# Test Strategy Test Plan, and Test Techniques

## Test Strategy

### 🧭 Definition

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

A **test strategy** is a **high-level document** (or section of a test plan) that describes **how testing will be approached** for a project or organization.

Think of it as the **roadmap** or **policy** — it doesn’t list individual test cases, but it defines the **overall direction, scope, goals, and methods** for testing.

It answers the big questions:

* What kind of testing will we do?
* What levels (unit, integration, system, acceptance) will we cover?
* What tools, techniques, and standards will we follow?
* What is “in scope” and “out of scope”?
* How will we measure success?

**# Source**: **Chatgot (GPT-5-Standard) at [9/8/2025]**

It means you decide **how you will do the testing** and **what will be tested**. This can include:

* **Scope of testing**: which features, modules, or components will be included or excluded.
* **Types of testing to use**: e.g., functional, performance, security, usability.
* **Techniques**: e.g., black-box, white-box, exploratory testing.
* **Sequence or prioritization**: which tests to run first or on which components.
* **Resources and tools**: what testing tools or environments you will use.

In short, the **approach** is your overall plan or method for performing the tests effectively **within the defined scope**.

### 🔹 Key Components of a Test Strategy

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

Most strategies (especially in ISTQB and industry practice) cover:

1. **Scope and Objectives**
   * What will be tested, and what will not.
2. **Testing Types / Levels**
   * Functional, non-functional (performance, security, usability, etc.).
   * Unit, integration, system, acceptance.
3. **Approach / Methodology**
   * Manual vs automated testing.
   * Risk-based, exploratory, model-based, etc.
4. **Test Deliverables**
   * What documents, reports, logs will be produced.
5. **Environment & Tools**
   * Hardware, software, databases, networks, testing tools.
6. **Entry and Exit Criteria**
   * Conditions for starting/stopping testing.
7. **Schedule & Resources**
   * Who will test, when, and with what dependencies.
8. **Risk Management**
   * How risks will be identified and mitigated in testing.
9. **Metrics and Reporting**
   * How progress and quality will be measured (e.g., defect density, test coverage).

### 🔹 Purpose

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

* Provides **consistency**: all teams test in the same direction.
* Communicates the testing plan to stakeholders.
* Sets **expectations**: what quality level to aim for, what risks are accepted.
* Supports **compliance**: in regulated industries, a strategy is often mandatory.

### 💡 In simple terms:

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

* **Test Plan** = project-specific details (who, when, where).
* **Test Strategy** = organization/project-wide high-level policy (what, why, how in broad terms).

### Sample Test Strategy Document

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

Excellent 👍 — let’s build this step by step: a **full example** of a **Test Strategy** document, so you can see exactly how it looks in practice.

**🧭 Sample Test Strategy Document**

**1. Introduction**

This document defines the test strategy for the **Online Banking System (OBS)** project.  
It describes the overall testing approach, scope, objectives, and resources to ensure the product meets functional and non-functional requirements.

**2. Scope & Objectives**

* **In Scope**
  + Core banking features: Login, Account Management, Funds Transfer, Bill Payment.
  + Web and mobile platforms.
  + Functional, security, and performance testing.
* **Out of Scope**
  + Legacy modules (e.g., old reporting system).
  + Third-party integrations not updated in this release.
* **Objectives**
  + Validate that all functional requirements are met.
  + Verify security (authentication, authorization, encryption).
  + Ensure acceptable performance under expected load.

**3. Test Levels & Types**

* **Levels**
  + **Unit Testing**: Performed by developers.
  + **Integration Testing**: Validate APIs and data flow.
  + **System Testing**: Verify end-to-end features.
  + **User Acceptance Testing (UAT)**: Final validation by business users.
* **Types**
  + **Functional Testing**: Positive, negative, boundary, and regression tests.
  + **Non-Functional Testing**:
    - Performance (load and stress testing).
    - Security (penetration testing, data privacy checks).
    - Usability testing on different devices/browsers.

**4. Test Approach**

* Combination of **manual** and **automated** testing:
  + **Manual Testing**: Exploratory, usability, ad-hoc checks.
  + **Automated Testing**: Regression suite for core functionalities using **Selenium** and **JUnit**.
* **Risk-Based Prioritization**:
  + High-risk modules (Funds Transfer, Login) tested first and more thoroughly.
* Defect tracking and reporting via **JIRA**.

**5. Test Environment & Tools**

* **Environment**
  + Web server: Apache Tomcat.
  + Database: Oracle 19c with test data.
  + Mobile testing: Android 12, iOS 16 devices.
  + Network: Simulated production bandwidth and security configuration.
* **Tools**
  + Test Management: TestRail.
  + Automation: Selenium WebDriver, JUnit.
  + Performance: JMeter.
  + Security: OWASP ZAP.

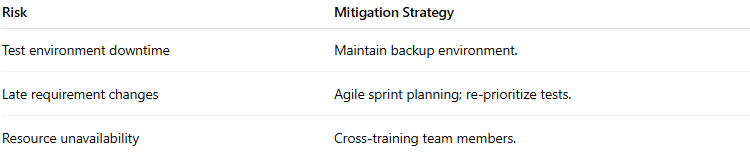
**6. Entry & Exit Criteria**

* **Entry**
  + All features implemented and unit tested.
  + Test environment ready.
  + Test data prepared.
* **Exit**
  + All critical and high defects fixed.
  + Test coverage ≥ 95% of defined test cases.
  + UAT sign-off received.

**7. Deliverables**

* Test plans for each cycle.
* Test cases and automation scripts.
* Daily and final test execution reports.
* Defect logs with severity and priority.
* Final test summary report.

**8. Risks & Mitigation**



| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Test environment downtime | Maintain backup environment. |
| Late requirement changes | Agile sprint planning; re-prioritize tests. |
| Resource unavailability | Cross-training team members. |

**9. Schedule & Resources**

* **Test Cycle**: 4 weeks (2 sprints).
* **Team**: 2 QA engineers (manual), 1 QA automation engineer, 1 performance tester.
* **Reporting**: Daily stand-ups, weekly status reports to project stakeholders.

**10. Metrics & Reporting**

* Test case execution progress (planned vs executed).
* Defect density per module.
* Test pass/fail rates.
* Automation coverage percentage.
* Mean time to defect detection.

This Test Strategy is reviewed and approved by the **Project Manager**, **QA Lead**, and **Business Owner**. It serves as a guiding document for all testing activities on the OBS project.

✅ This is a **realistic, professional-style example** — you can adapt it to any project by replacing the project name, scope, tools, and specific details.

### real-world-inspired sample

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

Here’s a **real-world-inspired sample** of a **Test Strategy document**—not copied from a private or internal source, but based on authentic, publicly available references that reflect industry-standard structure and content:

**Sample Test Strategy: E-Commerce Web Application**

This example is adapted from publicly shared guidance, not a proprietary document, and illustrates how real testing strategies are structured.

**Overview**

* **Product Name**: E-Commerce Web Application
* **Revision**: v1.5
* **Overview**: An online e-commerce platform where users can browse products, add things to their cart, and complete secure purchases. Comprises a responsive web interface, a mobile app, backend systems, plus inventory and payment services.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Product Background**

* **Previous Versions**: v1.0, v1.2, v1.3
* **Defect History**: Previous issues included payment gateway integration failures and cart item retention errors. These were resolved via unit and integration testing, improving system reliability.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Features to Be Tested**

* **User Features**: Product search, cart functionality, user registration, checkout workflow.
* **Layers**:
  + Frontend: React.js
  + Backend: Node.js
  + Database: MySQL
  + API integrations
  + Mobile app flows and push notifications
* **Non-functional Tests**:
  + **Performance**: Stress testing up to 5,000 concurrent users using JMeter
  + **Security**: Vulnerability scanning using OWASP ZAP  
    ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Entry Criteria (System Test)**

* Core functionality (search, login, cart) must be functional.
* All unit tests must pass successfully.
* Code freeze in place; latest build available in the repository.
* All known issues logged in the defect tracking system.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Exit Criteria (System Test)**

* All planned system tests executed.
* All happy-path scenarios (e.g., user registration, product purchase) pass.
* A working build is available for all supported platforms.
* No critical (showstopper) defects remain.
* Maximum thresholds: ≤5 major bugs, ≤10 minor bugs.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Deliverables**

* Test plans (system, regression, performance).
* Documented test cases in tools like Jira or TestRail.
* Execution logs of all tests.
* Defect reports hosted in the bug-tracking system.
* Coverage reports showing percentage of features/code tested.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Metrics & Targets**

* **Test Coverage**: Aim for 95% across unit, integration, and system tests.
* **Defect Density**: Maintain less than 1 defect per 1,000 lines of code.
* **Performance Target**: Ensure key transactions respond within 2 seconds.
* **Defect Leakage**: Keep production leakage under 2%.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Risks**

* Instability of payment gateway under high load.
* Cross-browser inconsistencies, especially in older browsers.
* Degraded performance with >5,000 concurrent users.
* Potential security vulnerabilities introduced by new authentication features.  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**References**

* Internal API documentation
* Selenium and JMeter setup guides
* OWASP standards for security testing  
  ([Katalon](https://katalon.com/resources-center/blog/test-strategy?utm_source=chatgpt.com" \o "What is Test Strategy? Guide To Develop Test Strategy (With Sample)))

**Summary**

This **real example** reflects the structure and content found in genuine organizational test strategies: inclusive of scope, planned tests, quality metrics, tools, and risk management. It’s based on practical patterns widely taught and used in the industry.

Would you like a similarly detailed **Agile-style test strategy example** or an annotated template based on this structure?

## Test Plan

### 📝 Definition

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

A **Test Plan** is a **formal document** that describes **what**, **how**, **when**, and **who** will test a software product.

It acts like a **roadmap** for testing — guiding the QA team and communicating with stakeholders.

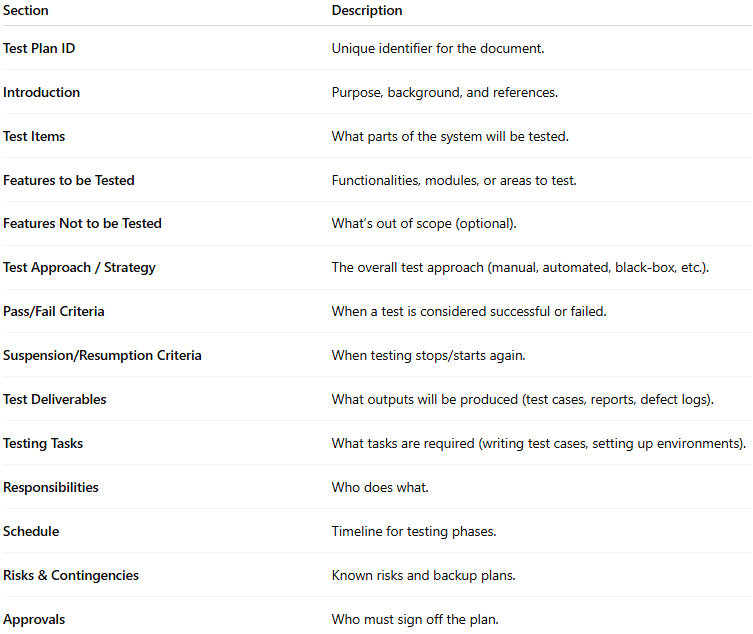
### 🎯 Purpose of a Test Plan

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

* To **define the scope** of testing (what will and will not be tested).
* To **describe the testing approach** (manual, automated, tools).
* To **set schedules** and **responsibilities**.
* To **identify risks** and how to mitigate them.
* To **align** QA with business and project goals.

**🧩 Typical Components (based on IEEE 829 / ISTQB)**

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



| **Section** | **Description** |
| --- | --- |
| **Test Plan ID** | Unique identifier for the document. |
| **Introduction** | Purpose, background, and references. |
| **Test Items** | What parts of the system will be tested. |
| **Features to be Tested** | Functionalities, modules, or areas to test. |
| **Features Not to be Tested** | What’s out of scope (optional). |
| **Test Approach / Strategy** | The overall test approach (manual, automated, black-box, etc.). |
| **Pass/Fail Criteria** | When a test is considered successful or failed. |
| **Suspension/Resumption Criteria** | When testing stops/starts again. |
| **Test Deliverables** | What outputs will be produced (test cases, reports, defect logs). |
| **Testing Tasks** | What tasks are required (writing test cases, setting up environments). |
| **Responsibilities** | Who does what. |
| **Schedule** | Timeline for testing phases. |
| **Risks & Contingencies** | Known risks and backup plans. |
| **Approvals** | Who must sign off the plan. |

### 🖥️ Example (simplified)

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

**Project:** Online Shopping App  
**Version:** 1.0  
**Prepared by:** QA Team

1. **Introduction:**  
   Testing the checkout and payment modules of the Online Shopping App.
2. **Scope:**  
   Test the following modules: Cart, Checkout, Payment, Order Confirmation.  
   Out of scope: Admin backend.
3. **Approach:**
   * Manual functional testing (black-box techniques: EP, BVA).
   * Automated regression testing using Selenium.
   * Risk-based prioritization.
4. **Environment:**  
   Test environment mirrors production: Windows Server, MySQL, Payment Gateway Sandbox.
5. **Schedule:**
   * Test case design: July 1–5
   * Test execution: July 6–15
   * Bug fix verification: July 16–20
6. **Responsibilities:**
   * QA Lead: Approve test plan, monitor progress.
   * QA Engineers: Write & execute test cases.
   * Developers: Fix defects.
7. **Risks:**
   * Payment gateway downtime may delay testing.
   * Limited test data for some scenarios.
8. **Deliverables:**
   * Test cases, test execution reports, defect logs, final test summary report.
9. **Approval:**  
   Signed by Project Manager, QA Lead.

### Is Test Strategy is same test plan?

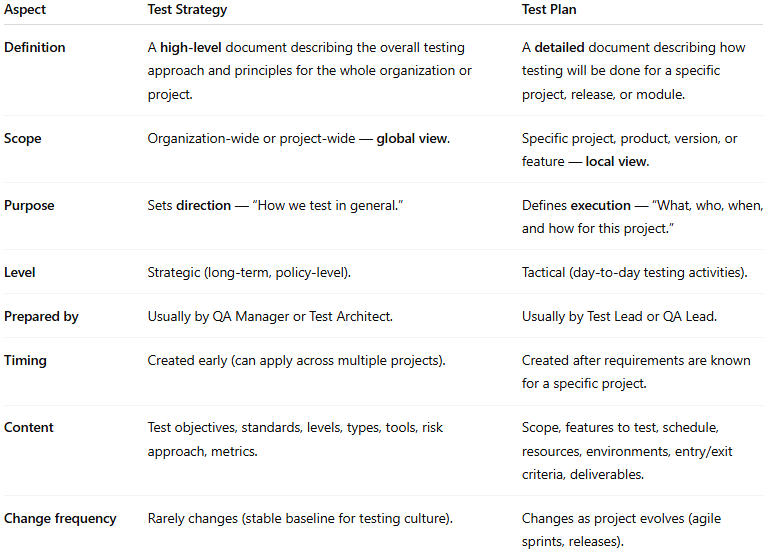
**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

Excellent follow-up 👏 — this is where many people get confused.

**Short answer:** ❌ No — a Test Strategy is **not** the same as a Test Plan. They are related, but different in **purpose**, **scope**, and **level**.

#### 🧭 Difference between Test Strategy & Test Plan

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



| **Aspect** | **Test Strategy** | **Test Plan** |
| --- | --- | --- |
| **Definition** | A **high-level** document describing the overall testing approach and principles for the whole organization or project. | A **detailed** document describing how testing will be done for a specific project, release, or module. |
| **Scope** | Organization-wide or project-wide — **global view**. | Specific project, product, version, or feature — **local view**. |
| **Purpose** | Sets **direction** — “How we test in general.” | Defines **execution** — “What, who, when, and how for this project.” |
| **Level** | Strategic (long-term, policy-level). | Tactical (day-to-day testing activities). |
| **Prepared by** | Usually by QA Manager or Test Architect. | Usually by Test Lead or QA Lead. |
| **Timing** | Created early (can apply across multiple projects). | Created after requirements are known for a specific project. |
| **Content** | Test objectives, standards, levels, types, tools, risk approach, metrics. | Scope, features to test, schedule, resources, environments, entry/exit criteria, deliverables. |
| **Change frequency** | Rarely changes (stable baseline for testing culture). | Changes as project evolves (agile sprints, releases). |

#### 🖥️ Example relationship:

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

* **Test Strategy:**

"All company projects will use risk-based testing, automated regression, and mandatory security testing following ISO standards."

* **Test Plan (for Online Shopping App v1.0):**

"We will test checkout and payment modules manually using black-box techniques, and run automated regression on the login and cart modules using Selenium."

#### 💡 Think of it this way:

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

* **Test Strategy** = *Company / Organization / Project-wide policy* → like the “testing constitution.”
* **Test Plan** = *Project-specific roadmap* → like a “mission plan” for this version or release.

### Summary

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

💡 In short: A test plan is **the master plan** for how testing will be done — it ensures everyone is aligned and nothing important is forgotten.

## Test Techniques

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

### 🧪 Definition

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

A **test technique** is a **method** used to design test cases.

In other words: it’s **how** you decide **what inputs, conditions, and data** to use when testing a system.

It’s about **systematically selecting** a set of tests that have a good chance of finding defects — without testing every single possible input (which is impossible).

### 🔹 Why They Matter

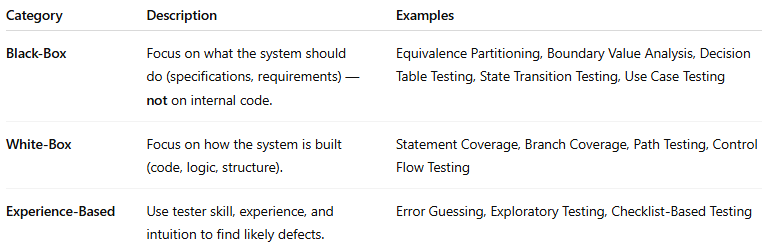
**# Source**: **Chatgot (GPT-5-Standard) at [9/4/2025]**

* They **increase coverage** (more of the system is checked).
* They **reduce effort** (fewer but smarter tests).
* They **make testing measurable** (you can define coverage criteria).

### 🔹 Main Categories of Test Techniques

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

There are three broad groups (ISTQB standard):



| **Category** | **Description** | **Examples** |
| --- | --- | --- |
| **Black-Box** | Focus on what the system should do (specifications, requirements) — **not** on internal code. | Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transition Testing, Use Case Testing |
| **White-Box** | Focus on how the system is built (code, logic, structure). | Statement Coverage, Branch Coverage, Path Testing, Control Flow Testing |
| **Experience-Based** | Use tester skill, experience, and intuition to find likely defects. | Error Guessing, Exploratory Testing, Checklist-Based Testing |

### 🖤 Black-Box Test Techniques (examples)

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

1. **Equivalence Partitioning (EP)**  
   Divide input data into groups (“partitions”) where all values are expected to behave the same — then test just one from each group.  
   Example: Age input (1–120) → partitions: <1 (invalid), 1–120 (valid), >120 (invalid).
2. **Boundary Value Analysis (BVA)**  
   Test at the edges of input ranges, where errors are common.  
   Example: If age is valid from 18–60 → test 17, 18, 60, 61.
3. **Decision Table Testing**  
   Use a table of conditions and actions to make sure all combinations are tested.  
   Example: Login success/failure based on username, password, account status.
4. **State Transition Testing**  
   Test how the system moves between states.  
   Example: ATM → Idle → Card Inserted → PIN Entered → Transaction → Idle.
5. **Use Case Testing**  
   Design tests based on user journeys or scenarios.

### 🤍 White-Box Test Techniques (examples)

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

1. **Statement Coverage** — Make sure every line of code runs at least once.
2. **Branch Coverage** — Ensure every decision (true/false) is executed at least once.
3. **Path Coverage** — Execute all possible paths through a piece of code.
4. **Loop Testing** — Test loops with zero, one, and many iterations.

### 🧠 Experience-Based Test Techniques (examples)

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

1. **Error Guessing** — Guess where defects might be based on experience (e.g., null inputs, special characters).
2. **Exploratory Testing** — Simultaneous learning, test design, and execution.
3. **Checklist-Based Testing** — Use known problem checklists (e.g., security, accessibility).

### 💡 In short:

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

* **Test techniques** are tools in a tester’s toolkit to design better, smarter tests.
* You don’t have to use only one — professional testing usually combines several techniques.

## 💡 Summary:

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

* **Test Strategy** → *High-level approach: overall goals, scope, and direction of testing.*
* **Test Plan** → Project-specific document: detailed schedule, tasks, resources, and deliverables for testing.
* **Test Techniques** → *Practical methods: how to design test cases.*