L2 TASK

TESTING ENGINEERING {TESTOPS}

Presented by Anshu Anand

SOFTWARE TESTING

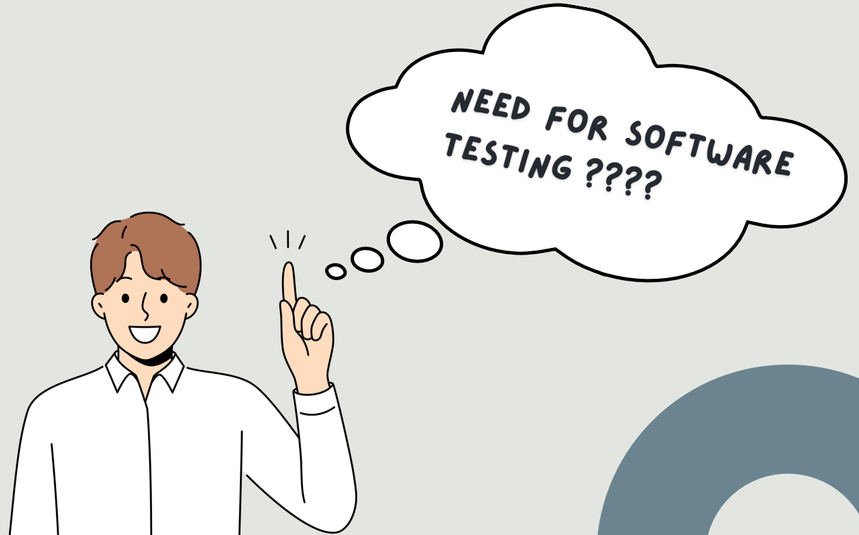
What is software testing ?

Software testing is an important process in the software development lifecycle. It involves **verifying** and **validating** that a software application is free of bugs, meets the technical requirements set by its design and development, and satisfies user requirements efficiently and effectively.

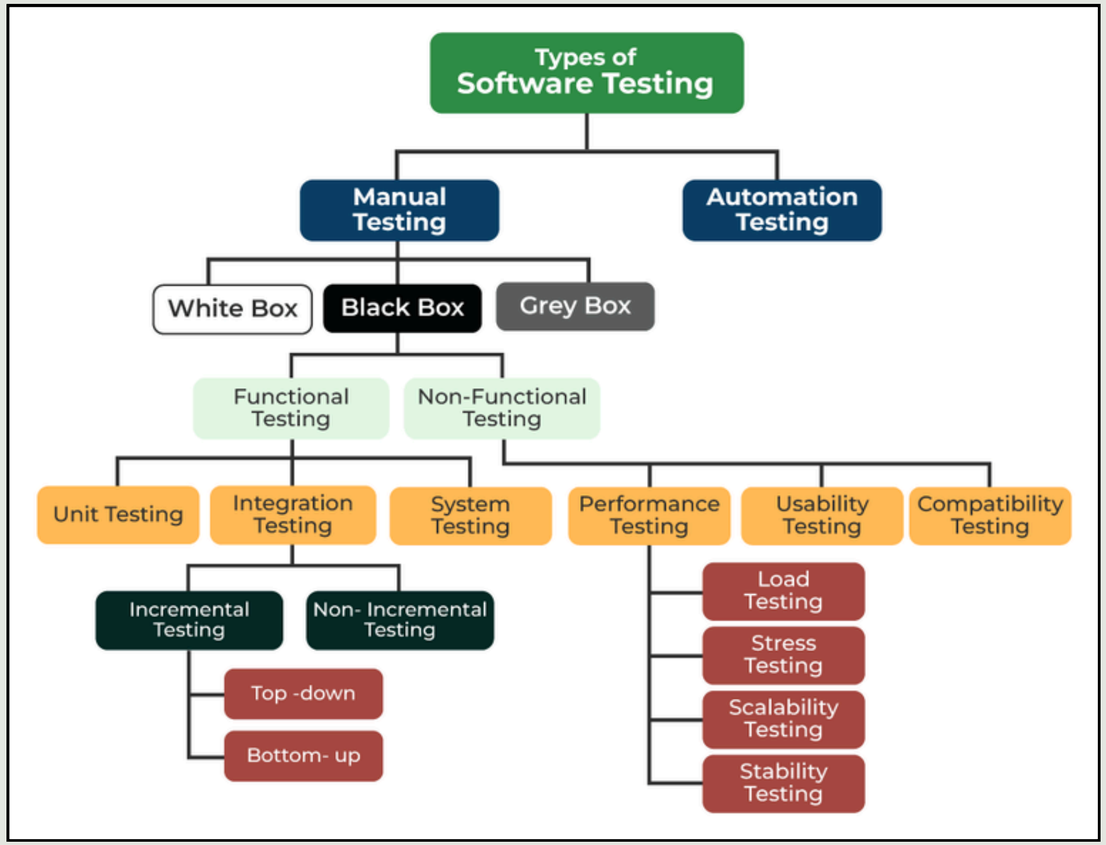
- **Verification:** It ensures that the software is being built correctly according to design specifications and requirements. Example: It's like checking the recipe while cooking to make sure you are following each step correctly and adding the right ingredients in the right amounts.
- **Validation:** It ensures that the final software product meets the user's needs and expectations. Example: It's like having someone taste the dish after it's cooked to ensure it tastes good and meets their expectations.

Importance of Software Testing :

- Defects can be Identified Early
- Improves Quality of Software
- Increased Customer Satisfaction
- Helps with Scalability
- Saves Time and Money



TYPES OF TESTING



- **Manual Testing:** It includes testing software manually, i.e., without using any automation tool or script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it. Example: A tester might manually log in to a website, navigate through different pages, and fill out forms to ensure everything works correctly.
- **Automation Testing**: It is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing. Example: Using an automation tool like Selenium to automatically test the login functionality of a web application across different browsers.

TYPES OF TESTING

Black-Box Testing

Testing the software without any knowledge of the internal workings or code. Testers focus on the input and output of the software.

Example: A tester verifies that a login feature works by entering valid and invalid credentials and checking the response.

White-Box Testing

Testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing. **Example:**A tester checks the code to ensure that all possible branches and loops in an algorithm are tested.

Functional Testing

Functional testing verifies that the software functions as expected according to the requirements and specifications. It focuses on what the system does.

Example: Testing the login functionality of an app by entering valid and invalid credentials to see if it allows access to authorized users and denies unauthorized access.

Non-Functional Testing

Non-functional testing evaluates aspects of the software that are not related to specific functions or features, such as performance, usability, reliability, and scalability. It focuses on how the system performs.

Example: Testing the app's performance by simulating a large number of users to see how it handles high traffic and whether it remains responsive.

FUNCTIONAL TESTING

Unit Testing

Unit testing involves testing individual components or modules of a software application in isolation to ensure they work as expected.

Example: Testing a function that calculates the sum of two numbers to ensure it returns the correct result for various input values.

Integration Testing

Integration testing focuses on verifying the interactions and data flow between integrated components or modules of a software application.

Example: Testing the interaction between the login module and the user profile module to ensure that user data is correctly passed and displayed after login.

System Testing

System testing involves testing the entire software application as a whole to ensure it meets the specified requirements and works correctly in its entirety.

Example: Testing an e-commerce website to ensure that users can search for products, add items to the cart, proceed to checkout, make payments, and receive order confirmations.

Acceptance Testing

Acceptance testing is the final phase of testing before the software is released to the users. It is conducted to ensure that the software meets the business requirements and is ready for deployment. This type of testing is performed by end-users or stakeholders to validate that the software is fit for purpose and satisfies their needs.

NON FUNCTIONAL TESTING

Performance testing

Performance testing is a type of software testing that evaluates how well a system performs under specific conditions. Its primary goal is to measure system responsiveness, scalability, and stability to ensure it can handle real-world workloads.

Load testing

- This tests the system's behavior under expected user load.
- Example: Checking how an e-commerce website like Amazon performs during a regular shopping day with thousands of users making purchases simultaneously.

Stress Testing

- This evaluates the system's behavior under extreme or unexpected load conditions, beyond the normal operating capacity.
- Example: Testing a video conferencing platform like Zoom when a massive number of participants join a global webinar simultaneously.

Scalability testing

- This measures the system’s ability to scale up or down based on demand.
- Example: Ensuring a cloud storage service like Google Drive can expand resources efficiently when user data requirements increase.

.

Stability testing

- Stability testing, often considered a part of endurance testing, focuses on determining whether a software system can consistently perform under a fixed workload over an extended period.
- Example: Testing an online food delivery app like Zomato to ensure it remains stable and responsive when orders are continuously being placed for 24-48 hours without downtime.

Alpha testing

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance tests. It is the first stage of software testing, during which the internal development team tests the program before making it available to clients or people outside the company.

Beta Testing

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one type of User Acceptance Testing. A pre-release version of the product is made available for testing to a chosen set of external users or customers during the second phase of software testing.

Smoke testing

Smoke Testing is a type of preliminary testing performed after a new build of software is developed. It is conducted to verify that the critical functions of the application work as expected and that the build is stable enough for more detailed testing.
Imagine you're testing an e-commerce website after a new build:

- Check if the homepage loads.
- Verify that users can log in.
- Test whether a product can be searched, added to the cart, and proceed to checkout.

If any of these critical functionalities fail, the build is rejected and sent back to developers for fixes.

CX TESTING

CX Testing **(Customer Experience Testing)** focuses on evaluating and improving the overall experience customers have when interacting with a product, service, or brand. It ensures that every touchpoint in the customer journey is seamless, intuitive, and satisfying.

Microservice testing

Microservice is a small, independent piece of a bigger application. Think of an app like Amazon—it's not one big program, but instead made of many small services like:

- Search Service: Helps you find products.
- Cart Service: Manages items you add to the cart.
- Payment Service: Processes your payments.

Each of these "microservices" works independently but communicates with other services to form the whole system.

When testing microservices, you check:

1. Each Service Independently:
2. How Services Work Together:
3. APIs (Communication):(Since microservices talk to each other through APIs, you need to test these APIs to ensure proper data flow.

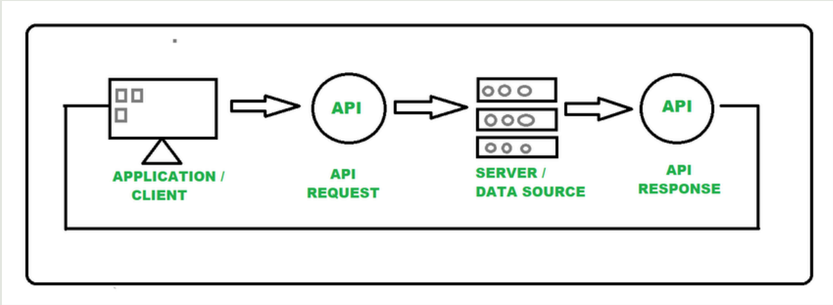
API TESTING

API testing, or application programming interface testing, is a type of software testing that focuses on the testing of individual API methods and the interactions between different APIs. This type of testing is typically performed at the **integration level**, after unit testing is completed, and before user interface testing begins. It is used to validate that the API behaves correctly and that it meets the requirements of the system.

MOBILE TESTING

Mobile Application Testing is an important process in software development that focuses on ensuring the quality, functionality, usability, security, and performance of mobile applications across various devices and platforms.

Example: Testing a ride-hailing app like Uber to confirm it works well on both Android and iOS devices, supports GPS navigation, and remains fast under low network bandwidth.



Domain Driven Testing

Domain-Driven Testing focuses on real-world rules and logic specific to the business or industry (known as the "domain") you are working with. The goal is to make sure the software behaves exactly how the business or users expect it to work in their specific context. **Imagine you're testing a banking application:**

1. In banking, there are specific rules like:
    - Interest rates on loans should calculate based on the agreed terms.
    - Withdrawal limits depend on the type of account.
2. Domain-Driven Testing ensures:
    - These banking rules (specific to the domain) are tested thoroughly.
    - Developers and testers work closely with domain experts (e.g., bankers) to ensure the application matches the real-world banking processes.

Keyword-Driven Testing

This approach uses keywords (action words) to represent specific testing steps. These keywords are predefined, and testers don't need coding knowledge to create test cases.

How it works:

- Keywords like "Login", "Search", or "Checkout" are assigned actions.
- Testers create scripts using these keywords, and automation tools execute them.

TDD - Test-Driven Development:

A development practice where you write the test cases before the code.How it works:

1. Write a test that describes how a feature should behave.
2. Develop code to pass that test.
3. Refactor the code while keeping the test passing.

- Example: Before coding a "Login" feature, you'd first write a test to check if the system validates credentials. Then, build the feature to pass the test.

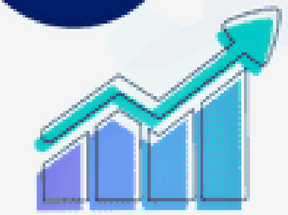
BDD - Behavior-Driven Development:

- Extends TDD by focusing on user behavior. It's more collaborative and written in plain language so business teams, testers, and developers can understand.
- How it works:
- Uses human-readable syntax like Gherkin to write test scenarios.
- Example format: Given: A user is on the login page. When: They enter correct credentials. Then: They are redirected to the dashboard.

SDLC vs. STLC

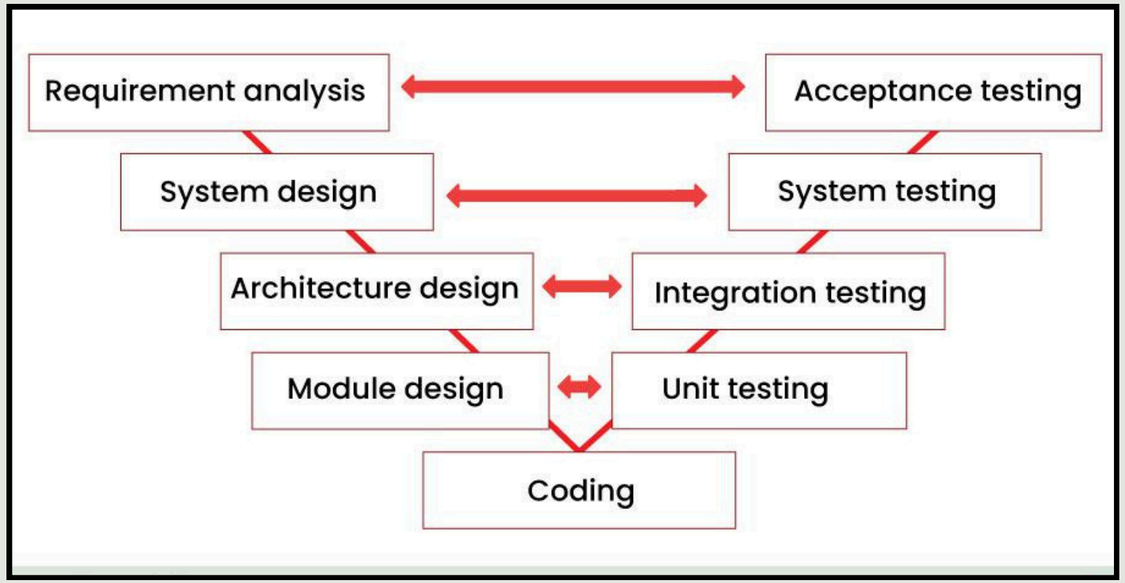

Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle


Requirement Analysis
Test Planning
Test Design
Environment Setup
Test Execution
Test Closure
Software Testing Life Cycle

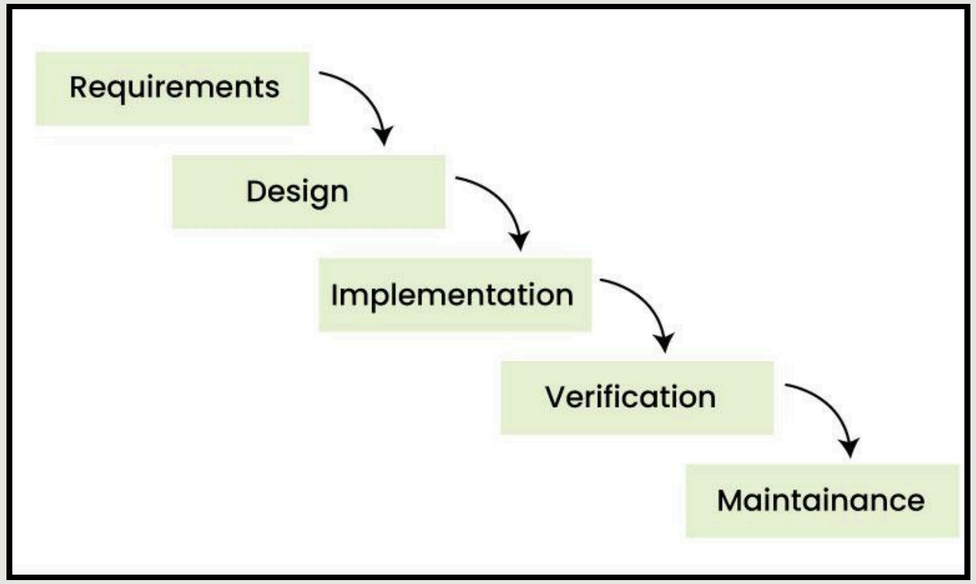


SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

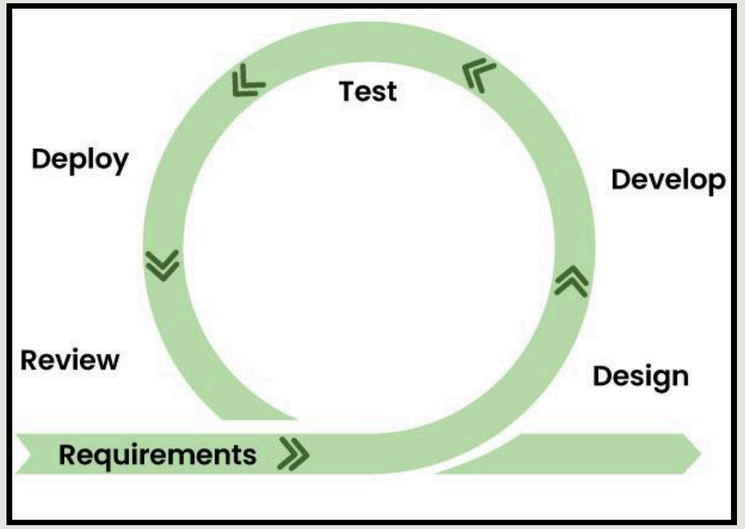
V-Model



Waterfall Model

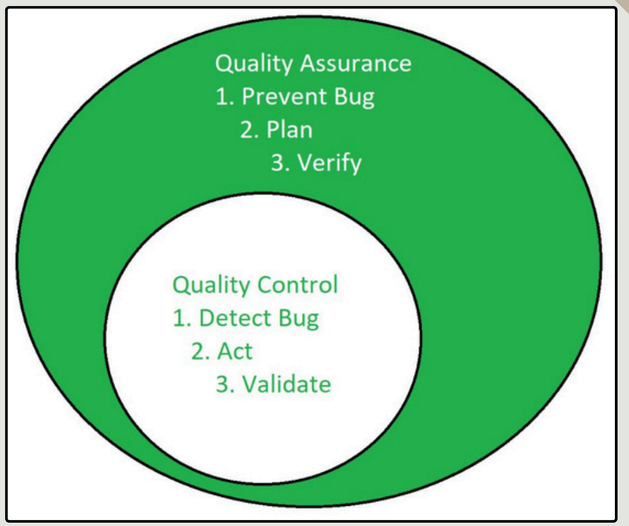


Agile Model



QUALITY ASSURANCE (QA) VS QUALITY CONTROL (QC)

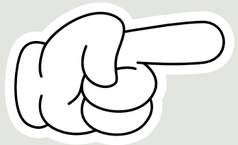
Quality Assurance (QA) and Quality control (QC) are both important methods in software engineering to get high-quality software. Quality Assurance (QA) prevents software defects or minimizes the number of defects in software before delivery by making sure that proper methods and processes are followed during the software development process. Whereas Quality Control (QC) identifies and fixes the defects or errors that exist after development.



WHAT IS TESTOPS ?

TestOps (Testing Operations) integrates testing into the software delivery lifecycle, ensuring seamless collaboration between development, operations, and QA teams. It focuses on planning, managing, and analyzing testing activities to optimize quality and delivery speed.

Future Trends of Testing



The future of testing is shaped by technological advancements and evolving methodologies:

- AI-Driven Testing: Leveraging AI for predictive analytics, self-healing scripts, and intelligent test case generation.
- Continuous Testing: Integrating testing into CI/CD pipelines for real-time validation.
- Scriptless Testing: Using low-code/no-code platforms to enable non-technical testers.
- Cybersecurity Testing: Emphasizing security in the development lifecycle.
- Digital Twin Testing: Simulating real-world scenarios for IoT and complex systems.

ROLE OF TEST AUTOMATION

Test automation plays a critical role in modern software development:

- Efficiency: Reduces manual effort and accelerates testing cycles.
- Consistency: Ensures repeatable and reliable test execution.
- Scalability: Handles large-scale testing for complex systems.
- Cost-Effectiveness: Identifies defects early, reducing overall costs.

QUALITY ENGINEERING AND PERFORMANCE ENGINEERING

Quality Engineering is a holistic approach to ensuring software quality is embedded into every stage of the development lifecycle. Unlike traditional QA, which focuses mainly on testing after development, QE integrates quality practices from design to delivery.

Key Aspects of Quality Engineering:

- Shift-Left Testing: Testing starts early in the development process to catch issues early.
- Continuous Testing: Integrating testing into CI/CD pipelines for ongoing validation.
- Automation: Emphasis on automated tests to improve efficiency and reduce human error.
- Collaboration: Ensures close collaboration between developers, testers, and business teams.
- Metrics and Analytics: Uses data-driven insights to monitor and improve software quality.

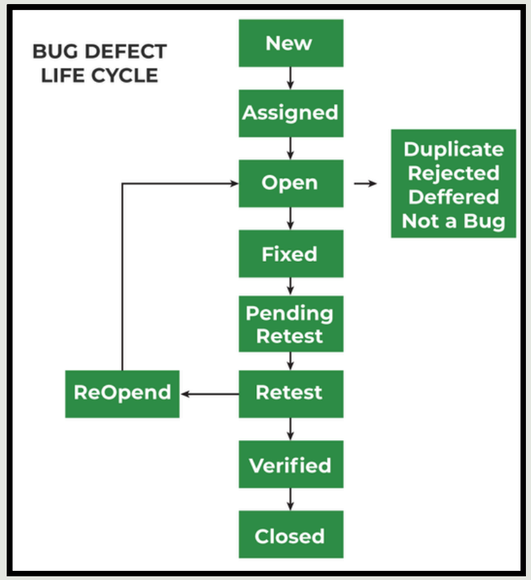
Performance Engineering focuses on building reliable, scalable, and efficient systems. It goes beyond performance testing by proactively addressing performance at all stages of the software lifecycle, including design, development, and testing.

Key Aspects of Performance Engineering:

- Performance Planning: Defining performance goals and benchmarks early in the project.
- Proactive Design: Creating system architectures that optimize performance (e.g., load balancing, caching strategies).
- Continuous Monitoring: Tracking system performance in real time to identify bottlenecks.
- Optimization: Fine-tuning code, databases, and infrastructure to achieve desired performance levels.
- Scalability: Ensuring the system can scale up or down efficiently based on demand.

WHAT IS DEFECT TRACKING ?

Defect tracking, is the systematic process of identifying, recording, monitoring, and managing defects or issues in a product or system throughout its development lifecycle. These defects can encompass various aspects, including software bugs, hardware malfunctions, design flaws, or other imperfections that may hinder the product's functionality, performance, or quality.



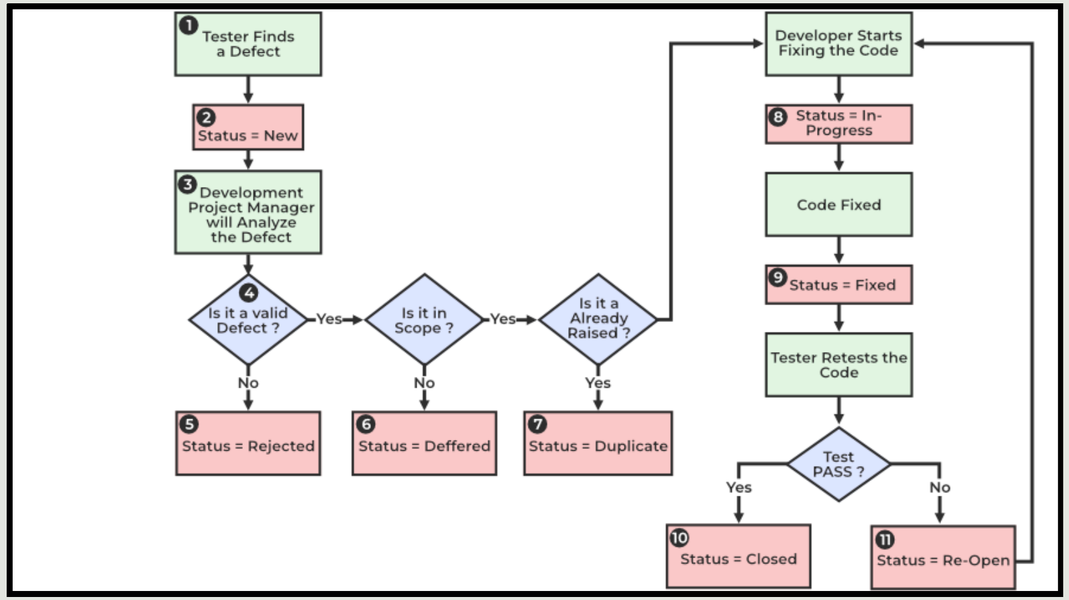
DEFECT AND BUG LIFE CYCLE

What is a Bug/Defect?

A defect is an error or bug in an application that is created during the building or designing of software due to which software starts to show abnormal behaviors during its use.

What is the Defect Life Cycle?

In the Software Development Process, the Defect Life Cycle is the life cycle of a defect or bug that it goes through covering a specific set of states in its entire life. Mainly bug life cycle refers to its entire state starting from a new defect detected to the closing off of that defect by the tester. Alternatively, it is also called a Bug Life Cycle.



SEVERITY AND PRIORITY

Severity:
Severity is defined as the extent to which a particular defect can create an impact on the software.
Levels:

- Critical: A major feature is broken (e.g., payment fails on an e-commerce app).
- High: Significant impact but not blocking (e.g., product details missing).
- Medium: Less impact, affects usability (e.g., alignment issues).
- Low: Minor defect with negligible impact (e.g., a typo in text).

Priority:
Priority is defined as a parameter that decides the order in which a defect should be fixed. Defects having a higher priority should be fixed first.
Levels:

a. High Priority: Needs immediate attention (e.g., the login function fails).
b. Medium Priority: Can wait but should be fixed soon.
c. Low Priority: Can be addressed later without affecting users.

TEST PLAN

A Test Plan is a detailed document that outlines the testing strategy, objectives, resources, schedule, and scope for a project.
Key Elements of a Test Plan:

- Purpose: Why the testing is needed.
- Scope: What features or functionalities will be tested.
- Test Strategy: Methods and types of testing (e.g., functional, performance).
- Resources: Required tools, team members, and testing environments.
- Schedule: Timeline for executing tests.
- Entry and Exit Criteria: Conditions to start and stop testing.
- Risks: Possible challenges and mitigation strategies.

TEST CASE AND TEST SCENARIO

TEST CASE

Test cases are sets of actions that are executed to verify particular features or functionality of software applications. It consists of test data, test steps, preconditions, and post conditions that are developed for specific test scenarios to verify any requirement.

Example: Test Case for Login Functionality:

- Step: Enter valid username and password.
- Expected Result: The user is redirected to the homepage.

TEST SCENARIO

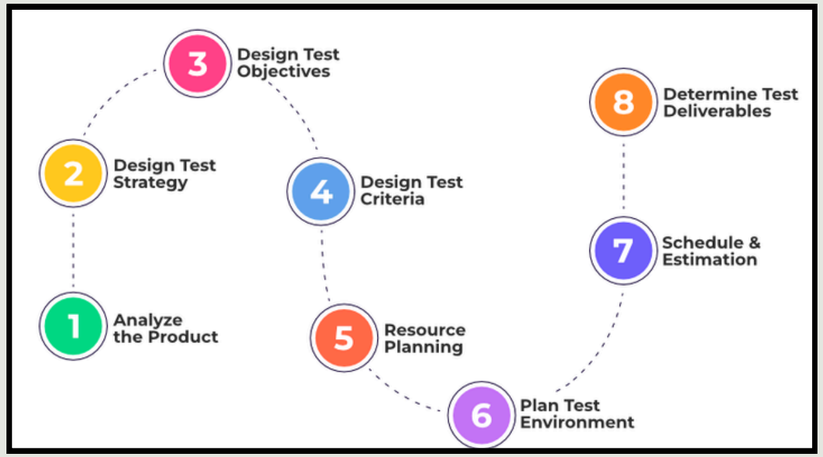
A test scenario is a collective set of test cases that helps the testing team determine the positive and negative features of the project.

Example of a Test Scenario:
For an e-commerce site:

- Scenario: Verify the checkout process.
  - This may include test cases like "Verify adding a product to the cart," "Verify entering payment details," and "Verify successful order placement."

HOW TO CREATE A TEST PLAN

Analyze the product: This phase focuses on analyzing the product, Interviewing clients, designers, and developers, and performing a product walkthrough.
Choose a Testing Strategy: Select testing types (e.g., manual, automated, functional, non-functional).
Define Objectives and Scope:Determine what needs to be tested (e.g., specific features or modules).
Define test criteria: Two main testing criteria determine all the activities in the testing project: Suspension criteria, Exit criteria



HOW TO CREATE A TEST PLAN (CONTINUE..)

Resource planning: This phase aims to create a detailed list of all the resources required for project completion. For example, human effort, hardware and software requirements, all infrastructure needed, etc.

Plan test environment: This phase is very important as the test environment is where the QAs run their tests. The test environments must be real devices, installed with real browsers and operating systems so that testers can monitor software behavior in real user conditions.

Schedule and Estimation: Break down the project into smaller tasks and allocate time and effort for each task. This helps in efficient time estimation.

Determine test deliverables: Test deliverables refer to the list of documents, tools, and other equipment that must be created, provided, and maintained to support testing activities in the project.

Thank You

For your attention