

How a Tester Thinks

Tools can be learned in weeks. The tester's way of thinking is what makes you good at the job.

1. The tester's mindset

- **Be curious.** Keep asking "what if?" What if I leave this blank? What if I click twice? What if the internet drops?
- **Think like the user.** Real users are in a hurry, make mistakes, and use the app in ways nobody planned.
- **Try to break it, politely.** Your job is to find problems before customers do. A found bug is a win for the team, not a fight with the developer.
- **Question everything.** If a requirement is unclear, ask. Many bugs come from unclear requirements.
- **Be detail-oriented.** Spelling, alignment, numbers, messages: notice small things.
- **Communicate clearly.** A bug is useful only if others can understand and repeat it.

2. The 7 principles of testing

PRINCIPLE	MEANING IN SIMPLE WORDS
1. Testing shows that bugs exist	Testing can show bugs are present. It can never prove an app has zero bugs.
2. Exhaustive testing is impossible	You cannot test every input and every path. A single phone number box has crores of possible inputs. So choose tests smartly, by risk.
3. Early testing saves time and money	Start testing from the requirement stage. A bug found in requirements is cheap to fix. The same bug found after release is costly.
4. Defects cluster together	A few modules usually hold most of the bugs (about 80% in 20% of the app). Test risky or complex areas more.
5. Pesticide paradox	If you run the same tests again and again, they stop finding new bugs, like insects becoming immune to the same pesticide. Review and update your tests regularly.

6. Testing depends on context	A banking app, a game and a hospital system need different kinds of testing.
7. Absence-of-errors fallacy	An app with no bugs is still a failure if it does not do what the user needs.

3. Two simple techniques to design test data

Equivalence partitioning

Split inputs into groups that behave the same, then test one value from each group.

Boundary value analysis

Bugs love the edges. Test the values just below, on, and just above each limit.

EXAMPLE	AN AGE FIELD ACCEPTS 18 TO 60
Groups	Below 18 (invalid), 18 to 60 (valid), above 60 (invalid)
One test per group	10, 35, 75
Boundary values	17, 18, 19, 59, 60, 61

Also try: empty field, letters, special characters like @#\$, negative number, very large number, spaces only.

More test design techniques

TECHNIQUE	IDEA	EXAMPLE
Decision table	When the result depends on a combination of conditions, list all combinations and the result of each.	Login: valid email Y/N × valid password Y/N → only Y+Y logs in. Discount: member Y/N × coupon Y/N.
State transition	The app moves between states depending on actions. Test each move and the not-allowed moves.	Order: Placed → Packed → Shipped → Delivered. A delivered order cannot go back to Placed. Account: Active → Locked after 3 wrong passwords.

Error guessing	Use your experience to guess where bugs hide.	Blank input, double click, special characters, very large values, back button.
Exploratory testing	Learn, design and test at the same time, without fixed steps. Set a goal and a time (for example 45 minutes) and note what you find.	"Explore the checkout with different payment methods for 45 minutes."

4. Detailed case studies: how to test a feature

For every feature, think in these buckets. Each case below uses the ones that fit.

BUCKET	ASK YOURSELF
Positive	Does it work when used correctly?
Negative	Does it handle wrong input without breaking?
Edge	What happens at limits and in odd situations?
UI / usability	Is it clear, readable and easy to use?
Security	Can someone misuse it?
Network / performance	What if the internet is slow or drops? What if many users come?
Data	Is the data saved and shown correctly everywhere?

Case 1: Login page

POSITIVE

- Correct email and password opens the Home page.
- Login works with Enter key and with the button.
- "Remember me" keeps the user logged in after closing the browser.

NEGATIVE

- Wrong password, unknown email, both blank, only one blank.
- Error message is clear but does not say which field was wrong (so attackers learn nothing).

- Correct email with wrong case, such as `USER@mail.com` : check what the rule is and that it is consistent.

EDGE

- Spaces before or after email or password.
- Very long email or password (200+ characters), emoji, non-English letters.
- Copy-paste into fields, browser auto-fill, password manager.
- Click the Login button many times quickly: only one login should happen.
- Login in two browsers or two devices at once.
- Session expires after being idle: user should go back to login with a clear message.
- Press browser Back after logout: the account page must not open again.

UI AND SECURITY

- Password is hidden, and the show/hide eye icon works.
- Tab order of fields is correct. The page looks fine on a phone.
- Account locks after several wrong tries.
- SQL injection text like `' OR 1=1 --` in the email box must not log anyone in.
- Password is not visible in the URL, and the page is on HTTPS.

Case 2: Sign-up form

POSITIVE

- All valid details create the account. A welcome email or OTP arrives.
- The new user can log in at once.

NEGATIVE

- Each mandatory field left blank, one at a time. The message must point to that field.
- Invalid email formats: `abc` , `abc@` , `abc@mail` , `a b@mail.com` .
- Email already registered. Same email in different upper/lower case.
- Password and confirm password do not match.

EDGE

- Name with a dot, hyphen, apostrophe (O'Neil), or in Hindi or Tamil.
- Phone number: 9 digits, 10 digits, 11 digits, letters, +91 prefix, spaces.
- Password rules at the exact limit: minimum length minus 1, equal, plus 1. Missing capital, number or symbol.

- Date of birth: today, future date, under-age (17 years 364 days vs 18 years), 29 Feb on a leap and non-leap year.
- Submit twice quickly. Refresh after submit. Browser Back after submit. Only one account must be created.
- Terms checkbox not ticked: sign-up should be blocked.

DATA AND SECURITY

- Details saved in the database exactly as entered (no cut-off, no wrong characters).
- Password is stored in encrypted form, never shown in emails or logs.
- Script text like `<script>alert(1)</script>` in the name field must not run.

Case 3: Search box on a shopping site

POSITIVE

- "laptop" shows laptops. Results are relevant and the result count is right.
- Pressing Enter and clicking the search icon give the same result.
- Suggestions appear while typing, and clicking one searches it.

NEGATIVE

- No matching word: a friendly "no results" message and maybe suggestions. No blank page or error.
- Empty search, or only spaces.

EDGE

- CAPITAL letters, mixed case, extra spaces between words, spelling mistakes ("laptop").
- Singular and plural ("shoe" vs "shoes"). Short and very long text (500 characters).
- Special characters: `% _ ' " & # @`. Numbers only. Emoji. A word in Hindi.
- Multiple words in different order ("red shirt" vs "shirt red").
- Exact product code or model number.
- Search, open a product, press Back: previous results and filters should still be there.

FILTERS, SORTING, PAGES

- Price filter: min greater than max, min equal to max, negative, blank.
- Sort low-to-high: check the order really is correct, including equal prices.
- Pagination: first page, last page, a page number beyond the last, results exactly equal to one page.
- Combine several filters, then clear all filters.

PERFORMANCE AND SECURITY

- Results come in about 2 seconds. Typing fast does not freeze the page.
- Script or SQL text in the box must not run, and the page must not break.

Case 4: Add to cart and checkout

POSITIVE

- Item is added. Cart count, item price and total update correctly.
- Increase or decrease quantity updates the total. Remove item works.
- Discount code reduces the correct amount. Tax and delivery charges are right.

NEGATIVE

- Out-of-stock item cannot be added. Quantity 0, negative, decimal (1.5) or letters rejected.
- Expired, wrong or already-used discount code shows a clear message.
- Delivery pin code where delivery is not possible.

EDGE

- Add the same item twice: quantity should become 2, not two separate lines (or follow the requirement).
- Maximum allowed quantity and one more than that. Only 1 item left in stock and two users try to buy it together.
- Price or stock changes while the item is sitting in the cart.
- Discount applies, then an item is removed so the minimum order value is no longer met.
- Total with many items: check rounding to 2 decimals (for example 3 items at 33.33).
- Log out and log in again: cart should still be there. Add as guest, then log in: cart should merge.
- Same cart open on phone and laptop: change on one, check the other.
- Empty cart: checkout button should be disabled or give a message.

UI AND DATA

- Product image, name and price in the cart match the product page.
- Currency symbol and number format are consistent (₹1,299.00).
- Order summary matches the cart exactly.

Case 5: Online payment (UPI or card)

POSITIVE

- Payment succeeds, the order is created once, and confirmation page, SMS or email are sent.
- Amount charged is the same as the amount shown on the order summary.
- Each method works: UPI, card, net banking, wallet, cash on delivery.

NEGATIVE

- Wrong card number, wrong CVV, expired card, wrong UPI PIN, wrong OTP, insufficient balance.
- Card type that the site does not support. Blank fields.
- After failure the user can retry or pick another method, and no order is wrongly marked as paid.

EDGE: THE DANGEROUS ONES

- Double click on Pay: money must be taken only once.
- Internet drops after money is deducted but before the confirmation page loads. The order status must still become correct.
- User presses Back or Refresh on the payment page, or closes the browser mid-payment.
- Bank is slow and the payment stays "pending" for several minutes. The user should see a clear status.
- Payment page times out. Same order paid from two tabs.
- Amount with paise (₹499.50) and a very large amount above the bank limit.
- Coupon or wallet balance used together with a card (partial payment).

AFTER PAYMENT

- Failed payment must not reduce stock or create an order.
- Cancel the order: refund amount, refund time and message are correct.
- Invoice has the right name, address, tax and amount.

SECURITY

- Card number is masked after entry (shown as **** 1234). The page uses HTTPS.
- The amount cannot be changed from the browser or URL before paying.

Case 6: Forgot password with OTP

POSITIVE

- Enter registered email or phone: OTP arrives within a reasonable time.
- Correct OTP opens the "new password" page. The new password works to log in.

NEGATIVE

- Unregistered email or phone. Blank field. Wrong OTP. Blank OTP. OTP with letters.
- OTP of 5 or 7 digits when 6 are needed.
- Wrong OTP entered several times: the user is blocked for some time.

EDGE

- OTP used after it expires (for example 5 minutes), one second before and one second after.
- OTP used a second time after success. Must fail.
- Click "Resend OTP" quickly many times: there should be a wait timer or a limit, and only the newest OTP must work.
- OTP received in a different phone or email than the one entered.
- New password equal to old password, or breaking the password rules. New and confirm password mismatch.
- Reset link opened twice, or opened after the password was already changed.
- Two reset requests from two tabs at the same time.

AFTER RESET AND SECURITY

- Old password stops working. Other logged-in sessions are logged out (if the requirement says so).
- A confirmation message or email is sent that the password was changed.
- The OTP is not shown in the page, URL or API response.
- Message is the same for registered and unregistered emails (so nobody can find who has an account).

Case 7: File upload (profile photo or resume)

POSITIVE

- Allowed types (jpg, png, pdf) upload and show or open correctly.
- Both "Browse" button and drag-and-drop work.
- Uploaded file can be viewed, downloaded, replaced and deleted.

NEGATIVE

- Not-allowed type such as .exe, .zip or .js. Empty (0 KB) file. Corrupt file that will not open.

- Click Upload with no file selected.
- A virus test file, if the site scans uploads.

EDGE

- File size: exactly at the limit, 1 KB under, 1 KB over. For example 5 MB limit: test 4.99, 5.00 and 5.01 MB.
- Extension changed on purpose, for example an .exe renamed to .jpg. It must be rejected.
- Capital extension (.JPG), double extension (photo.jpg.exe), no extension.
- Very long file name, spaces, Hindi letters or special characters in the name.
- Same file uploaded twice, or two different files with the same name.
- Image with huge dimensions or a very small one. Rotated phone photo shows the right way up.
- Multiple files at once: the maximum count and one more. One bad file among good ones.
- Upload cancelled halfway, internet drops during upload, browser refreshed during upload.

UI AND SECURITY

- Progress bar moves and shows a final state. Error messages are clear: "File too large (max 5 MB)".
- Uploaded files cannot be opened by another user through the link, unless allowed.
- The file is not run as code on the server.

5. Classic "how will you test..." questions

Interviewers often ask these to see how you think. Do not rush to answer. First **ask questions**, then list what you would test in groups.

Test a pen

- **First ask:** what type of pen, who will use it, what is it for?
- **Functional:** it writes, ink flows smoothly, cap fits, click works (for a click pen), refill can be changed.
- **Usability:** comfortable grip, good weight, writes on different paper, left-hand use.

- **Edge and stress:** writes upside down or at an angle, after being dropped, in hot or cold weather, after leaving the cap off for hours.
- **Other:** ink does not leak or smudge, colour matches the label, safe material, how many pages it writes (durability).

Test a lift (elevator)

- **Functional:** pressing a floor button takes the lift to that floor, doors open and close, up and down buttons from each floor work, the display shows the right floor.
- **Negative and edge:** pressing many buttons together, pressing the current floor, door-open button during moving, holding the door button.
- **Safety:** weight limit alarm, door sensor stops closing when someone is in the way, emergency button, alarm and light work, power cut behaviour, fire mode.
- **Performance:** waiting time at peak hours, speed, smooth stop.

Test an ATM

- **Positive:** insert the card, correct PIN, withdraw an allowed amount, correct cash and receipt, balance reduces by the same amount.
- **Negative:** wrong PIN (card blocks after 3 tries), expired or damaged card, amount more than the balance, more than the daily limit, amount not in multiples of 100, ATM has no cash.
- **Edge:** power or network goes off in the middle of a withdrawal (money must not be cut without cash), card not taken back, cancel at any step, session timeout, two transactions together.
- **Security:** PIN hidden while typing, card is not kept without the user, screen resets after finishing.

Test a chat app message (like WhatsApp)

- **Positive:** send and receive a text, delivered and read ticks change, time shown correctly.
- **Edge:** empty message, very long text, emoji, links, other languages, sending while offline (message sends once the network is back), sending to a blocked user.
- **Other:** notification arrives, works on different phones and OS versions, large images and videos, message order is right, group chat with many members, battery and data

usage.

Test a web page with a "Contact us" form

- **Positive:** valid name, email, message submit and a thank-you message appears. The team receives the details.
- **Negative:** blank required fields, wrong email format, very long message, special characters and script text.
- **Other:** captcha works, double submit sends only one message, form looks right on mobile and in all browsers, tab key order, error messages are clear.

Answer pattern to remember: ask questions → functional (positive, negative, edge) → usability → performance → security → compatibility → other (safety, data, accessibility).

6. A simple checklist for any feature

1. **Understand** what the feature should do. Ask if it is not clear.
2. **List** the inputs, outputs and steps.
3. **Test the happy path** first (everything correct).
4. **Then negative and edge cases** using the techniques above.
5. **Think about the surroundings:** different browsers and devices, slow network, many users, data stored correctly, security.
6. **Report** clearly and **retest** after the fix.
7. **Ask yourself:** "If I were the customer, would I be happy?"

7. When to stop testing and what to do with limited time

You can never test everything (principle 2). Testing stops when the exit criteria are met, for example:

- All planned test cases are executed and the pass rate is at the agreed level.
- No open critical or high-severity bugs.
- Test coverage target is reached, and the number of new bugs found each day has become very low.

- Time or budget is over. Then tell the team clearly **what was tested, what was not, and the risk**.

If time is short: run smoke tests, then high-risk and most-used features, then regression on key areas. Leave rare, low-impact areas last.

If requirements are missing: ask the product owner or developer, read similar features, use the old version of the app as a guide, test as the user would, and write down the assumptions you made.

8. Working with developers and the team

- Report **facts, not blame**: "Login fails when the password has a space", not "Your code is wrong".
- Give steps, screenshot or video, build number, browser, and test data. A bug that cannot be repeated is ignored.
- Check **not-reproducible** bugs again on the same build and environment. Share the exact data.
- If a developer says "works on my machine": compare the environment, data, browser and build version.
- If you disagree whether it is a bug, show the requirement or acceptance criteria and ask the product owner to decide.
- Before releasing, give an honest status: what passed, what failed, what is left.
- Ask when you are stuck, and write down what you learned.

What makes a good bug report

- Clear title: what is wrong and where.
- Exact steps, expected result, actual result.
- One bug per report. Do not mix many issues.
- Environment and build number. Proof attached. Severity and priority set.
- No opinions or harsh words.

9. Quick interview answers

- **Can you test everything?** No. We use risk, equivalence partitioning and boundary values to choose the most useful tests.

- **When should testing start?** As early as possible, from the requirement stage.
- **What is the pesticide paradox?** Repeating the same tests stops finding new bugs, so tests must be reviewed and updated.
- **How do you test a feature you know nothing about?** Ask questions, read the requirement, explore the feature as a user, then write test cases for positive, negative and edge cases.
- **Boundary value for 1 to 100?** 0, 1, 2, 99, 100, 101.
- **What if you have little time?** Test the highest-risk and most-used features first, and tell the team what was not tested.
- **What makes a good tester?** Curiosity, attention to detail, clear communication, and thinking like the user.
- **Smoke, sanity and regression: which first?** Smoke on the new build first. If it passes, run the detailed tests, then regression to see old features still work.
- **How will you test a lift or a pen?** Ask questions first, then cover functional, usability, performance, security and safety, with positive, negative and edge cases.
- **What is a decision table?** A table of all combinations of conditions with the expected result of each.
- **What is exploratory testing?** Learning and testing together without written steps, guided by a goal and by your experience.
- **When do you stop testing?** When exit criteria are met, or when time is over and risks are clearly communicated.
- **What is risk-based testing?** Testing the most important and most likely to fail areas first.
- **A bug is found in production. What do you do?** Reproduce it, raise it with severity and priority, check how it escaped testing, add a test for it, and include it in regression.
- **Developer says it is not a bug. What now?** Show the requirement and proof, stay polite, and let the product owner decide.