

Manual Testing Basics, Jira and Sprints

Read this once from top to bottom. Every word is explained in simple language with an example.

1. What is software testing?

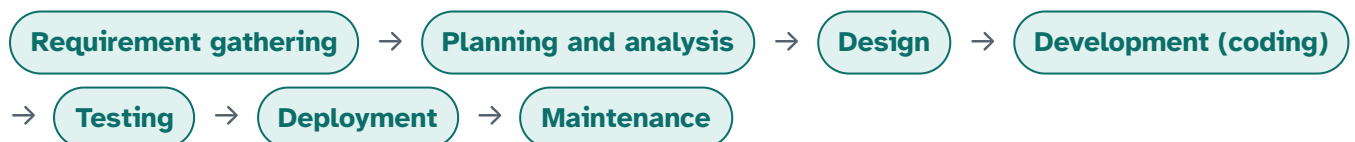
Testing means **using an app like a user and checking that it works the way it should**. If it does not, we report the problem (called a **bug**) so developers can fix it before real users face it.

Manual testing means a person does the checking by hand: opening the app, clicking, typing, and comparing what happens with what was expected. No code is needed.

Simple example: a login page. You type a correct email and password and click Login. You **expect** the Home page to open. If an error comes instead, that is a bug.

2. SDLC: how software is built

SDLC means Software Development Life Cycle. It is the set of steps a team follows to take software from an idea to a live product. Testing is one step, but a good tester is involved in every step.



PHASE	WHAT HAPPENS	TESTER'S PART
Requirement gathering	Business explains what is needed. Written as documents or user stories.	Read it, find unclear or missing points, ask questions.
Planning and analysis	Team decides scope, time, cost, people, risks.	Prepare the test plan and estimate effort.
Design	Screens (UI) and system structure are designed.	Review the design, start writing test scenarios.

Development	Developers write the code.	Write test cases, prepare test data and environment.
Testing	The built app is tested and bugs are fixed.	Execute tests, report and retest bugs.
Deployment	App goes live for users (release).	Smoke test on production after release.
Maintenance	Fixing issues and adding new features after release.	Regression testing for every change.

SDLC vs STLC: SDLC is the whole journey of building software. STLC is only the testing journey inside it.

SDLC models (ways of following the steps)

MODEL	HOW IT WORKS	GOOD / BAD
Waterfall	One phase after another, no going back. Testing starts only after coding is fully finished.	Simple and clear. But bugs are found late and changes are hard.
V-Model	Every development phase has a matching testing phase, planned at the same time.	Testing starts early. But still rigid like Waterfall.
Iterative / Incremental	Build the product in small parts, one after another, improving each time.	Early working parts. Needs good planning of parts.
Spiral	Repeated cycles, each one with risk analysis.	Good for big risky projects. Costly and complex.
Agile	Short sprints (1 to 4 weeks). Design, code and test together. Feedback every sprint.	Flexible, fast feedback. Needs strong teamwork. Most companies use this today.

V-Model: which testing matches which phase

DEVELOPMENT SIDE	TESTING SIDE
Requirement analysis	Acceptance testing (UAT)

System design	System testing
Architecture design	Integration testing
Module design and coding	Unit testing

3. Important words you must know

TERM	MEANING IN SIMPLE WORDS
Bug / Defect	Something that does not work as expected.
Requirement	What the app should do, as written by the business or product team.
Test scenario	One-line idea of what to test. Example: "Check login".
Test case	Step-by-step instructions with expected result. Example: enter valid email, enter valid password, click Login, home page opens.
Test data	The values you type in while testing (emails, passwords, amounts).
Test suite	A group of related test cases, such as all login test cases.
Test plan	A document saying what to test, who tests, when, and with which tools.
Build	A new version of the app given to testers.
Environment	The place where the app runs: Dev (developers), QA/Test (testers), Staging (final check, like real), Production (real users).
Expected / Actual	Expected = what should happen. Actual = what really happened. A difference means a bug.
Verification	"Are we building the product right?" Checking documents and design, without running the app.
Validation	"Are we building the right product?" Running the app to see that it meets the user's need.
QA vs QC	QA (Quality Assurance) is about the process, to prevent bugs. QC (Quality Control) is about checking the product, to find bugs. Testing is part of QC.

4. Types of testing

TYPE	WHAT IT MEANS
Smoke testing	Quick check of the main features right after a new build. If basic things fail, we reject the build. Like checking whether a machine switches on.
Sanity testing	Small, focused check after a bug fix or small change, to see that the change works.
Regression testing	Checking that old working features still work after new changes. This is the work that is later automated.
Retesting	Running the same failed test again after the developer fixes the bug.
Functional testing	Checks what the app does (login, search, payment).
Non-functional testing	Checks how well it does it: speed (performance), safety (security), ease of use (usability), many devices (compatibility).
Exploratory testing	Testing without fixed steps, using your own thinking and curiosity to find bugs.
UAT	User Acceptance Testing. The client or business user checks the app before go-live.
Black box / White box	Black box: you test without seeing the code (testers do this). White box: you test knowing the code (developers do this).

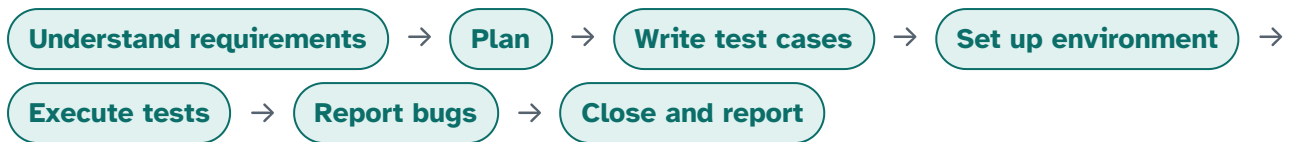
Levels of testing (small to big)



- **Unit:** developer tests one small function.
- **Integration:** checks that two parts work together, for example the app and the database.
- **System:** tester checks the whole app end to end.
- **UAT:** business user approves it.

5. STLC: steps a tester follows

STLC means Software Testing Life Cycle.



6. Writing a test case

FIELD	EXAMPLE
Test case ID	TC_LOGIN_01
Title	Login with valid email and password
Precondition	User account already exists
Steps	1. Open login page. 2. Enter valid email. 3. Enter valid password. 4. Click Login.
Expected result	User lands on the Home page
Actual result	Filled while testing
Status	Pass or Fail

Good test cases cover **positive** (valid input), **negative** (wrong input) and **edge** (limits, like empty or very long text) situations.

7. Bug report and bug life cycle

A good bug report lets the developer repeat the problem without asking you anything. Include:

- **Title:** short and clear, e.g. "Login button does nothing when password has a space".
- **Steps to reproduce:** exact steps.
- **Expected and actual result.**
- **Environment:** browser, device, build number.
- **Severity and priority.**
- **Attachment:** screenshot or video.

Severity vs priority

MEANING	EXAMPLE

Severity	How badly the bug hurts the app (decided by tester).	App crashes on payment: High.
Priority	How soon it must be fixed (decided by business).	Company logo spelled wrong on home page: Low severity, High priority.

Bug life cycle



If the bug still happens after the fix, status becomes **Reopened**. If it is not really a bug, it can be **Rejected**. If it will be fixed later, it is **Deferred**.

8. More types of testing

TYPE	WHAT IT CHECKS
Performance	Speed and stability. Load : expected number of users. Stress : far more than expected, to see when it breaks. Spike : sudden jump of users. Endurance : long running time. Tool: JMeter.
Security	Can data be stolen or misused? Login rules, SQL injection, access control.
Usability	Is it easy and clear for a normal user?
Compatibility	Works on different browsers, devices, screen sizes and operating systems (cross-browser testing).
Accessibility	Usable by people with disabilities, for example screen reader support and enough colour contrast.
Localization	Correct language, date, currency and number format for each country.
Database testing	Data saved in the database is correct and complete.
Ad-hoc	Random testing without planning or documents.
Monkey testing	Random clicks and inputs to check the app does not crash.
Alpha / Beta	Alpha: tested inside the company by internal users. Beta: given to a small group of real users outside before the final release.

Static vs dynamic	Static: reviewing documents and code without running the app. Dynamic: running the app and testing it.
End-to-end	Tests a full user journey across all parts, for example: sign up, search, add to cart, pay.

9. Test documents and measuring quality

Entry and exit criteria

- **Entry criteria:** conditions needed to **start** testing. Example: requirements are clear, build is deployed, test cases are ready, smoke test passed.
- **Exit criteria:** conditions to **stop** testing. Example: all planned tests run, no open critical or high bugs, pass percentage above the agreed number.

Requirement Traceability Matrix (RTM)

A table that links each requirement to its test cases and bugs. It proves that **every requirement is tested** and nothing is missed.

REQUIREMENT	TEST CASES	STATUS	BUG
REQ-01 User can log in	TC_LOGIN_01 to 08	Passed	None
REQ-02 User can reset password	TC_PWD_01 to 06	1 failed	BUG-45

Test environment and test data

The test environment should be as close to real (production) as possible. Test data should be realistic but **never real customer data**. Prepare it before starting.

Test closure and reports

At the end the tester shares a summary: how many tests were planned, run, passed, failed, which bugs are still open, and whether the build is safe to release.

Common measures (metrics)

METRIC	MEANING
Test case pass %	Passed tests divided by executed tests, times 100.

Defect density	Number of bugs per size of the module (for example per 1000 lines of code).
Defect leakage	Bugs that testers missed and users found after release. Lower is better.
Test coverage	How much of the requirements or code is covered by tests.
Defect age	How many days a bug stays open.

Risk-based testing

When time is short, test the parts that are **most used** and **most damaging if they fail** first (payment, login), and the less important parts later.

10. Agile, Scrum and sprint

Agile means building software in small pieces, with frequent feedback, instead of one big delivery at the end. **Scrum** is the most popular way to do Agile.

WORD	MEANING
Sprint	A fixed work period, usually 2 weeks. The team builds and tests a few features in it.
Product backlog	The full list of work wanted for the product.
Sprint backlog	The work the team chose for the current sprint.
User story	A feature written from the user's view: "As a user, I want to reset my password so that I can log in again."
Acceptance criteria	Conditions that must be true for the story to be called done. Testers write test cases from these.
Story points	A number showing how big or hard a story is.
Sprint planning	Meeting at the start: decide what to do in this sprint.
Daily stand-up	15-minute daily meeting: what I did yesterday, what I will do today, any blockers.
Sprint review / demo	At the end: show finished work to the stakeholders.
Retrospective	Team talks about what went well and what to improve.

Definition of Done

Team's checklist for a story to be fully complete, including tested.

Roles: Product Owner (decides what to build), Scrum Master (helps the team follow the process), Developers, and QA/Testers.

11. Jira: tickets, board and sprint

Jira is a tool used by teams to track work. Every piece of work is a **ticket** (also called an issue) with a key like `PAY-123`.

Ticket types

TYPE	USE
Epic	A big feature made of many stories. Example: "User Login".
Story	One small feature for the user.
Task	A technical or general work item.
Bug	A defect found in testing. Testers create these.
Sub-task	A small part of a story, e.g. "Write test cases".

Important fields in a ticket

Summary (title), Description, Assignee (who works on it), Reporter (who created it), Priority, Status, Sprint, Labels, Story points, Attachments, Comments.

Jira board

A board shows tickets as cards in columns. You drag a card to the next column as work moves ahead.



A **Scrum board** shows only the current sprint. A **Kanban board** shows continuous flow without sprints.

A tester's day in Jira

1. Open the story in the sprint and read the description and acceptance criteria.
2. Write test cases (often as sub-tasks or in a test management tool).
3. When the story reaches "In QA", test it on the QA environment.

4. If something fails, create a **Bug** ticket, link it to the story, and assign it to the developer.
5. After the fix, retest. Pass: close the bug and move the story to Done. Fail: reopen with a comment.

More Jira things asked in interviews

- **Workflow:** the path a ticket follows (To Do → In Progress → In QA → Done). Moving a ticket from one status to another is called a **transition**.
- **Linking tickets:** link a bug to its story with "relates to", "is blocked by" or "duplicates", so everyone can trace it.
- **Components and labels:** tags to group tickets by part of the app (Payments, Login) or by topic (regression, UI).
- **Affects Version / Fix Version:** in which release the bug was found, and in which release it will be fixed.
- **Dashboard and filters:** saved searches and charts, such as "all open bugs assigned to me".
- **Backlog:** the list of tickets not yet taken into a sprint. **Burndown chart** shows work left in the sprint.
- **Test management add-ons:** Xray and Zephyr let you write and run test cases inside Jira. **Confluence** is used for documents.

JQL: Jira's search language

```
project = PAY AND issuetype = Bug AND status ≠ Done  
assignee = currentUser() ORDER BY priority DESC  
sprint in openSprints() AND issuetype = Bug
```

These find: all open bugs in project PAY; tickets assigned to me, highest priority first; bugs in the current sprint.

12. How to answer common fresher questions

- **Tell me about yourself:** your degree, the skills you learned (manual testing, Selenium, API, Git), one project or practice work, and why you want a QA role. Keep it under one minute.
- **Why testing?** "I like finding problems before users do, I pay attention to details, and QA is a strong start for a career in quality and automation."

- **Explain a project:** pick one app you practised on. Say what it does, what you tested (login, search, forms), how many test cases you wrote, which bugs you found and how you reported them.
 - **If you do not know an answer:** say "I have not worked on this yet, but I understand it as..." and explain what you do know. Never guess wildly.
 - **Practise live:** do a demo test on a public practice website. Write test cases, find a bug, and write the bug report in the standard format.
-

13. Quick interview answers

- **Smoke vs sanity?** Smoke is a broad, shallow check of a new build. Sanity is a narrow, deep check after a small change or fix.
- **Regression vs retesting?** Retesting checks one fixed bug again. Regression checks that old features are not broken.
- **Severity vs priority?** Severity is the impact on the app. Priority is how soon to fix.
- **What if a developer says "not a bug"?** Show the requirement, steps and proof (screenshot or video), and discuss politely with the product owner.
- **What is a test case vs a test scenario?** Scenario is the idea ("test login"). Test case is the detailed steps for it.
- **What do you do if you find a bug?** Repeat it, collect proof, write a clear bug report in Jira, assign it, retest after the fix.
- **What is a sprint?** A fixed period (about 2 weeks) in which the team completes a set of stories.
- **What is SDLC? Name some models.** The steps to build software: requirements, design, coding, testing, deployment, maintenance. Models: Waterfall, V-Model, Iterative, Spiral, Agile.
- **Waterfall vs Agile?** Waterfall does phases one after another and tests at the end. Agile works in short sprints and tests in every sprint.
- **SDLC vs STLC?** SDLC covers building the whole software. STLC covers only the testing steps.
- **What is RTM?** A table that maps requirements to test cases so no requirement is left untested.
- **Entry vs exit criteria?** Conditions to start testing vs conditions to stop testing.

- **Load vs stress testing?** Load: expected users. Stress: more than expected, to find the breaking point.
- **Alpha vs beta testing?** Alpha: inside the company. Beta: with a small group of real users outside.
- **What is defect leakage?** Bugs that were missed in testing and found by users in production.
- **Verification vs validation?** Verification: building the product right (reviews). Validation: building the right product (running tests).
- **What is a test plan and who writes it?** A document of scope, approach, schedule and resources; usually written by the QA lead.