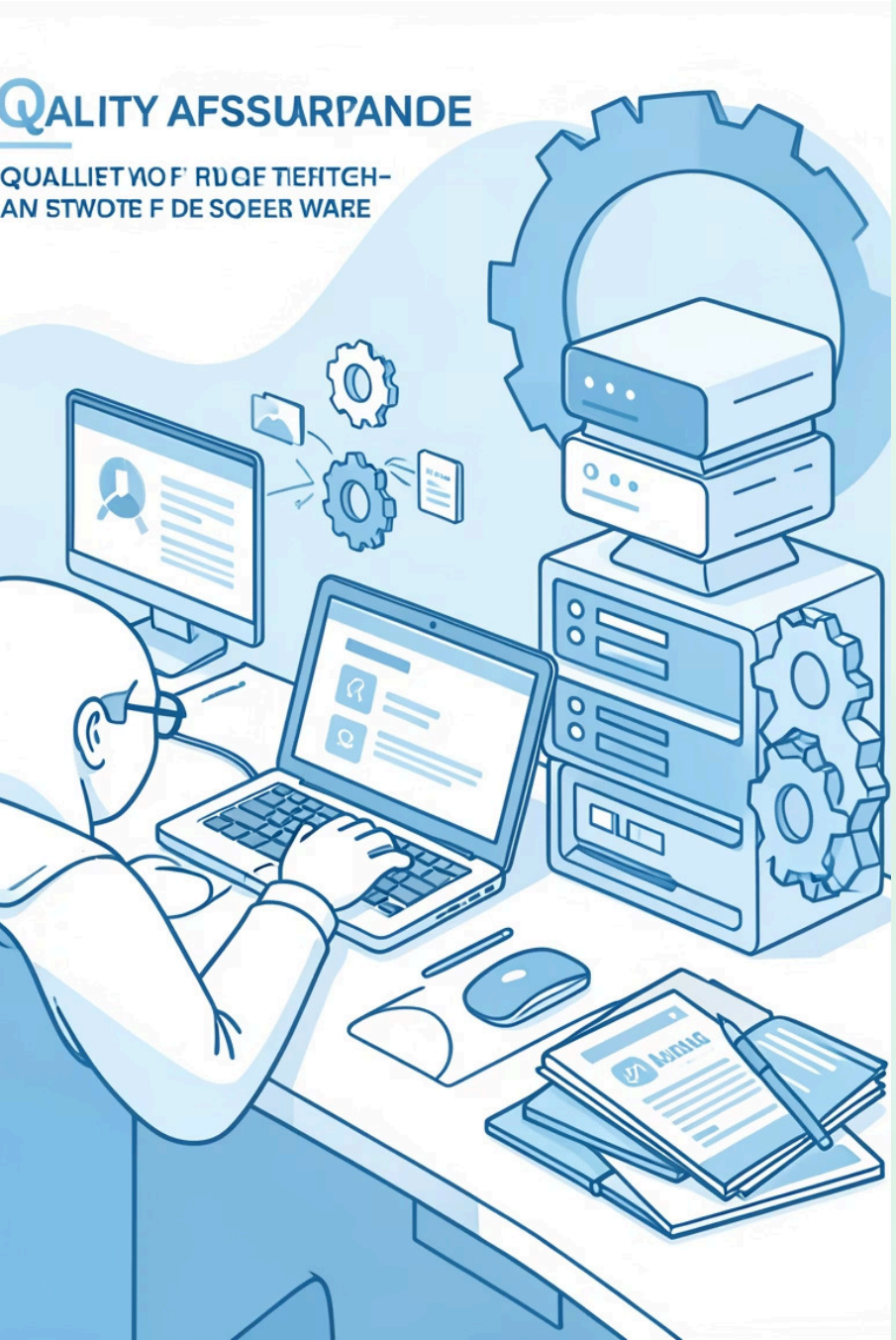
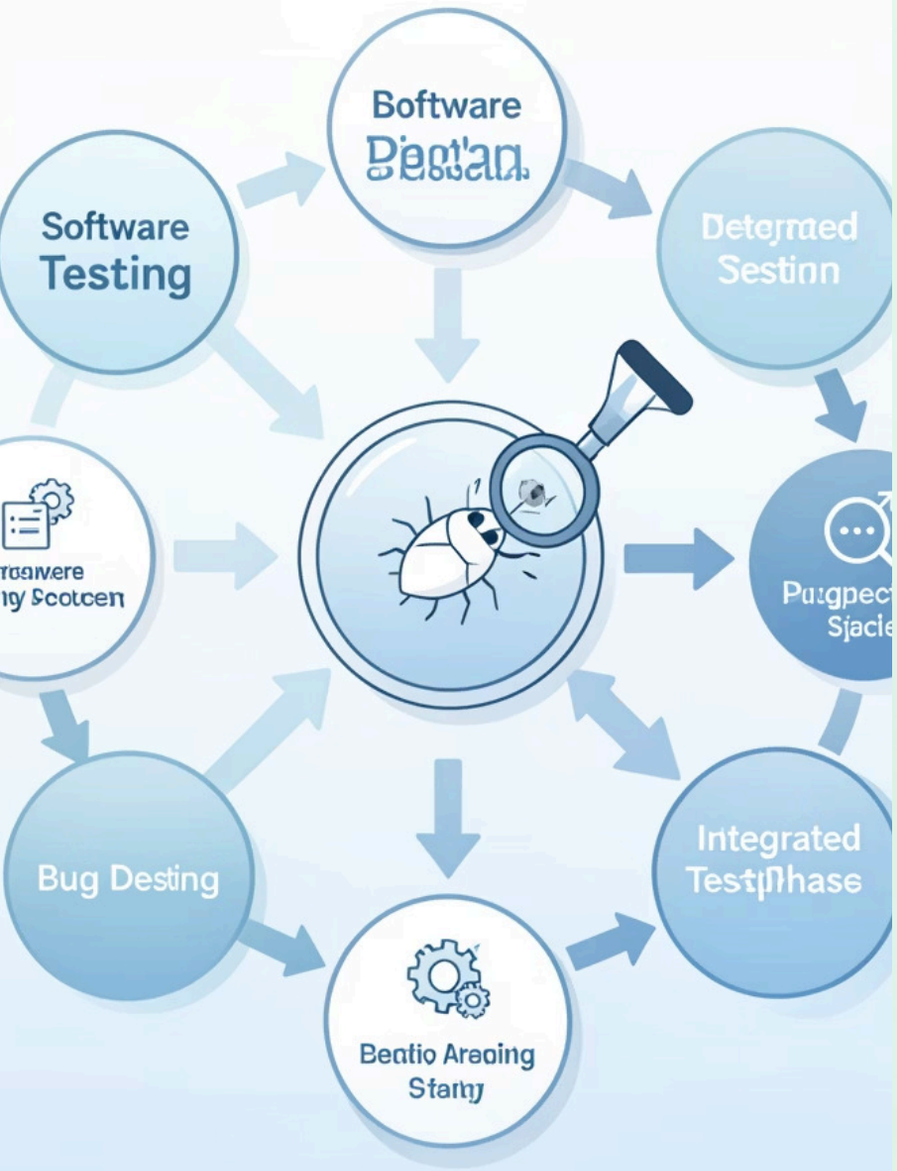


Quality Assurance (QA) Training – Day 2

BIT – 5th Semester (Professional Class)

Facilitator: **Neha Rouniyar**



What Are Testing Types?

Software testing is a big part of making software. It helps make sure the software works well and you can count on it. We use different tests to check different parts of the software.

Different Ways

Each type of test uses special ways to look at certain parts of the software.

Full Check

All these tests together check if the software works right, how fast it goes, and if it's easy to use.

Find Problems Early

When we test early, we find and fix problems fast. This saves time and money.

Software Testing: The Big Picture

Software testing is essential for creating reliable applications. It involves two main approaches: manual and automated testing. Both are crucial for checking how well software works and performs throughout its development, but they use different methods to get the job done.

Manual Testing

Human-driven validation of functionality

Automation Testing

Scripted tests run by tools

Exploratory

Ad-hoc and exploratory test sessions

CI Integration

Automated tests in pipelines

Manual Testing: Testing Done by People

Manual testing is a fundamental approach where human testers meticulously execute test cases without the aid of automation tools. This method relies on human observation, intuition, and experience to identify defects, validate functionality, and ensure a smooth user experience.

White Box Testing

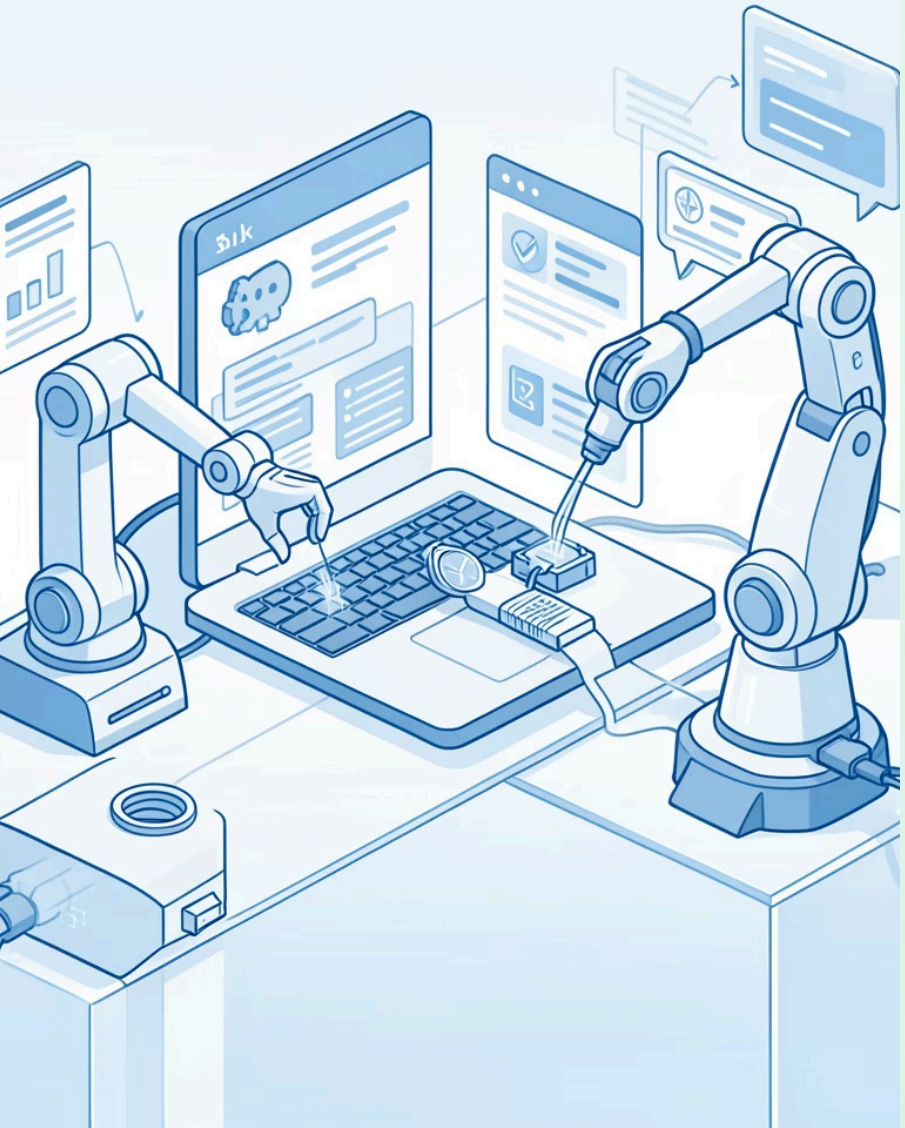
Testers delve into the internal structure, design, and coding of the software to verify its logic and functionality. This requires programming knowledge.

Black Box Testing

Testers evaluate the software from an end-user perspective without any knowledge of its internal code structure. They focus on input and output behavior.

Grey Box Testing

A hybrid approach where testers have partial knowledge of the internal code or design, allowing them to design more intelligent test cases while still maintaining a user-focused perspective.



Automation Testing: Testing Done by Tools

Automation testing uses special software tools and scripts to automatically execute test cases. This approach helps find bugs quickly and consistently, making it ideal for large and complex software projects that require frequent and repetitive checks.

Functional Testing

Checks if the software performs its intended functions correctly, according to specifications.

Non-Functional Testing

Evaluates software for non-functional aspects like performance, usability, reliability, and security.

Functional Testing: Does It Work Right?

Functional testing checks if the software does what it's supposed to do. It makes sure the program works as planned for the people who use it.

1

Unit Testing

Checks each small part of the software.

2

Integration Testing

Looks at how connected parts work together.

3

System Testing

Checks the whole software program.

Unit Testing: Checking Small Parts in Detail

Unit testing is the first and most basic level of functional testing. It checks each small, separate part of a program (called a 'unit') in isolation. Each unit is tested alone to make sure it works correctly.

1

Checks Small Parts

Tests individual functions or methods to ensure they work as intended.

2

Developer Responsibility

Developers typically write and execute these tests during the coding phase.

3

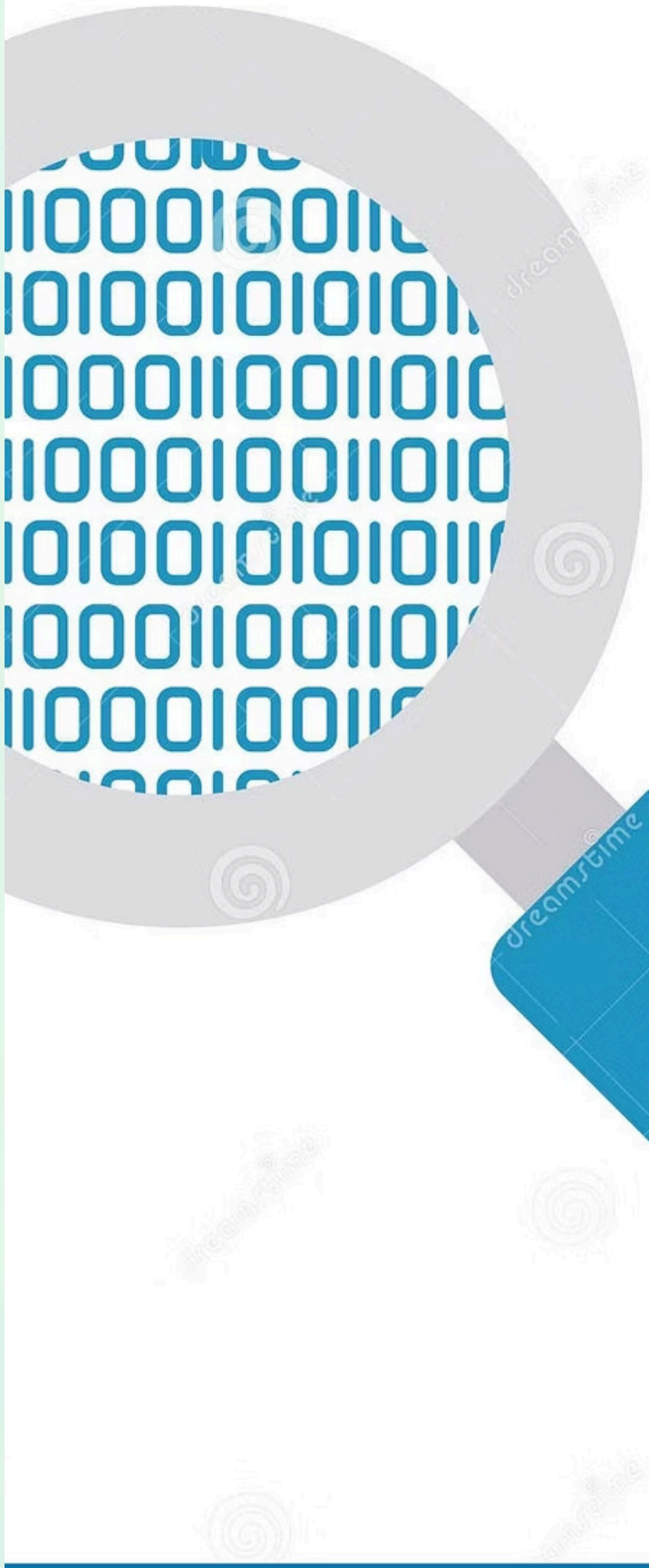
Early Problem Detection

Identifies and fixes bugs early in the development cycle, reducing overall cost.

4

Integration Approaches

- **Top-down:** Start from main modules and progressively integrate smaller sub-modules.
- **Bottom-up:** Begin with testing the smallest components and then integrate them upwards.



Integration Testing: How Parts Work Together

Integration testing checks if different parts of the software work well together. After unit testing, we need to make sure all the separate pieces connect and communicate correctly.

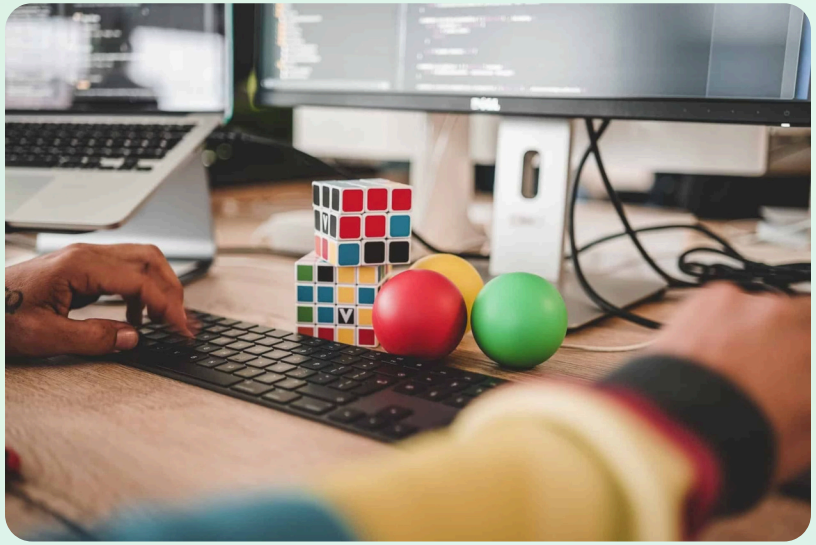
1. Tests how different modules or components interact
2. Checks if data flows correctly between parts
3. Makes sure all parts work as one complete system
4. Happens after unit testing is done

System Testing: Testing the Whole Program

System testing checks the complete, finished software program from start to end. It tests the entire system as a whole to make sure everything works together correctly.

- Tests the whole program, not just individual parts
- Checks if it does what it should do
- Checks if it works well, is fast, and is safe
- Tests real-world scenarios and user workflows
- Happens after integration testing

User Acceptance Testing: Is It Good Enough?



User Testing is the last step to check software. Real users try it out. They make sure the system does what they need for work. They check if it helps with their daily jobs.

- The people who will use the software, or the customer, do this test.
- It makes sure the software is right and ready to be used.
- This step is key to get important people to trust the software before it's given out.

Non-Functional Testing: How Well Does It Work?

Checks how well the system works, not what it does. Focuses on performance, stability, and user experience.

Performance Testing

Checks how fast the app responds and processes data.

Usability Testing

Evaluates how easy and enjoyable the software is to use.

Compatibility Testing

Verifies the app works on different devices, browsers, and operating systems.

Performance Testing: How Fast and Strong Is It?

Performance testing checks how fast the software responds and how well it handles work. It makes sure the app runs quickly and doesn't slow down when many people use it at the same time.

Load Testing

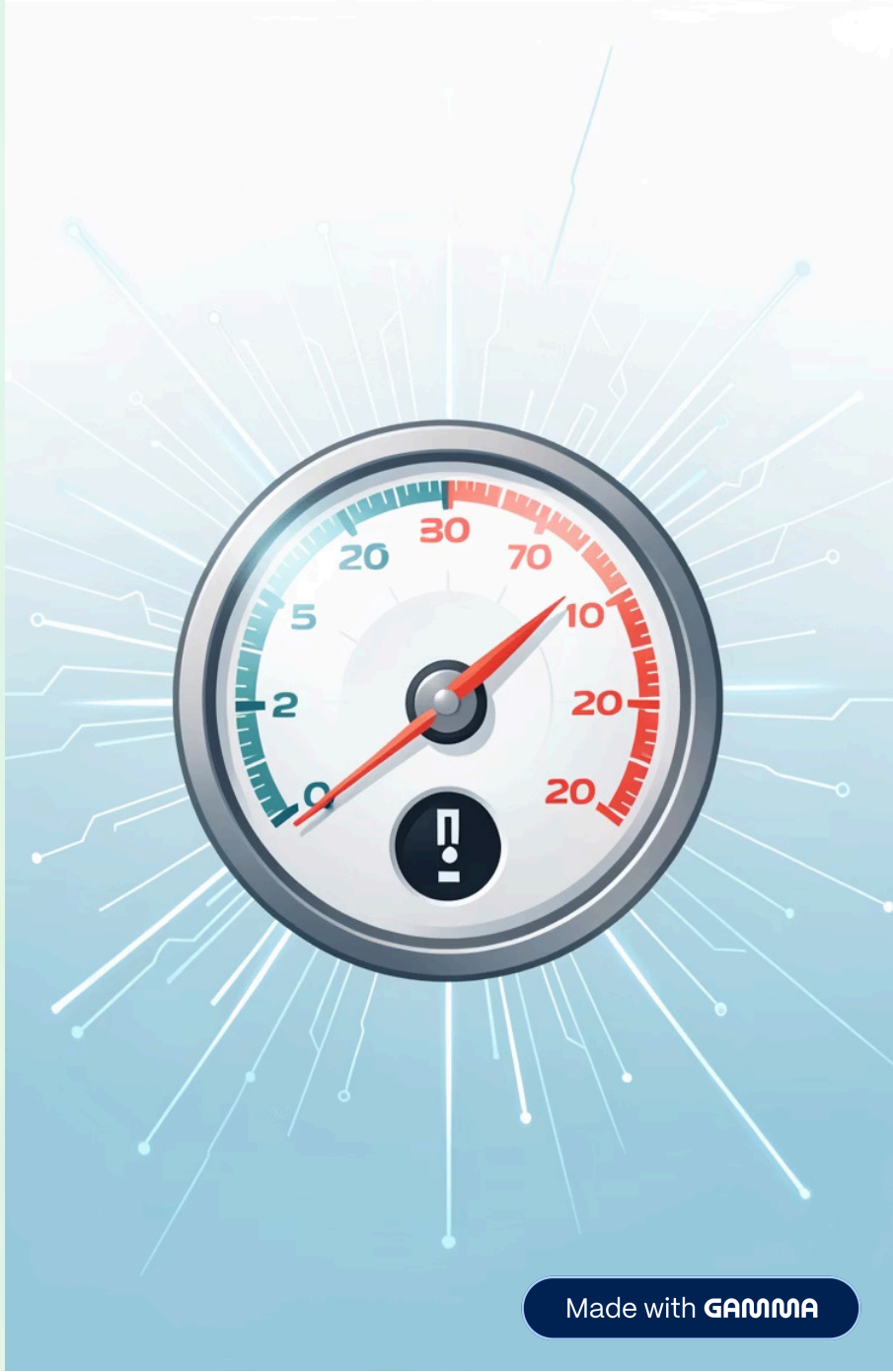
Makes sure the system works well when many users use it at the same time

Stress Testing

Checks if the system stays stable when it gets too much work or when something breaks

Scalability Testing

Tests if the system can grow and handle more users and data as the business grows



Usability Testing: Is It Easy to Use?

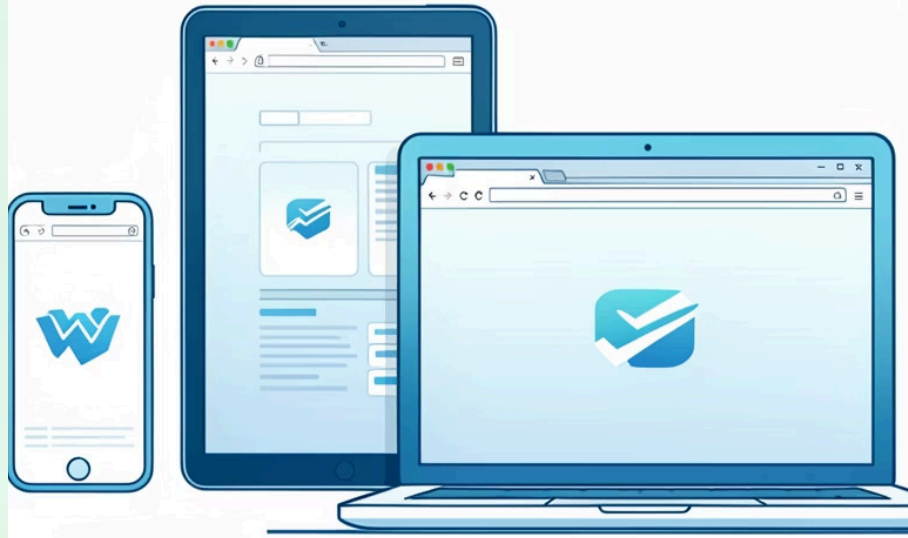
Usability testing checks how easy and enjoyable the software is to use. It makes sure the app has a clear design, easy buttons, and a smooth experience for people who use it.

- Tests if the user interface is clear and easy to understand
- Checks if buttons and menus are easy to find
- Makes sure users can do their tasks quickly and without confusion
- Focuses on user experience and satisfaction
- Often involves real users trying the software

Compatibility Testing: Does It Work Everywhere?

Compatibility testing checks if the software works correctly on different devices, web browsers, and operating systems. It makes sure everyone can use the app no matter what device or browser they have.

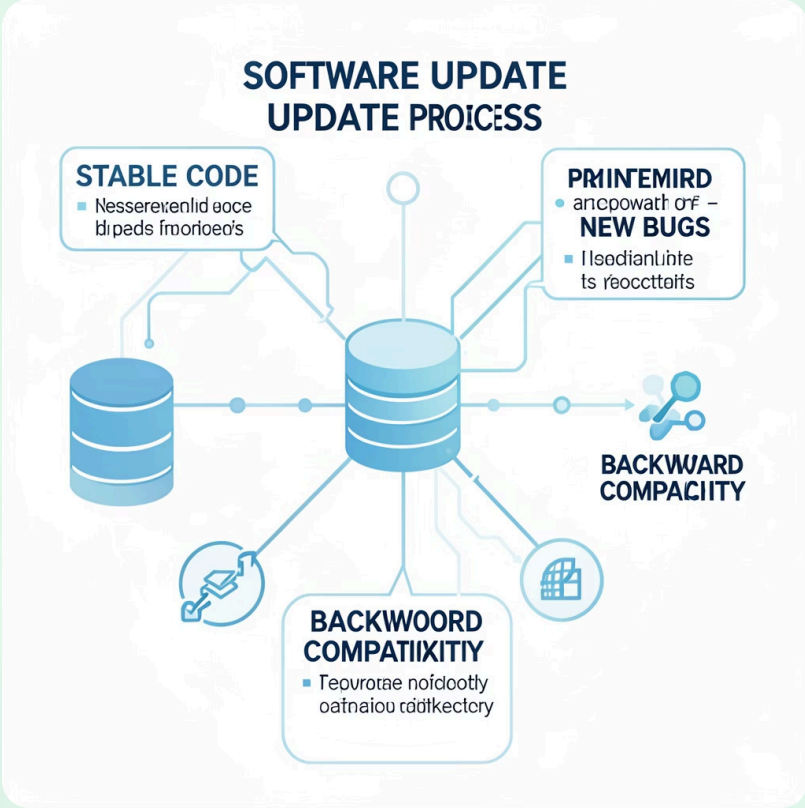
- Tests on different devices: phones, tablets, computers
- Tests on different browsers: Chrome, Firefox, Safari, Edge
- Tests on different operating systems: Windows, Mac, Linux, iOS, Android
- Makes sure the app looks good and works the same everywhere
- Important for apps that many different people use



Regression Testing: Making Sure Things Still Work

Regression testing makes sure new changes don't break old parts of the system or cause new problems.

- We do it after we change things, fix problems, or make things better.
- It checks if the old features still work correctly.
- This helps keep the software good and stops new, unwanted issues.



Quick Software Check

Stlabble App softworbaüld

Smoke Testing: A Fast Check

Main Things

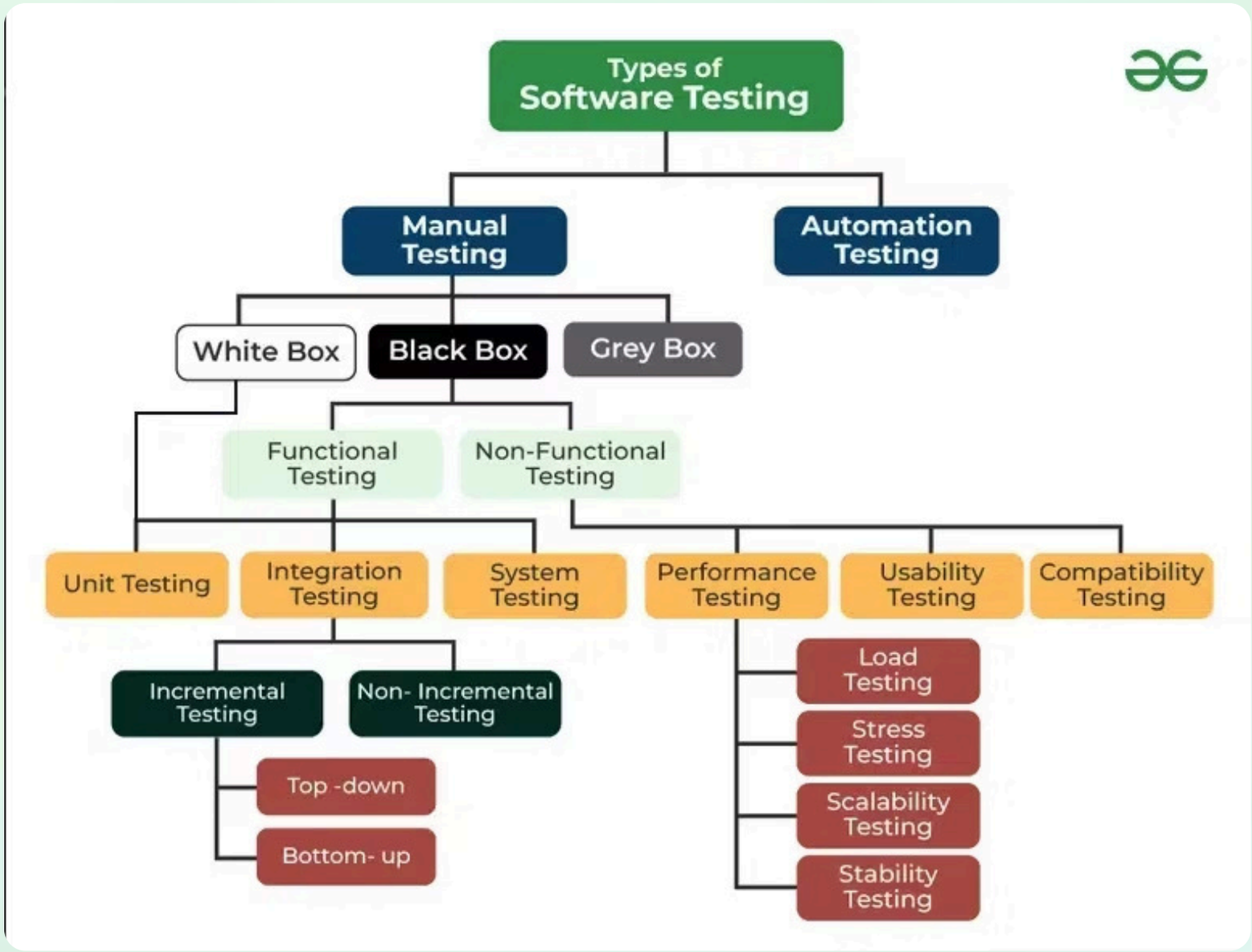
A fast check to make sure the most important parts of the program are working.

App Ready

Makes sure the main program works well enough for bigger tests to begin.

Key Actions

Checks main parts like logging in, moving around, and seeing the main pages.



Fun Time: What We Learned Today

Let's Try It!

Let's use what we learned. With a simple app, find:

- **Three checks** that show if parts work well.
- **Three checks** that show if the app is good.

Main Ideas

Today, we learned about different ways to check computer programs:

- We learned about **basic checks** (for small parts, big parts, or the whole program).
- We learned about **goodness checks** (how fast, how safe, how easy to use, if it works everywhere).
- We saw the difference between **checking again** (to make sure nothing broke) and **quick checks** (a first look at new software).

Questions & Discussion