INTRUDUCTION

🧪 Software Testing Workflow (Practical Guide)

Welcome to the **Software Testing Workflow Guide** 🚀.
This document explains the **end-to-end process of software testing** in a **simple and practical way** so that you can easily understand and gain **hands-on experience** with testing tools.

🔄 Software Testing Workflow (Step by Step)

1️⃣ Requirement Analysis

- **What happens?**
  Understand *what the software should do* (features, user stories, acceptance criteria).
- **Tools you can use:**
  - Jira, Confluence, Trello (to track requirements)
  - Google Docs/Sheets (simple projects)
- **Hands-on practice:**
  Take any app idea (e.g., **E-commerce App**) and write down requirements like:
  - User can log in
  - User can search products
  - User can add items to cart

2️⃣ Test Planning

- **What happens?**
  Create a roadmap: *What to test, how to test, resources, risks*.
- **Tools you can use:**
  - MS Word, Excel (simple test plan docs)
  - Jira / TestRail (professional planning)
- **Hands-on practice:**
  Write a **Test Plan** for your app → include scope, objectives, roles, schedule.

3️⃣ Test Case Design

- **What happens?**
  Write step-by-step **Test Cases** to check if features work.
- **Tools you can use:**
  - Excel / Google Sheets (basic)
  - TestRail, Zephyr (advanced)
- **Hands-on practice:**
  Write test cases for login:
  - ✅ Valid username + password → Login successful
  - ❌ Invalid password → Error message

4️⃣ Test Environment Setup

- **What happens?**
  Prepare the **system where testing happens** (hardware, software, network).
- **Tools you can use:**
  - Localhost setup (XAMPP, WAMP, Docker)
  - Cloud testing tools (BrowserStack, Sauce Labs)
- **Hands-on practice:**
  Install your app on local system or emulator → confirm all dependencies are working.

5️⃣ Test Execution

- **What happens?**
  Execute the **test cases** → run them step by step.
- **Tools you can use:**
  - Manual: Excel/Google Sheets
  - Automation: Selenium, Postman, PyTest
- **Hands-on practice:**
  Perform login tests manually → Record Pass/Fail status.

6️⃣ Defect Reporting

- **What happens?**
  Log any **bugs/issues** found.
- **Tools you can use:**
  - Jira, Bugzilla, Mantis
  - Excel (for beginners)
- **Hands-on practice:**
  If login fails with correct credentials → Write a bug report in Excel:
  - Bug ID: 001
  - Title: Login fails with valid credentials
  - Steps: Enter valid details → Click Login → Error appears
  - Severity: High

7️⃣ Test Reporting

- **What happens?**
  Share results: How many tests passed, failed, blocked.
- **Tools you can use:**
  - Excel, Google Sheets
  - TestRail, Jira Reports
- **Hands-on practice:**
  Create a simple **Test Summary Report** with total test cases and pass/fail ratio.

8️⃣ Re-testing & Regression Testing

- **What happens?**
  - **Re-testing**: Check if the bug is fixed.
  - **Regression Testing**: Check if the fix broke other features.
- **Tools you can use:**
  - Selenium, PyTest (automation)

    - Postman (API regression)

- **Hands-on practice:**
  Fix the login bug → Re-test login → Also check search/cart still works.

🚪 Test Closure

- **What happens?**
  Wrap up testing → Document learnings, metrics, defects, coverage.
- **Tools you can use:**
    - Word/Excel (basic)
    - Jira Reports (advanced)
- **Hands-on practice:**
  Write a closure note:
    - Total Test Cases: 50
    - Passed: 45 | Failed: 5
    - Lessons learned: Need better requirement clarity

🛠 Tools by Category

Phase	Tools (Beginner)	Tools (Professional)
Requirement Analysis	Google Docs/Sheets	Jira, Confluence
Test Planning	MS Word, Excel	TestRail, Zephyr
Test Case Design	Excel, Google Sheets	TestRail, QTest
Test Execution	Manual (Excel)	Selenium, PyTest, Postman
Bug Tracking	Excel	Jira, Bugzilla, Mantis
Reporting	Excel, Sheets	Jira Reports, TestRail

📌 Example Practical Flow (E-Commerce App)

1. Requirement: *Login, Search, Cart, Checkout*

2. Write test plan in Excel
3. Design test cases for login
4. Execute manually → Pass/Fail
5. Found bug → Report in Excel/Jira
6. Re-test after fix → Regression on search/cart
7. Create final test report

Chapter 1: Test Plan – The Roadmap

🌟 What is a Test Plan?

Imagine you and your friends are planning a school picnic. Before going, you decide:

- Where to go?
- What to take?
- Who is responsible for what?
- What problems may come?

👉 That "picnic plan" is just like a **Test Plan** in software testing.
It's a **document that describes how testing will be done for a software project**.

🎯 Why is it Important?

- Prevents confusion (everyone knows what to test and how).
- Saves time (no last-minute chaos).
- Keeps track of risks (what could go wrong).
- Ensures proper use of people and tools.

🔑 Key Terms (Explained Simply)

- **Objective** → Why are we testing? (e.g., check if login works)
- **Scope** → What we will test & what we won't (login page yes, admin panel no).
- **Strategy/Approach** → How will we test? (manual testing, automation).
- **Resources** → Who will test? (QA, developer, tools).
- **Schedule** → When testing will happen (start & end dates).
- **Risks** → What can go wrong? (server crash, less testers).

📋 Real-Life Example (Online Shopping App)

Let's say we are testing the **Login Page** of an online shopping app:

- **Objective** → Verify if users can log in successfully.

- **Scope** → Test valid/invalid login, forgot password; skip admin login.
- **Strategy** → Manual testing on Chrome & Firefox.
- **Resources** → QA Team (2 testers).
- **Schedule** → 1 week (10 Aug – 17 Aug).
- **Risks** → Server downtime, limited test devices.

📑 Simple Test Plan Template

Objective	Scope	Approach	Roles/Resources	Schedule	Risks
Test Login Page	Functional login testing (valid/invalid, forgot password)	Manual testing on browsers	QA Team (2 testers)	10 Aug – 17 Aug	Server downtime, fewer devices

---

🧠 Easy Memory Trick (Mnemonic)

👉 **OSARRS** → **O**bjective, **S**cope, **A**pproach, **R**oles, **R**isks, **S**chedule

Think of it like:
"Our School Always Runs Really Smooth" 🎒✨

Chapter 2: RTM (Requirement Traceability Matrix) – The Checklist

---

🌟 What is RTM?

Imagine your mom gives you a **shopping list**:

- Buy Milk
- Buy Bread
- Buy Apples

When you return, she checks if you got everything.
👉 That checking process is like **RTM (Requirement Traceability Matrix)** in software testing.

It's a **document that ensures every requirement is linked to at least one test case** (so nothing is missed).

🎯 Why is RTM Important?

- Ensures no requirement is left untested.
- Shows which test case verifies which requirement.
- Helps identify gaps (requirement without a test).
- Useful for clients to see coverage.

🔑 Key Terms (Explained Simply)

- **Requirement ID** → Unique ID of requirement (e.g., REQ-001: Login).
- **Requirement Description** → What needs to be built (User should log in with valid credentials).
- **Test Case ID** → The test(s) linked to this requirement.
- **Status** → Tested or not tested, Pass/Fail.

📋 Real-Life Example (Online Shopping App – Login Page)

- **Requirement (REQ-001)** → User should be able to log in with valid credentials.
- **Test Case (TC-101)** → Enter correct username/password → Successful login.

- **Status** → Pass
- **Requirement (REQ-002)** → User should see an error with wrong credentials.
- **Test Case (TC-102)** → Enter wrong password → Error message displayed.
- **Status** → Pass

📑 Simple RTM Template

Requirement ID	Description	Test Case ID(s)	Status
REQ-001	Login with valid credentials	TC-101	Pass
REQ-002	Show error for wrong credentials	TC-102	Pass
REQ-003	Forgot Password should send reset link	TC-103	Fail

🧠 Easy Memory Trick (Mnemonic)

👉 **RTS** → **R**equirement, **T**est Case, **S**tatus

Think of it like:
"Remember To Shop" 🛒

Chapter 3: Test Case – The Recipe

🌟 What is a Test Case?

Imagine you are cooking noodles. You follow steps like:

1. Boil water.
2. Add noodles.
3. Put spices.
4. Taste it.

👉 That step-by-step recipe is like a **Test Case** in software testing.
It is a **document that tells exactly how to test something, what to expect, and what actually happened**.

🎯 Why are Test Cases Important?

- Anyone can follow them, even a new tester.
- Avoids confusion (clear steps).
- Helps check if the software behaves as expected.
- Useful for regression (when testing again after fixes).

🔑 Key Terms (Explained Simply)

- **Test Case ID** → Unique number (like recipe name).
- **Precondition** → Things that must be ready before starting (ingredients ready).
- **Test Steps** → Actions to do (add noodles, add spices).
- **Expected Result** → What should happen (noodles ready, tasty).
- **Actual Result** → What really happened (maybe too salty).
- **Status** → Pass/Fail (tasty or not tasty).

📋 Real-Life Example (Online Shopping App – Login Page)

- **Test Case ID** → TC-101

- **Precondition** → User is on login page.
- **Steps** →
    1. Enter valid username.
    2. Enter valid password.
    3. Click Login button.
- **Expected Result** → User is successfully logged in.
- **Actual Result** → User is successfully logged in.
- **Status** → Pass ✅

📑 **Simple Test Case Template**

Test Case ID	Precondition	Steps	Expected Result	Actual Result	Status
TC-101	User on login page	Enter valid username & password → Click Login	User logs in successfully	User logs in successfully	Pass
TC-102	User on login page	Enter wrong password → Click Login	Error message appears	Error message appears	Pass
TC-103	User on login page	Leave fields empty → Click Login	Validation message shows	Validation message missing	Fail ❌

🧠 **Easy Memory Trick (Mnemonic)**

👉 **PSEAS** → **P**recondition, **S**teps, **E**xpected, **A**ctual, **S**tatus

Think of it like:
"Please Serve Every Appetizer Soon" 🍽️✨

Chapter 4: Test Scenarios – The Story

🌟 What is a Test Scenario?

Imagine you are planning to read a storybook. You don't write every single line of the story — you just know **the main theme**:

- "A boy goes on an adventure."
- "A girl solves a mystery."

👉 That main theme is like a **Test Scenario**.

A **Test Scenario** is a high-level description of what to test.
It is like an **outline of testing**, not detailed step-by-step like test cases.

🎯 Why are Test Scenarios Important?

- Gives a **bigger picture** of what needs to be tested.
- Saves time (you don't need to write hundreds of cases if scenarios are enough).
- Useful when requirements are large (you can break them into small stories).
- Helps ensure **coverage** (all features are thought about).

🔑 Key Terms (Explained Simply)

- **Scenario** → A story of what to test (e.g., "User logs into app").
- **Positive Scenario** → User does the right thing (correct login).
- **Negative Scenario** → User does the wrong thing (wrong password, empty fields).
- **Priority** → Importance (High = must test, Low = optional).

📋 Real-Life Example (Online Shopping App – Login Page)

- **Scenario 1 (Positive)** → User logs in with valid credentials.
- **Scenario 2 (Negative)** → User logs in with invalid credentials.
- **Scenario 3 (Negative)** → User tries to log in with empty fields.
- **Scenario 4 (Positive)** → User clicks on "Forgot Password" and gets reset link.

📑 Simple Test Scenario Template

Scenario ID	Scenario Description	Positive/Negative	Priority
SC-001	User logs in with valid credentials	Positive	High
SC-002	User logs in with invalid credentials	Negative	High
SC-003	User tries to log in with empty fields	Negative	Medium
SC-004	Forgot Password feature sends reset link	Positive	High

---

🧠 Easy Memory Trick (Mnemonic)

👉 **SPP** → **S**cenario, **P**ositive, **P**riority

Think of it like:
"Simple Positive Plans" 📖✨

Chapter 5: Bug Report – The Complaint Form

🌟 What is a Bug Report?

Imagine you buy a new toy car. When you press the button, instead of moving forward, it **makes a weird noise and stops**.
You go back to the shopkeeper and say:

“This toy is broken. I pressed the button and it didn’t work.”

👉 That explanation is like a **Bug Report**.

A **Bug Report** is a document where testers report problems (defects) they find in the software.

🎯 Why are Bug Reports Important?

- Helps developers understand and fix the issue.
- Makes sure issues are tracked and not forgotten.
- Avoids miscommunication (“It’s broken” → clear explanation instead).
- Improves software quality.

🔑 Key Terms (Explained Simply)

- **Bug ID** → Unique number for each bug (like complaint token).
- **Summary** → Short explanation of the bug.
- **Steps to Reproduce** → How the bug happens (like instructions).
- **Expected Result** → What should happen.
- **Actual Result** → What really happened.
- **Severity** → How serious it is (Minor / Major / Critical).
- **Priority** → How soon it should be fixed (High = urgent, Low = later).

📋 Real-Life Example (Online Shopping App – Login Page)

- **Bug ID** → BUG-001

- **Summary** → Login button not working.
- **Steps to Reproduce**:
    1. Open login page.
    2. Enter valid username and password.
    3. Click the login button.
- **Expected Result** → User should log in.
- **Actual Result** → Nothing happens when button is clicked.
- **Severity** → Critical (login is main feature).
- **Priority** → High (must fix immediately).

📑 Simple Bug Report Template

Bug ID	Summary	Steps to Reproduce	Expected Result	Actual Result	Severity	Priority
BUG-001	Login button not working	1. Enter valid credentials → 2. Click Login	User should log in	Nothing happens	Critical	High
BUG-002	Error message missing	1. Enter wrong password → 2. Click Login	Error message should show	No message displayed	Major	Medium

---

🧠 Easy Memory Trick (Mnemonic)

👉 **SSEASP** → **S**ummary, **S**teps, **E**xpected, **A**ctual, **S**everity, **P**riority

Think of it like:
"Some Small Errors Are Serious Problems" 🐞 ⚠️

Chapter 6: Test Report – The Report Card

🌟 What is a Test Report?

Think about your school exams. At the end, you get a **report card** that says:

- How many subjects you appeared for.
- How many subjects you passed.
- How many subjects you failed.
- Final conclusion: Passed or Failed.

👉 A **Test Report** in software testing is just like that report card.
It shows the **summary of testing activities and results** after execution.

🎯 Why is a Test Report Important?

- Provides managers/clients a **clear picture** of software quality.
- Shows how many test cases passed/failed.
- Tracks defect details.
- Helps decide if software is ready for release or needs more work.

🔑 Key Terms (Explained Simply)

- **Total Test Cases** → How many were written.
- **Executed** → How many were actually run.
- **Passed** → How many worked fine.
- **Failed** → How many didn't work.
- **Blocked** → Could not test due to dependency.
- **Defects Found** → Total bugs discovered.
- **Conclusion** → Ready for release or needs fixing.

📋 Real-Life Example (Online Shopping App – Login Page)

- **Total Test Cases** → 10

- **Executed** → 9 (1 was blocked due to server down).
- **Passed** → 7
- **Failed** → 2
- **Blocked** → 1
- **Defects Found** → 3
- **Conclusion** → Login module not ready for release, needs fixes.

📑 Simple Test Report Template

Total Test Cases	Executed	Passed	Failed	Blocked	Defects Found	Conclusion
10	9	7	2	1	3	Not Ready for Release 🚫

🧠 Easy Memory Trick (Mnemonic)

👉 **EPFD** → **E**xecuted, **P**assed, **F**ailed, **D**efects

Think of it like:
"Every Paper Feels Difficult" 📄 😅 (just like exams!)

Chapter 7: Sprint Report – The Weekly Diary

🌟 What is a Sprint Report?

Imagine your **weekly diary in school** 📘 :

- What homework you completed,
- What homework is pending,
- Any challenges you faced,
- Teacher's remarks.

👉 In Agile Scrum, a **Sprint Report** is just like that diary.
It shows **what was planned in the sprint vs. what was achieved** during that sprint.

🎯 Why is a Sprint Report Important?

- Provides **visibility** of sprint progress to team & stakeholders.
- Helps identify **incomplete tasks & blockers**.
- Improves **planning for the next sprint**.
- Acts as a **performance summary** for the sprint.

🔑 Key Sections in a Sprint Report

1. **Sprint Details** → Sprint number, duration, team.
2. **Planned Work** → How many stories/tasks were committed.
3. **Completed Work** → How many were actually done.
4. **Pending/Spillover Work** → Tasks not finished (moved to next sprint).
5. **Defects/Issues Raised** → Bugs logged during sprint.
6. **Velocity** → Work completed in terms of story points.
7. **Remarks/Conclusion** → Sprint successful or not.

📋 Real-Life Example (E-commerce Sprint Report)

- **Sprint Number** → Sprint 5

- **Duration** → 2 weeks (1st Aug – 14th Aug)
- **Team** → 5 members

Planned Stories	Completed	Pending	Defects Found	Velocity (Story Points)	Remarks
15	12	3	5	35/40	Sprint Successful 🎯 (80% completed)

📑 **Sprint Report Template**

Sprint No	Duration	Team Members	Planned Stories	Completed	Pending	Defects Raised	Velocity	Remarks

👉 Download Sprint Report Template (Word)

🧠 **Easy Memory Trick (Mnemonic)**

👉 **PCPVDR** → **P**lanned, **C**ompleted, **P**ending, **V**elocity, **D**efects, **R**emarks

Think of it like:
"Please Check Progress Very Diligently, Regularly" ✅

Chapter 8: RCA (Root Cause Analysis) – Finding the Real Problem

---

🌟 What is RCA?

Imagine you have **fever** 🤒.

- Taking paracetamol only reduces the temperature,
- But if the actual **root cause is an infection**, you'll keep falling sick.

👉 In testing, **RCA (Root Cause Analysis)** is the method of finding the **real reason behind a defect**, not just fixing symptoms.

🎯 Why Do We Need RCA?

- Prevents the **same bugs from happening again**.
- Improves **software quality**.
- Helps in **continuous improvement** of processes.
- Builds **trust** with clients & stakeholders.

🔑 Common Root Causes of Bugs

1. **Requirement Gap** → Misunderstanding business needs.
2. **Design Flaws** → Wrong architecture or logic.
3. **Coding Mistakes** → Typos, syntax, wrong implementation.
4. **Testing Gap** → Missed scenarios, poor test coverage.
5. **Environment Issues** → Incorrect configuration, version mismatch.

🛠️ RCA Techniques (Most Popular)

1. **5 Whys Technique** 🖐️
    - Keep asking "Why?" until you reach the real cause.
      Example:
    - Bug: Payment not successful
        - Why? → API failed

        - Why? → Timeout
        - Why? → Server overloaded
        - Why? → No load balancer
        - Why? → Architecture not designed for scaling. ✅ (Root cause found)
2. **Fishbone Diagram (Ishikawa Diagram)** 🐟
    - Categories → People, Process, Tools, Environment, Requirements.
    - Helps visually brainstorm causes.
3. **Pareto Analysis (80/20 Rule)**
    - 80% of defects come from 20% of causes → Focus on the main ones.

📔 Real-Life Example (E-commerce Login Bug)

Bug: Users unable to log in.
Symptom: Login page crashes with error.

👉 RCA using 5 Whys:

1. Why? → Database not responding.
2. Why? → Too many queries.
3. Why? → Poor indexing.
4. Why? → Developer missed database optimization.
5. Why? → No database review process in place. ✅

Root Cause: Lack of database review in coding standards.

📑 RCA Template

Defect ID	Defect Description	Root Cause	RCA Method (5 Whys/Fishbone)	Corrective Action	Preventive Action

👉 Download RCA Template (Excel)

🧠 **Easy Memory Trick (Mnemonic)**

👉 **RCA = R**eal **C**ause **A**nalysis
Think of it as:
“Remove Cause, Avoid Again” ✅

Chapter 9: Test Estimation – Planning the Effort Like a Budget

---

🌟 What is Test Estimation?

Think of **test estimation** like planning a **trip** 🚗:

- You decide **how many days**, **how much money**, and **what resources** are needed.

👉 In testing, **Test Estimation** means predicting the **time, effort, and cost** required to complete testing activities.

🎯 Why is Test Estimation Important?

- Ensures **projects finish on time** ⏰.
- Avoids **over-budgeting** 💰.
- Helps in **resource allocation** 👥.
- Builds **client confidence**.

🔑 Key Factors Affecting Estimation

1. **Project Size & Complexity**
    - Small website vs. banking application.
2. **Requirement Clarity**
    - Clear vs. ambiguous requirements.
3. **Test Coverage Needed**
    - Functional, performance, security, etc.
4. **Resources & Skills**
    - Number of testers & their experience.
5. **Tools & Environment**
    - Automation tools, test servers, devices.

🛠️ Test Estimation Techniques

1. Work Breakdown Structure (WBS) 🧩

- Break testing into smaller tasks.
- Estimate each task → Sum = total effort.

Example:

- Requirement analysis – 5 hrs
- Test case design – 15 hrs
- Test execution – 25 hrs
- Reporting – 5 hrs
  👉 Total = 50 hrs

---

2. Three-Point Estimation (PERT) 🎲

Formula:

Estimate = (Optimistic + Pessimistic + Most Likely) / 3

Example:

- Optimistic = 4 days
- Most Likely = 6 days
- Pessimistic = 10 days
  👉 Estimate = (4+6+10)/3 = 6.6 days

---

3. Expert Judgment 🧑‍🏫

- Ask experienced testers → They estimate based on past projects.

---

4. Function Point Analysis (FPA) 🔢

- Based on **number of functionalities** and their complexity.

---

5. Percentage Distribution Method 📊

- Distribute total project effort by percentage:
  - Test Planning = 10%

- Test Design = 30%
- Test Execution = 40%
- Reporting = 20%

📋 **Example: Estimation for E-Commerce Checkout Feature**

Task	Estimated Hours	Resources	Comments
Requirement Analysis	4 hrs	Tester A	Discuss with BA
Test Case Design	12 hrs	Tester A	Covers positive & negative
Test Data Preparation	6 hrs	Tester B	Dummy cards, coupons
Test Execution	20 hrs	Tester A+B	Manual testing
Defect Reporting & Retest	8 hrs	Tester A	Bugzilla tool
Final Report	4 hrs	Tester B	To client

👉 **Total = 54 hrs (~7 days)**

🧠 **Easy Memory Trick (Mnemonic)**

👉 **E.A.S.T. = Estimation Always Saves Time**

- **E** – Effort prediction
- **A** – Avoids surprises
- **S** – Smart planning
- **T** – Time saved

Chapter 9: Test Metrics & KPIs (Measuring Testing Success)

👉 What is it?

Think of test metrics like a cricket scoreboard 📊. They tell us how well the testing is going, just like runs, wickets, and overs show how well the game is progressing.

👉 Why do we need it?

Without metrics, we won't know:

- Are we testing enough?
- Are bugs reducing with time?
- Are we meeting deadlines?

👉 Types of Metrics:

1. **Test Coverage** – How much of the software is tested?
   (Like: Did we cover all subjects before an exam?)
2. **Defect Density** – Bugs per module/feature.
3. **Test Execution Rate** – How many test cases are executed per day?
4. **Pass/Fail Percentage** – Out of all test cases, how many passed/failed?
5. **Defect Leakage** – Bugs that slipped into production.

👉 Simple Example (Like School Report):

- Total Questions in Exam (Test Cases) = 50
- Answered (Executed) = 40
- Correct (Passed) = 30
- Wrong (Failed) = 10
- Unanswered = 10

👉 Template (Metrics Table):

Metric	Formula / Calculation	Example
Test Coverage	(Executed Cases ÷ Total Cases) × 100	40/50 = 80%
Pass %	(Passed Cases ÷ Executed Cases) × 100	30/40 = 75%
Fail %	(Failed Cases ÷ Executed Cases) × 100	10/40 = 25%

Metric	Formula / Calculation	Example
Defect Density	Defects ÷ Size of Module	5 ÷ 100 LOC
Defect Leakage	(Prod Defects ÷ Total Defects) × 100	2/10 = 20%

👉 **Mnemonic to Remember Metrics:**

“C-P-F-D-L” → Coverage, Pass, Fail, Density, Leakage

📩 **Template Download:** Test Metrics Template (Google Sheets)

Chapter 10: Test Strategy (The Big Master Plan)

👉 What is it?

If **Test Plan** is a roadmap 🗺️ for one trip, **Test Strategy** is the **city's master map** 🌆 showing all possible routes. It defines the **overall approach to testing** at an organizational/project level.

👉 Why is it important?

Without strategy, every team may test in a different way, leading to chaos. Strategy ensures **consistency & quality** across all projects.

👉 Key Elements of Test Strategy:

1. **Testing Approach** – Manual, Automation, or Hybrid.
2. **Test Levels** – Unit, Integration, System, UAT.
3. **Test Types** – Functional, Performance, Security, Compatibility.
4. **Test Environment Setup** – Servers, Tools, Data.
5. **Test Data Strategy** – Real data, mock data, synthetic data.
6. **Defect Management** – How bugs will be reported & tracked.
7. **Tools to be Used** – JIRA, Selenium, Postman, etc.

👉 Simple Analogy (Cricket Example):

- Strategy = How the **whole team will play** (Batting first, defensive/attacking style).
- Plan = The **specific batting order for today's match**.

👉 Template (Test Strategy Outline):

Section	**Description Example**
Purpose	Define why testing is needed
Scope	In-scope: Mobile App, API. Out-of-scope: Backend DB tuning
Approach	Hybrid Testing (Manual + Automation)
Test Levels	Unit, Integration, System
Tools	JIRA, Selenium, Postman
Defect Management	Bugzilla/JIRA

Section	Description Example
Risks & Mitigation	Risk: Delayed test env → Mitigation: Use staging env
Deliverables	Test Plan, Test Cases, Reports

👉 **Mnemonic to Remember Sections:**
“P-S-A-L-T-D-R-D” → Purpose, Scope, Approach, Levels, Tools, Defects, Risks, Deliverables

📥 **Template Download:** Test Strategy Template (DOCX)

Chapter 11: Test Closure Report

🔹 What is a Test Closure Report?

A **Test Closure Report** is like the **final goodbye letter** from the testing team to the project.
It says:

- "We tested everything."
- "Here's what we found."
- "Here's what went well, what didn't, and what we learned."

It's created at the **end of a test cycle** (end of release, sprint, or project).

🔹 Why do we need it?

- To **summarize the whole testing effort**.
- To **share lessons learned** for the next project.
- To **confirm that testing is officially done**.
- To give management a **clear closure document**.

🔹 Key Components of a Test Closure Report

Think of it like a **Graduation Report Card** 🎓

1. **Project Information**
    - Project Name
    - Version/Release Number
    - Test Cycle Dates
2. **Testing Summary**
    - What was tested
    - What was NOT tested
    - Types of testing done (Functional, Regression, Performance, etc.)
3. **Metrics / Numbers**

- Total Test Cases Designed
- Total Test Cases Executed
- Pass/Fail/Blocked statistics
- Defects raised & status (Open/Closed/Deferred)

4. **Defect Summary**
   - High severity issues fixed
   - Issues still open (with justification)
5. **Environment Details**
   - Tools used
   - Test environments (Staging, UAT, Production)
6. **Lessons Learned**
   - What went well
   - What could be improved
   - Suggestions for future projects
7. **Sign-off**
   - Final confirmation from QA Lead / Manager
   - Stakeholders acknowledgment

🔹 **Example (Simple Table)**

Section	**Details**
Project Name	Online Shopping App – Release 2.0
Test Cycle Dates	1st Aug – 15th Aug 2025
Testing Summary	Functional, Regression, and UAT completed. Performance skipped (time).
Metrics	Total Cases: 120, Executed: 115, Passed: 110, Failed: 5, Blocked: 5
Defects	20 raised, 18 fixed, 2 deferred (minor UI).
Environment	Browser: Chrome 139, OS: Windows 11, Tools: JIRA, Selenium.

Section	Details
Lessons Learned	Need earlier test data setup. Automation saved time.
Sign-off	QA Lead: ✅

🔹 **Downloadable Template**

📄 Download Test Closure Report Template (DOCX)

✅ Mnemonic to remember → **"S-M-D-E-L-S"**

- **S**ummary
- **M**etrics
- **D**efects
- **E**nvironment
- **L**essons learned
- **S**ign-off

Chapter 12: Test Metrics and Measurements

🔹 What are Test Metrics?

- **Metrics** are like the **scoreboard** of testing.
- They give **numerical data** about the progress, quality, and efficiency of testing.
- Example: "Out of 100 test cases, 90 passed, 10 failed" → That's a **metric**.

🔹 Why are Metrics Important?

- To **measure quality** of the product.
- To **track progress** of testing.
- To **find gaps** and improve processes.
- To **show evidence** to management with numbers instead of just words.

🔹 Types of Test Metrics

1. Process Metrics 🛠️

Measure the **testing process efficiency**.

- Example: Test case preparation time, Test execution rate.

2. Product Metrics 📦

Measure the **quality of the product** under test.

- Example: Defect density, Defect severity index.

3. Project Metrics 📊

Measure the **overall health of the project**.

- Example: Effort variance, Schedule variance.

🔹 Common Test Metrics with Formulas

1. **Test Case Execution Progress**

   Progress (%)=Executed Test Cases×100/ Total Test Cases

Example: 80/100 = 80% executed

---

2. **Defect Density**

   Defect Density= Total Defects /Size of Module (e.g., KLOC or Function Points)

Example: 15 defects / 1000 lines of code = 0.015

---

3. **Defect Leakage**

   Leakage (%)=Defects found in UAT or Production×100/ Total Defects

   Example: 5 leaked defects / 50 total defects = 10% leakage

---

4. **Defect Removal Efficiency (DRE)**

   DRE (%)=Defects removed during testing×100/ Total Defects (Testing + Prod)

Example: 45 fixed / 50 total = 90%

---

5. **Test Case Effectiveness**

   Effectiveness (%)=Defects detected×100/ Total Test Cases Executed

   Example: 20 defects / 100 test cases = 20%

🔹 **Sample Metrics Report (Simple Table)**

Metric Name	Value	Status
Total Test Cases	120	✅
Executed Test Cases	110	91% done
Passed Test Cases	100	91% pass
Failed Test Cases	10	🔴 Alert
Defect Density	0.02	Average
Defect Leakage	5%	Acceptable

Metric Name	Value	Status
Defect Removal Efficiency	95%	✅ Good

🔹 **Tools to Track Metrics**

- **JIRA** → defect metrics, execution trends
- **TestRail** → test case execution reports
- **Excel / Google Sheets** → manual tracking
- **Power BI / Tableau** → advanced dashboards

✅ Mnemonic to remember → **“P-P-P-M” (Process, Product, Project, Metrics)**
Or in rhyme:
"Metrics keep track,
So quality won’t lack!" 🎵

Chapter 11: Test Metrics and KPIs (Measuring Testing Success)

👶 Kid-Friendly Explanation

Imagine you're studying for exams. Your parents want to know:

- How many chapters you finished reading 📚
- How many times you got full marks ✅
- How many silly mistakes you made ❌

This "report" is nothing but **Metrics & KPIs** in software testing.
They help managers see whether testing is successful or needs improvement.

🧩 Key Concepts

1. **Metrics** → Numbers we collect during testing.
    - Example: *Number of test cases executed*.
2. **KPIs (Key Performance Indicators)** → Important metrics that show progress or success.
    - Example: *% of test cases passed*.

📊 Common Test Metrics

Metric	Meaning	Example
Test Case Execution Rate	How many tests were run	80/100 test cases executed
Defect Density	Bugs per module/feature size	10 bugs in 1000 lines of code
Test Coverage	How much of app is tested	90% code covered by tests
Defect Leakage	Bugs missed in testing, found later	5 defects found by users after release
Defect Severity Index	Average seriousness of bugs	2.3 (on scale of 1–4)

📈 Example KPI Dashboard

KPI	Target	Actual	Status
Test Coverage	85%	88%	✅
Defect Leakage	< 5%	2%	✅
Execution Rate	95%	80%	⚠️
Severity Index	< 3	2.5	✅

📁 Template: Test Metrics & KPI Report

Project Name: _____
Sprint/Release: _____

Metric/KPI	Target	Actual	Status	Comments
Test Case Execution Rate	95%	___	___	___
Test Coverage	85%	___	___	___
Defect Density	< 1 per KLOC	___	___	___
Defect Leakage	< 5%	___	___	___

✅ Download Word/Excel Template: Test Metrics & KPI Template

🎯 Mnemonic for Memory

👉 **"CARE"**

- **C** – Coverage
- **A** – Accuracy of Execution
- **R** – Rate of Bugs
- **E** – Escaped Defects

Chapter 12: Test Closure Report (Final Goodbye Report)

👶 Kid-Friendly Explanation

Imagine you completed your exams 🎓. Before you relax, you check:

- How many subjects you studied 📖
- How many questions you got right ✅
- What mistakes you made ❌
- What you'll do better next time 📝

That's exactly what a **Test Closure Report** does in software testing.
It's the *final goodbye report* after testing is completed.

🧩 Key Concepts

What is a Test Closure Report?

A **formal document** prepared at the end of the testing phase to summarize:

- Testing activities performed
- Bugs found & fixed
- Test coverage achieved
- Lessons learned

📊 Contents of a Test Closure Report

1. **Project Information**
    - Project name, release version, test manager
2. **Test Summary**
    - Number of test cases planned vs executed
    - % passed, failed, blocked
3. **Defect Summary**
    - Number of defects raised, fixed, deferred
    - Severity & priority distribution

4. **Test Coverage**
    - How much requirement/code is tested
5. **Environment Details**
    - Hardware, software, tools used
6. **Metrics & KPIs**
    - Execution rate, defect density, leakage
7. **Risks & Issues**
    - Unresolved defects, limitations
8. **Lessons Learned**
    - What worked well, what can improve
9. **Sign-off**
    - Test Manager approval

📁 **Template: Test Closure Report**

Project Name: _____
Version: _____
Test Manager: _____

1. Test Summary

- Planned Test Cases: ___
- Executed Test Cases: ___
- Passed: ___ | Failed: ___ | Blocked: ___

2. Defect Summary

- Total Defects: ___
- Fixed: ___ | Deferred: ___ | Open: ___

3. Coverage

- Requirement Coverage: ___%
- Code Coverage: ___%

4. Metrics

- Execution Rate: ___%
- Defect Density: ___ per KLOC
- Leakage: ___%

5. Risks & Issues

- Pending defects: ___
- Technical limitations: ___

6. Lessons Learned

- Positives: _____
- Improvements: _____

7. Sign-off

- QA Lead: _____
- Test Manager: _____

📥 Download Template: Test Closure Report Template

🎯 **Mnemonic for Memory**

👉 **"SCORE"**

- **S** – Summary of testing
- **C** – Coverage details
- **O** – Outstanding defects
- **R** – Risks & issues
- **E** – End sign-off