# Test Condition, Test case, Test data, Test Basis and Test environment

## 1-Test Condition

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

A **test condition** is **something that you want to verify or validate during testing** — it’s a specific feature, function, requirement, or characteristic of the system that you need to check.

It answers the question: **“What should I test?”**

### Key points

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

* **Level:** Higher-level than a test case (more general).
* **Source:** Comes from requirements, designs, user stories, or risks.
* **Purpose:** Guides what test cases will be written.
* **Type:** Can be functional (feature-based) or non-functional (performance, security, usability).

### Examples

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

📱 **Food delivery app**:

1. User can log in with a valid username and password. *(Functional)*
2. App must display search results within 2 seconds. *(Performance)*
3. Payment must be secure (HTTPS connection). *(Security)*
4. User can apply a discount coupon at checkout. *(Functional)*

💡 **Hierarchy**:

* **Test condition** → broad “thing to check”
* **Test case** → detailed steps to verify it
* **Test step** → smallest individual action in the test

## 2-Test case

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

A **test case** is a **set of specific actions, inputs, and expected results** designed to check whether a particular part of the system works as intended.

It answers: **“How will I test it?”**

### Key points

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

* **Derived from**: Test conditions or requirements.
* **Purpose**: Provide clear, repeatable steps so anyone can perform the test.
* **Includes**:  
  1. **Test case ID** (unique number/name)
  2. **Test description** (what it checks)
  3. **Preconditions** (what must be true before starting)
  4. **Test steps** (exact actions)
  5. **Test data** (input values)
  6. **Expected result**
  7. **Actual result** (after execution)
  8. **Status** (Pass/Fail)

### Example

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

**Requirement:** “User can log in with valid credentials.”

* **Test Case ID:** TC-001
* **Description:** Verify login with valid username & password.
* **Preconditions:** User account exists; app is open.
* **Test steps:**
  1. Enter valid username.
  2. Enter valid password.
  3. Click **Login**.
* **Test data:** Username: testuser, Password: Pass123
* **Expected result:** User is taken to the dashboard page.
* **Actual result:** (filled during execution)
* **Status:** Pass/Fail

💡 **Difference from test condition**:

* **Test condition**: *User can log in with valid credentials* (broad “what to test”)
* **Test case**: Detailed, step-by-step instructions for testing that condition.

### Full example

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

Sure — here’s a **full example** showing your test condition and its matching test case in a real QA table format.

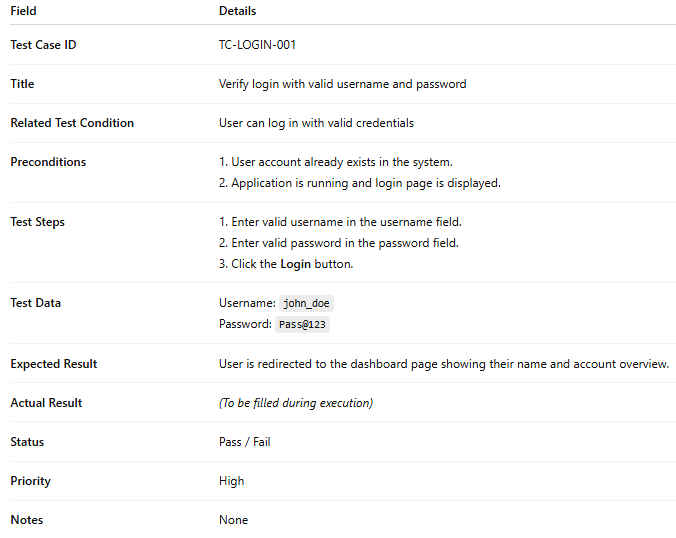
#### Test Condition

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

User can log in with valid credentials.

#### Test Case Example

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**



|  |  |
| --- | --- |
| **Field** | **Details** |
| **Test Case ID** | TC-LOGIN-001 |
| **Title** | Verify login with valid username and password |
| **Related Test Condition** | User can log in with valid credentials |
| **Preconditions** | 1. User account already exists in the system.2. Application is running and login page is displayed. |
| **Test Steps** | 1. Enter valid username in the username field.2. Enter valid password in the password field.3. Click the **Login** button. |
| **Test Data** | Username: john\_doePassword: Pass@123 |
| **Expected Result** | User is redirected to the dashboard page showing their name and account overview. |
| **Actual Result** | *(To be filled during execution)* |
| **Status** | Pass / Fail |
| **Priority** | High |
| **Notes** | None |

✅ **Key difference**:

* **Test Condition** → *Broad goal*: “User can log in with valid credentials.”
* **Test Case** → *Exact recipe* for a tester to follow, including input data and what should happen.

### Test Case Prioritization

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

**Test Case Prioritization** is the process of arranging and executing test cases in an **order of importance, risk, or value** so that the **most critical tests are run earlier**.

👉 The goal is to **maximize fault detection, improve efficiency, and ensure higher quality within limited time or resources**.

It’s especially important in **Regression Testing** and **Agile/Continuous Delivery**, where testing time is often short.

#### 🎯 Objectives of Test Case Prioritization

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

1. **Early fault detection** – find critical defects as soon as possible.
2. **Maximize test effectiveness** – run the most valuable test cases first.
3. **Optimize limited resources** – when time/budget is restricted.
4. **Support continuous integration (CI/CD)** – run quick, high-priority tests in every build.
5. **Improve customer satisfaction** – ensure business-critical features are tested early.

#### 🧮 Factors Used for Prioritization

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

When deciding which test cases are more important, testers consider:

* **Risk level** (high-risk features tested first).
* **Business value** (customer-critical functionality).
* **Frequency of use** (functions used daily by end users).
* **Defect history** (areas where bugs were often found before).
* **Complexity** (more complex modules are more error-prone).
* **Recent changes in code** (modified or new code gets top priority).
* **Dependencies** (core modules before dependent ones).

#### 📊 Example

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

Suppose you are testing an **online banking system** with 10 test cases.  
 You only have time to run 4 before release.

✅ You prioritize:

1. **Login with valid credentials** (High risk + critical function).
2. **Transfer funds between accounts** (Core business function).
3. **Check account balance** (High frequency use).
4. **Logout securely** (Security risk).

👉 Less critical test cases (e.g., “Update profile picture”) are executed later.

#### ⚙️ Common Prioritization Techniques

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

1. **Risk-based prioritization** – tests ordered by risk impact/probability.
2. **Requirement-based prioritization** – focus on high-priority business requirements.
3. **Coverage-based prioritization** – maximize code/requirement coverage early.
4. **History-based prioritization** – prioritize modules with frequent past failures.
5. **Hybrid approaches** – combination of the above.

#### ✅ Summary

**# Source**: **Chatgot (GPT-5-Standard) at [8/18/2025]**

**Test Case Prioritization** = The process of selecting which test cases to run first, based on **risk, business value, defect history, and changes**. It ensures **critical defects are found early** and testing remains **effective under constraints**.

## 3-Test Suite

### 🔹 What is a Test Suite?

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

A **test suite** is a **collection of test cases** that are grouped together because they:

* Test the same feature, functionality, or module.
* Or serve a specific testing purpose (e.g., regression, performance, smoke testing).

Think of it like a **folder** 📂 that contains related test cases, making it easier to manage and run them.

### 🔹 Example

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

Imagine you’re testing a **Login feature** of an app:

* **Test Cases** inside the suite:
  + Login with valid username + password ✅
  + Login with invalid password ❌
  + Login with empty username ❌
  + Login with SQL injection attempt ❌

All these cases together form a **“Login Test Suite”**.

### 🔹 Types of Test Suites

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

* **Functional test suite** → for specific functions (e.g., shopping cart).
* **Regression test suite** → to ensure old features still work after changes.
* **Smoke test suite** → quick basic tests to check if the system is stable.
* **Performance test suite** → stress, load, scalability tests.

#### 1-Functional Test Suite

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

**🔹 Functional Test Suite**

A **functional test suite** is a collection of **test cases** that verify a specific **function or feature** of the software system, based on the functional requirements.

* It focuses on **“what the system should do”**, not how it’s implemented.
* Ensures each function works correctly under different conditions (valid, invalid, boundary inputs).

**🔹 Example**

Let’s say we’re testing the **“User Registration” feature** in a web app.  
A **Functional Test Suite** for this might include test cases like:

1. Register with valid username, email, and password → ✅ success.
2. Register with missing required field (e.g., email) → ❌ error message shown.
3. Register with weak password → ❌ rejected.
4. Register with already-used email → ❌ rejected.
5. Register with maximum allowed characters in fields → ✅ success.

All these test cases together form the **Functional Test Suite for Registration**.

**✅ In short:**

A **Functional Test Suite** = a group of test cases designed to check that a **specific feature or function** works as expected according to requirements.

#### 2-Regression Test Suite

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

A **regression test suite** is a collection of test cases designed to ensure that **new code changes** (bug fixes, enhancements, or new features) **have not broken or negatively impacted existing functionality** of the system.

* Focus: **Check that old features still work** after modifications.
* It is usually **re-executed after every build, release, or code change**.
* Can include both **manual and automated tests** (automation is very common here).

**🔹 Example**

Imagine we already have a working **User Registration + Login system**.

Now the dev team updates the app to support **social login (Google/Facebook)**.

A **Regression Test Suite** may include:

1. Register with email + password (old function) → ✅ should still work.
2. Login with email + password → ✅ still works.
3. Password reset → ✅ still works.
4. Session timeout handling → ✅ unchanged.
5. New social login → ✅ works without breaking old login.

**✅ In short**

* **Functional Test Suite** → verifies a **specific feature** works according to requirements.
* **Regression Test Suite** → re-tests **existing features** to confirm they still work after changes.

#### 3-Smoke Test Suite

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

A **smoke test suite** is a small collection of **high-level test cases** that check whether the **basic and critical functions** of the system are working correctly after a new build or deployment.

* It’s like a **quick health check** of the application.
* The purpose: **decide if the build is stable enough** for further testing.
* If smoke tests fail → no point continuing with deeper testing.

**🔹 Example**

Suppose we’re testing an **e-commerce website** 🛒.  
A **Smoke Test Suite** may include:

1. Can the homepage load successfully?
2. Can a user log in with valid credentials?
3. Can a user search for a product?
4. Can a product be added to the cart?
5. Can checkout start (not full payment flow, just opening the page)?

👉 If any of these fail → the build is considered **unstable** and returned to developers.

**✅ Difference from Regression Test Suite**

* **Smoke Test Suite** → runs a **quick check** of the most critical features to see if build is testable.
* **Regression Test Suite** → runs **detailed checks** to confirm nothing broke after changes.

#### 4-Performance Test Suite

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

A **performance test suite** is a collection of test cases designed to **evaluate how the system performs** under specific conditions, focusing on **speed, stability, scalability, and responsiveness**.

It checks **non-functional requirements** like:

* How fast does the system respond? ⏱
* Can it handle many users at the same time? 👥
* Does performance stay stable under long usage? 📈

**🔹 What it Includes**

Performance test suite may contain:

1. **Load Tests** → Measure system behavior under expected user load.
2. **Stress Tests** → Push system beyond limits to see where it breaks.
3. **Scalability Tests** → Check how well the system handles growth in data/users.
4. **Endurance (Soak) Tests** → Run the system for long durations to find memory leaks or slowdowns.

**🔹 Example (Banking App 💳)**

For a **banking web application**, the performance test suite may include:

* Test if **login API responds within 2 seconds** under 10,000 concurrent users.
* Test if the **fund transfer process** can handle **1,000 transactions per minute**.
* Test if the system stays stable when used continuously for **24 hours**.

✅ **Key difference from functional/regression suites**:

* Functional & regression suites → check if the system **works correctly**.
* Performance test suite → checks if the system **works efficiently under load**.

### ✅ In short:

**# Source**: **Chatgpt (GPT-5-Standard) at [9/14/2025]**

A **test suite** = an organized set of related test cases, grouped for execution and management.

## 4-Test data

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

**Test data** is the **specific input values** you use while running a test case.

It’s what you **feed into the system** to check if it behaves as expected.  
 Without test data, you can’t actually execute most tests.

### Key points

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

* **Purpose**: Simulate real user inputs or system inputs.
* **Types**: Can be *valid* (correct) or *invalid* (to test error handling).
* **Source**: Can be created manually, taken from production data, or generated automatically by tools.
* **Examples**:  
  + Login credentials: username: testuser, password: Pass123
  + Search term: "iPhone 15"
  + Credit card number: 4111 1111 1111 1111
  + File upload: resume.pdf

### Example in context

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

**Test Case:** Verify login with valid credentials.

* **Test Data:**
  + Username: john\_doe
  + Password: Pass@123

When the tester runs this case, they’ll use these exact inputs in the fields.

💡 **Tip**: Test data is different from **expected result** —

* Test data = *What you put in*
* Expected result = *What you expect to get out*

## 5-Test Basis

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

**Test basis** is the **source of information** from which you derive your **test conditions** and **test cases**.

Think of it like the **blueprint** for your testing — you look at the documents, designs, or requirements, and from those, you decide *what* to test.

### **Key points**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

* **Purpose:** Provides the foundation for identifying what should be tested.
* **Examples of test basis sources:**
  + **Requirements documents** (e.g., “User must be able to log in with valid credentials”)
  + **Design specifications**
  + **User stories** in Agile
  + **Flowcharts**
  + **Wireframes / UI mockups**
  + **System architecture diagrams**
  + **Regulations or standards**
  + **Risk analysis documents**
* **Relation to testing:**
  + **Test Basis** → *What you read*
  + **Test Condition** → *What you decide to test from it*
  + **Test Case** → *How you test it step-by-step*

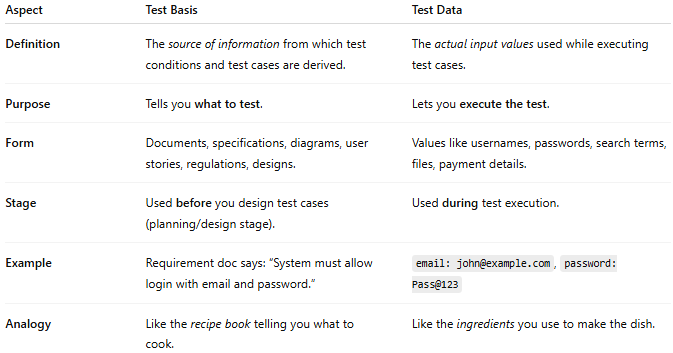
### **Example**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

* **Test Basis:** Requirement document says:  
    
    
   “The system shall allow users to log in using a valid email and password.”
* **Test Condition:** User can log in with valid credentials.
* **Test Case:**
  1. Open login page
  2. Enter john@example.com and Pass@123
  3. Click “Login”
  4. Verify dashboard is displayed.

### Difference between Test Basis and Test Data

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**



|  |  |  |
| --- | --- | --- |
| **Aspect** | **Test Basis** | **Test Data** |
| **Definition** | The *source of information* from which test conditions and test cases are derived. | The *actual input values* used while executing test cases. |
| **Purpose** | Tells you **what to test**. | Lets you **execute the test**. |
| **Form** | Documents, specifications, diagrams, user stories, regulations, designs. | Values like usernames, passwords, search terms, files, payment details. |
| **Stage** | Used **before** you design test cases (planning/design stage). | Used **during** test execution. |
| **Example** | Requirement doc says: “System must allow login with email and password.” | email: john@example.com, password: Pass@123 |
| **Analogy** | Like the *recipe book* telling you what to cook. | Like the *ingredients* you use to make the dish. |

### **Quick Example in Context**

**# Source**: **Chatgpt (GPT-5-Standard) at [8/12/2025]**

1. **Test Basis**: Requirement — *"User can log in with valid credentials"*.
2. **Test Condition**: Login with valid credentials.
3. **Test Data**: Username = "john", Password = "Pass@123".

## 6- Test environment

### 🖥️ Definition

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

A **test environment** is the **setup** where software is installed and tested. It includes all the **hardware**, **software**, **network**, **tools**, and **configurations** needed to run the application safely and accurately during testing. Think of it as a **safe playground** that mimics the real-world system — so testers can check features without breaking the live (production) system.

### 🔹 Components of a Test Environment

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

1. **Hardware**
   * Servers, computers, mobile devices, etc.
2. **Software**
   * Operating systems, databases, application servers, browsers, APIs.
3. **Network Configuration**
   * Firewalls, bandwidth, security settings, connections to other systems.
4. **Test Data**
   * Sample users, records, or files used for testing.
5. **Testing Tools**
   * Manual or automated testing tools (e.g., Selenium, JMeter, Postman).
6. **Access / Permissions**
   * User accounts with the right roles to test various scenarios.

### 🔹 Purpose

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

* To ensure testing happens in a **controlled**, **repeatable** environment.
* To **simulate production** (live system) as closely as possible.
* To **isolate defects** — making sure bugs are from the code, not from environment issues.

### 🔹 Example

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

If you’re testing an online banking system, a proper test environment might include:

* A database with fake accounts and balances.
* A web server with the same code as production.
* Simulated network conditions.
* Test users with roles like customer, admin, teller.

This way testers can try deposits, transfers, and logins **without touching real customer data**.

### 💡 In short:

**# Source**: **Chatgpt (GPT-5-Standard) at [9/4/2025]**  
A **test environment** = the technical “laboratory” where testers safely and accurately test software before it goes live.