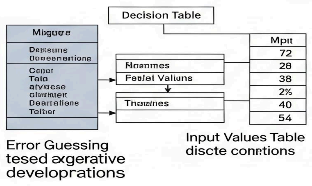


Techniques and Levels of Testing

Dynamic Testing



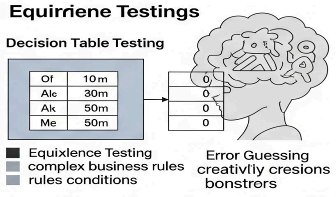
Equivalence Partitioning


Decision Table
Mbguee
Detsuns Doueenetions
Coner Talo alvaese olsnbrert Doarratione Taliner
Hoemnes
Feulat Valiuns
Theawines
Error Guessing tesed xgerative developrations
Input Values Table discte conrtions

Mpιt
72
28
38
2%
40
54

Static Testing

Equirriene Testings



Of	10m
Alc	30m
Ak	50m
Me	50m

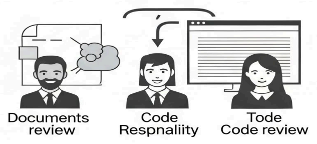
0
0
0
0

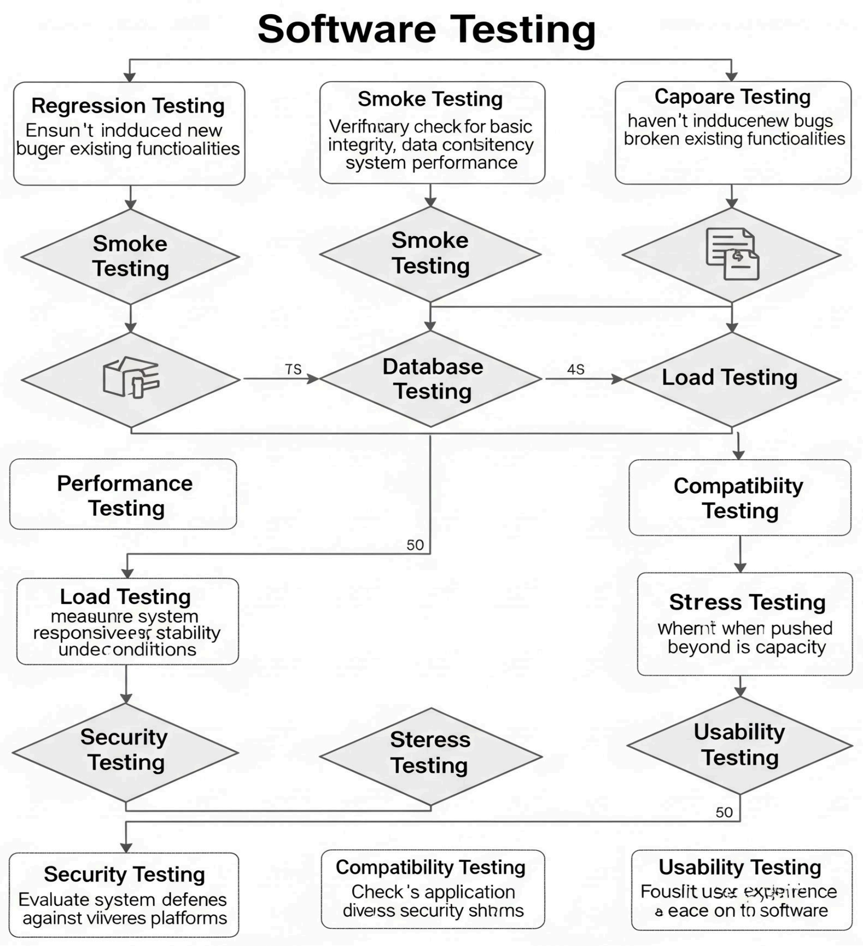
- ■ Equixlence Testing
- ▨ complex business rules
- ▨ rules conditions

Experience-based Tester
Exploratory Testing

Static Testing

STLC




Software Testing
Regression Testing
Ensun't indduced new buger existing functioalities
Smoke Testing
Verifnary check for basic integrity, data contsitency system performance
Capoare Testing
haven't indducenew bugs broken existing functioalities
Smoke Testing
Smoke Testing
7S
Database Testing
4S
Load Testing
Performance Testing
Compatibiity Testing
50
Load Testing
meaunre system responsiveesr stability undecondiitions
Stress Testing
whent when pushed beyond is capacity
Security Testing
Steress Testing
Usability Testing
50
Security Testing
Evaluate system defenes against viiveres plafforms
Compatibility Testing
Check's application divess security shms
Usability Testing
Fouslit usec expeirence a eace on to software