# Test Objectives, Test Scope, coverage, Testware and Test reporting

## Test Objectives

### 🎯 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Test Objectives** are the **specific goals** you want to achieve through testing.

They answer the question:

“**Why** are we testing this system?”

They give the testing effort a **purpose** — so the team knows what to look for, how to measure success, and when testing can stop.

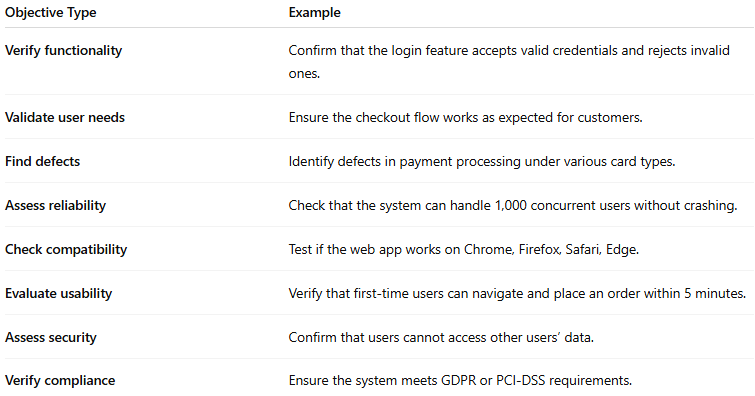
### 🔹 Why Test Objectives Matter

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* They align testing with **business and project goals**.
* They guide **test design** and **prioritization**.
* They provide a **basis for test completion criteria** (entry/exit conditions).
* They help in **reporting** (we can say: “We achieved our objectives or we didn’t”).

### 🔹 Common Examples of Test Objectives

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**



| **Objective Type** | **Example** |
| --- | --- |
| **Verify functionality** | Confirm that the login feature accepts valid credentials and rejects invalid ones. |
| **Validate user needs** | Ensure the checkout flow works as expected for customers. |
| **Find defects** | Identify defects in payment processing under various card types. |
| **Assess reliability** | Check that the system can handle 1,000 concurrent users without crashing. |
| **Check compatibility** | Test if the web app works on Chrome, Firefox, Safari, Edge. |
| **Evaluate usability** | Verify that first-time users can navigate and place an order within 5 minutes. |
| **Assess security** | Confirm that users cannot access other users’ data. |
| **Verify compliance** | Ensure the system meets GDPR or PCI-DSS requirements. |

### 🖥️ Example (Online Banking App)

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Project:** Online Banking System  
**Test Objectives:**

1. Verify that customers can successfully transfer funds between accounts.
2. Ensure that invalid transactions (e.g., insufficient funds) are properly blocked.
3. Confirm that all transactions are logged with correct timestamps and user IDs.
4. Test that the system can handle 500 simultaneous transactions without failure.
5. Validate that sensitive data (passwords, PINs) are never shown or stored in plain text.

### 💡 In short:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* Test Objectives = **what you want to accomplish** with testing.
* They’re set **before** test design starts and guide the entire testing process.

## Test Scope

### 📦 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Test Scope** describes **what will be tested** and **what will not be tested** in a project.

It answers the questions:

“**Where** will we apply testing?”  
“**Which features, systems, or platforms** are in or out of testing?”

It sets **boundaries** so the team knows exactly what to focus on and avoids wasting time on areas not relevant for this release.

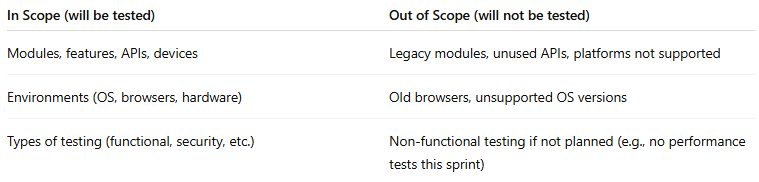
### 🔹 Why Test Scope Matters

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* It prevents **scope creep** (people assuming “everything” will be tested).
* It helps **estimate effort** and allocate the right resources.
* It clarifies **stakeholder expectations**.
* It gives testers and developers a **clear checklist** of what areas need coverage.

### 🔹 Typical Content of Test Scope

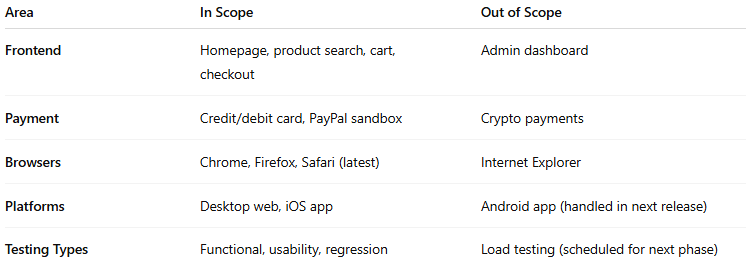
**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**



| **In Scope (will be tested)** | **Out of Scope (will not be tested)** |
| --- | --- |
| Modules, features, APIs, devices | Legacy modules, unused APIs, platforms not supported |
| Environments (OS, browsers, hardware) | Old browsers, unsupported OS versions |
| Types of testing (functional, security, etc.) | Non-functional testing if not planned (e.g., no performance tests this sprint) |

**🖥️ Example (E-Commerce App)**

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**



| **Area** | **In Scope** | **Out of Scope** |
| --- | --- | --- |
| **Frontend** | Homepage, product search, cart, checkout | Admin dashboard |
| **Payment** | Credit/debit card, PayPal sandbox | Crypto payments |
| **Browsers** | Chrome, Firefox, Safari (latest) | Internet Explorer |
| **Platforms** | Desktop web, iOS app | Android app (handled in next release) |
| **Testing Types** | Functional, usability, regression | Load testing (scheduled for next phase) |

### 💡 In short:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* **Test Objectives** → *Why* we test (the purpose).
* **Test Scope** → *What* we will and will not test (the boundaries).

## Coverage

### 📊 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Test Coverage** is a **measure of how much of the software has been tested**.

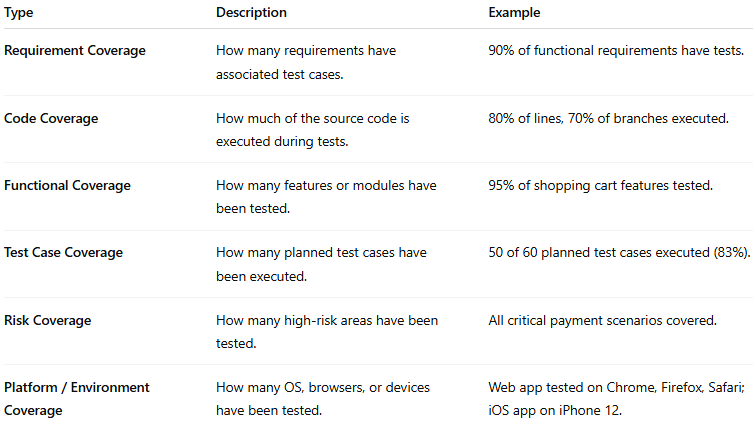
It tells you **what percentage of the system, requirements, or code has been exercised** by your tests.

Think of it as a way to answer:

“Have we tested enough to be confident the system works?”

### 🔹 Types of Coverage

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**



| **Type** | **Description** | **Example** |
| --- | --- | --- |
| **Requirement Coverage** | How many requirements have associated test cases. | 90% of functional requirements have tests. |
| **Code Coverage** | How much of the source code is executed during tests. | 80% of lines, 70% of branches executed. |
| **Functional Coverage** | How many features or modules have been tested. | 95% of shopping cart features tested. |
| **Test Case Coverage** | How many planned test cases have been executed. | 50 of 60 planned test cases executed (83%). |
| **Risk Coverage** | How many high-risk areas have been tested. | All critical payment scenarios covered. |
| **Platform / Environment Coverage** | How many OS, browsers, or devices have been tested. | Web app tested on Chrome, Firefox, Safari; iOS app on iPhone 12. |

### 🔹 Why Coverage Matters

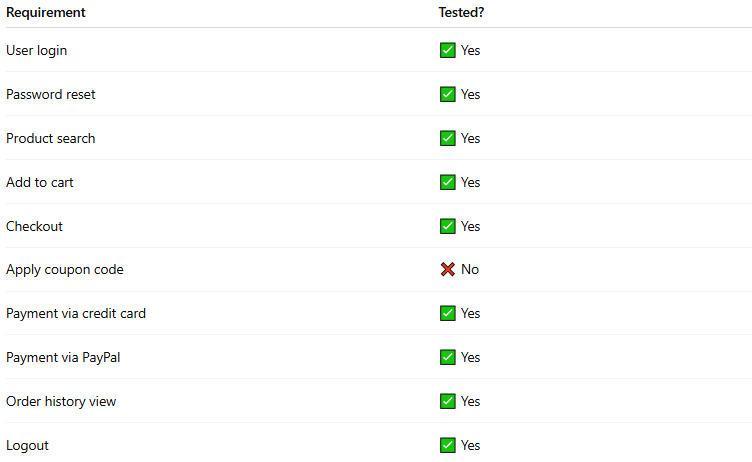
**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* Helps identify **gaps in testing**.
* Provides a **quantitative measure** to report testing progress.
* Guides decisions on **when to stop testing** (e.g., exit criteria).
* Ensures **critical areas get sufficient focus**.

### 🖥️ Example

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Imagine you have 10 requirements for an online store:



| **Requirement** | **Tested?** |
| --- | --- |
| User login | ✅ Yes |
| Password reset | ✅ Yes |
| Product search | ✅ Yes |
| Add to cart | ✅ Yes |
| Checkout | ✅ Yes |
| Apply coupon code | ❌ No |
| Payment via credit card | ✅ Yes |
| Payment via PayPal | ✅ Yes |
| Order history view | ✅ Yes |
| Logout | ✅ Yes |

* **Requirement Coverage** = 9/10 = **90%**
* If code coverage shows only 75% of lines in checkout module are executed, it indicates some paths were not tested.

### required level of coverage

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
Great question — the **“required level of coverage”** is about setting **how much testing is enough**.

#### 🎯 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

The **required level of coverage** is the **minimum percentage or extent of testing** that must be completed before testing can be considered sufficient to meet quality objectives.

It defines the **target coverage** for requirements, code, features, or risks, depending on the project.

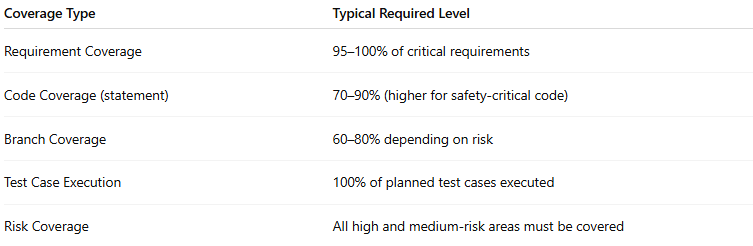
In other words: it answers the question  
“How much of the system must be tested before we can release it or consider it safe?”

#### 🔹 Factors That Influence Required Coverage

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

1. **Criticality / Risk**
   * High-risk or safety-critical modules → require higher coverage.
   * Example: Banking transactions, healthcare devices.
2. **Regulatory / Compliance Requirements**
   * Some industries mandate minimum coverage.
   * Example: ISO, FDA, PCI-DSS.
3. **Project Constraints**
   * Time, budget, available resources.
   * May limit how much coverage is achievable.
4. **Type of Testing**
   * Functional: often 100% of critical features.
   * Code: may aim for 70–90% statement or branch coverage.
   * Regression: cover all previously fixed defects.

#### 🔹 Examples of Required Coverage

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  


| **Coverage Type** | **Typical Required Level** |
| --- | --- |
| Requirement Coverage | 95–100% of critical requirements |
| Code Coverage (statement) | 70–90% (higher for safety-critical code) |
| Branch Coverage | 60–80% depending on risk |
| Test Case Execution | 100% of planned test cases executed |
| Risk Coverage | All high and medium-risk areas must be covered |

#### 🔹 Example Scenario

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Project:** Online Banking App

* Critical features: Fund transfer, login, payment processing → **100% tested**.
* Non-critical features: Profile settings → **80% tested**.
* Automated regression suite → **90% code coverage**.

**Exit Criteria:**

* All critical requirements tested successfully.
* No critical defects remain.
* Required coverage levels met.

#### 💡 Key point:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
The required level of coverage is **not universal**. It is **defined for each project** based on **risk, regulations, and business priorities**.

### 💡 In short:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
Coverage is **a metric or measure** — it does not guarantee there are no defects, but it shows **how thoroughly the system has been tested**.

## Testware

### 🧰 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Testware** is the **collection of all artifacts created and used during testing** to plan, design, execute, and report tests.

In simple terms: **everything a tester uses or produces to test software is “testware.”**

It’s like the **toolbox and instructions** combined for testing a project.

### 🔹 Components of Testware

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

1. **Test Cases / Test Scripts**
   * Step-by-step instructions for running a test.
   * Can be manual or automated.
2. **Test Data**
   * Input values, files, databases, or configurations used in tests.
3. **Test Environment Configurations**
   * Settings, servers, network setups, browsers, and hardware info.
4. **Test Documentation**
   * Test plans, test strategies, test designs, and traceability matrices.
5. **Defect Logs & Reports**
   * Records of defects, test results, and execution logs.
6. **Testing Tools**
   * Automation scripts, simulators, or frameworks needed to execute tests.

### 🖥️ Example

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

For an **E-commerce app**, the testware could include:

* Test Cases: “Verify that adding a product to the cart works.”
* Test Data: Product IDs, user accounts, coupon codes.
* Test Scripts: Selenium scripts automating login and checkout.
* Test Environment: Chrome browser on Windows 10 with a staging database.
* Test Reports: Execution logs showing passed and failed tests.

### 🔹 Key Points

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* Testware is **different from software under test** — it is for testing, not production.
* Proper management of testware ensures **repeatable, efficient, and reliable testing**.
* Testware can be **reused** across releases or projects.

### Testware products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Absolutely! Let’s break down **testware for each test activity** step by step. I’ll list them clearly without using a table.

#### 1. Test Planning Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

These are the artifacts created to guide and organize testing:

* Test Plan document
* Test Strategy document
* Risk assessment reports
* Test objectives
* Resource and schedule plans
* Test estimation sheets
* Templates for test cases and test logs

#### 2. Test Monitoring and Test Control Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

These are artifacts used to track progress and control testing:

* Test progress reports
* Test execution status reports
* Defect tracking logs
* Metrics dashboards (e.g., test coverage, defect density)
* Updated risk lists and mitigation plans
* Test schedule adjustments
* Action plans for deviations

#### 3. Test Analysis Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Artifacts created during analysis of requirements and test basis:

* Requirements traceability matrix (RTM)
* Identified test conditions
* Prioritized test items based on risk
* Defects found in requirements or specifications
* Testability reports
* Test design checklist references

#### 4. Test Design Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Artifacts created to define **how testing will be done**:

* Detailed test cases and expected results
* Test scripts (manual or automated)
* Test data requirements
* Test environment requirements
* Test charters (for exploratory testing)
* Coverage items (e.g., boundary conditions, equivalence classes)

#### 5. Test Implementation Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Artifacts prepared for execution:

* Implemented test cases and test scripts
* Test data sets loaded into the environment
* Automated test scripts ready to run
* Test suites (organized sets of test procedures)
* Test procedures for manual execution
* Configured test environment with necessary tools

#### 6. Test Execution Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Artifacts produced **during testing**:

* Test execution logs
* Actual vs expected results records
* Defect reports and anomaly reports
* Test run reports (pass/fail summary)
* Evidence of testing (screenshots, logs, videos)
* Updated status of test cases (executed, blocked, failed)

#### 7. Test Completion Work Products

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Artifacts created **after testing is finished**:

* Test summary report or final test report
* Lessons learned and retrospective notes
* Updated defect logs with resolution status
* Archived testware for future reuse
* Test metrics and coverage reports
* Sign-off or approval documents from stakeholders

#### 💡 Key point:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
All these artifacts together form **testware**, which ensures testing is **organized, repeatable, and traceable** throughout the project.

### Level of Detail of Testware

#### 📝 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

The **level of detail of testware** refers to **how much information is included in the test artifacts to execute, understand, and maintain tests effectively**.

In simple terms: it is **how precise and complete a test case, test script, or other testware is** — enough for the tester to perform the test correctly, without ambiguity, but not so detailed that it becomes cumbersome.

#### 🔹 Why Level of Detail Matters

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* Too **little detail** → testers may misinterpret steps, skip checks, or fail to reproduce defects.
* Too **much detail** → testware becomes hard to maintain, overly rigid, and slows down testing.
* The **level of detail should match the context**: experienced testers may need less detail than novices; automated scripts may require more precision than manual test cases.

#### 🔹 Factors Influencing Level of Detail

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

1. **Type of Testing**
   * Manual testing often requires clear step-by-step instructions.
   * Exploratory testing may need only a high-level charter.
2. **Experience of Testers**
   * Experienced testers can follow high-level instructions.
   * New or less experienced testers need more detailed steps.
3. **Criticality / Risk**
   * High-risk modules may require very detailed test cases.
   * Low-risk features can be tested with less detail.
4. **Automation**
   * Automated test scripts require precise inputs, expected outputs, and environment setup.
5. **Project Constraints**
   * Time, resources, and maintenance effort influence the appropriate detail level.

#### 🔹 Examples of Levels of Detail

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**High-level detail (light):**

* Test Case: “Verify user can log in with valid credentials.”
* Expected Result: “User is successfully logged in and sees the dashboard.”
* Suitable for experienced testers or exploratory testing.

**Medium-level detail:**

* Steps:
  1. Open login page.
  2. Enter valid username and password.
  3. Click “Login.”
  4. Expected Result: Dashboard appears.
* Suitable for most manual testing.

**High-level detail (very detailed):**

* Steps include exact field names, data to enter, button clicks, screenshots references, preconditions, postconditions, environment setup, and cleanup steps.
* Often used for automated tests or high-risk safety-critical systems.

#### 💡 Key point:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
The **level of detail of testware** should be **fit for purpose**: enough to ensure tests are executed correctly, reproducibly, and efficiently, but not so much that they become unmanageable.

### 💡 In short:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
**Testware = All the tools, data, scripts, and documentation needed to test a system effectively.**

## Test reporting

### 📝 Definition

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Test Reporting** is the **process of documenting and communicating the results of testing**.

In other words, it’s **how testers share what they found**, including defects, progress, coverage, and quality status, with stakeholders like developers, managers, and clients.

### 🔹 Purpose of Test Reporting

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* Provides **visibility** of testing progress.
* Helps **track defects** and evaluate their severity.
* Supports **decision-making** about release readiness.
* Offers a **record** of what was tested, how it was tested, and what issues were found.

### 🔹 Key Elements of a Test Report

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

1. **Test Summary**
   * Scope of testing performed
   * Objectives achieved
   * Features/modules tested
2. **Test Execution Results**
   * Number of test cases executed, passed, failed, blocked
   * Pass/fail percentage
3. **Defects / Anomalies**
   * Defects discovered (critical, major, minor)
   * Status of defects (open, fixed, deferred)
4. **Coverage Metrics**
   * Requirements coverage
   * Code coverage
   * Risk coverage
5. **Risks / Issues**
   * Any obstacles encountered during testing
   * Recommendations or mitigation plans
6. **Conclusion / Recommendations**
   * Quality status of the product
   * Readiness for release or further testing

### 🖥️ Example (Short Test Report Summary)

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

**Project:** Online Banking App  
**Testing Phase:** System Testing

* Total test cases: 100
* Executed: 100
* Passed: 90
* Failed: 8
* Blocked: 2
* Defects logged: 8 critical, 5 major, 2 minor
* Risk coverage: 100% of high-risk modules tested

**Conclusion:** Critical modules are stable; some defects must be fixed before release.

### 💡 Key point:

**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**  
Test reporting ensures **transparent communication** of testing progress and quality status — it’s essential for stakeholders to make informed decisions about software release.