Test strategy template

A test strategy sets how your team tests a product: scope, approach, test levels and types, automation, environments, default quality gates, tools, roles, metrics and risks. It changes rarely. Each release's test plan applies it. Fill in every section for your product, delete the prompts in italics once done, and remove any section you don't need. A filled example for an e-commerce web and mobile product follows the blank template.

1. Document control

Who owns this strategy, who reviews it, and when it is next reviewed.

Product or product line	
Strategy version	
Owner (role)	
Reviewers (roles)	
Approved by (role)	
Date	
Next review date	
Related documents	

2. Purpose and scope

Why this strategy exists, which products and teams it covers, and what is out of scope.

Purpose	
Products and components in scope	
Teams covered	
Out of scope	
Assumptions	

3. Quality objectives

What quality means for this product, how you will measure it and the target you aim for.

Objective	How we measure it	Target

4. Test approach

Your overall approach, and which approach applies when. Most teams combine several.

Overall approach (summary)	

Approach	What it means here	When it applies
Risk-based		
Shift-left		
Requirements-based		
Exploratory		
Automated regression		
Shift-right		

5. Test levels

Which levels you test at, who owns each one, where it runs and the gate it must pass.

Level	What it covers	Owner	Where it runs	Gate
Unit				
Integration				
System				
Acceptance				

6. Test types

Which types of testing you run, when each runs and who owns it.

Test type	What it checks	When it runs	Owner
Functional			
Regression			
Performance			
Security			
Accessibility			
Usability			
Exploratory			

7. Automation approach

What you automate and what stays manual, how you use the test pyramid, your tools and how you keep suites healthy. Treat the test pyramid as guidance: many fast unit tests, fewer integration and API tests, and a small set of UI end-to-end tests. It is a shape to aim for, not a fixed ratio.

What we automate	
What stays manual	
Test pyramid (guidance)	
Tools and frameworks	
Where automated tests run	
Flaky test rule	

8. Default entry and exit criteria

Team-wide default gates that every release starts from. Release-specific criteria, and suspension and resumption rules, live in each test plan.

Stage	Entry criteria (default)	Exit criteria (default)
Story testing		
System testing		
Release (go or no-go)		

Each release test plan can tighten these gates or add its own. Full entry and exit criteria with examples: katalon.com/resources-center/blog/test-plan

9. Test environments and test data

Each environment's purpose, the data it holds and who manages it. Add your browser and device coverage rule and test data rules.

Environment	Purpose	Test data	Managed by
Local			
CI			
QA			
Staging			
Production			

Browser and device coverage	
Test data rules	

10. Defect management

How you rate severity and priority, how triage works and the flow a defect follows.

Severity	Definition	Example
Critical		
High		
Medium		
Low		

Priority	Meaning	Fix by
P1		
P2		
P3		

Triage	
Defect flow	
Defect record	

11. Tools

The tool you use for each purpose and who owns it.

Purpose	Tool	Owner
Test management and reporting		
Test automation (web, mobile, API)		
Unit testing		
CI/CD		
Defect tracking		
Performance testing		
Security testing		
Accessibility testing		
Browser and device coverage		

12. Roles and responsibilities

Who does what in testing. Use roles, not names, so the strategy outlasts team changes.

Role	Responsibilities
Head of QA	
QA engineer	
Developer	
Product manager	
Engineering manager	
Security lead	
Platform team	

13. Metrics and reporting

Which metrics you track and why, which reports you send, to whom and how often.

Metric	Why we track it	Reviewed

Report	Audience	Cadence	Where

14. Risks and mitigations

Product and project risks that could hurt quality, how likely and severe each is, and what you will do about it.

Risk	Likelihood	Impact	Mitigation	Owner

15. Review and approval

How often the strategy is reviewed, what triggers an early review, and who approves it.

Review cadence	
Triggers for an early review	
Change process	

Role	Name	Decision	Date

Test strategy example: e-commerce web and mobile product

Example only. Example Store is a fictional product. Every number, target and threshold in this example is illustrative, not a benchmark. Replace them with values that fit your product and risk.

1. Document control

Product or product line	Example Store (fictional online store): storefront web app, iOS and Android shopping apps, public order API
Strategy version	1.0
Owner (role)	Head of QA
Reviewers (roles)	Engineering manager, product manager, mobile lead, security lead
Approved by (role)	VP of Engineering
Date	2026-11-06
Next review date	2027-05-06, or sooner if the architecture or risk profile changes
Related documents	Test policy; release test plans; test case repository; defect tracker workflow

2. Purpose and scope

Purpose	Give every Example Store release one agreed way of testing, so release teams don't re-decide coverage, environments and exit rules each time.
Products and components in scope	Storefront web app (desktop and mobile browsers), iOS and Android apps, search, cart, checkout and payment flow, accounts and loyalty, order management API.
Teams covered	Three product squads (Browse, Checkout, Accounts), the mobile team and the platform team.
Out of scope	Internal warehouse system (has its own strategy); the payment provider's own systems (we test our integration only); marketing site content.
Assumptions	Each squad has at least one embedded QA engineer. Staging matches the production configuration.

3. Quality objectives

Objective	How we measure it	Target
Customers can always browse, add to cart and pay	Results of P1 checkout journeys on supported browsers and devices	All P1 journeys pass before release (illustrative)
Find defects before customers do	Defect escape rate: production defects as a share of all defects	Falling quarter on quarter (illustrative)
Give developers fast feedback	Time from merge to automated regression result	Under 30 minutes for the CI suite (illustrative)
Stay accessible	WCAG 2.2 AA checks on key journeys	No open critical accessibility defects at release
Protect customer data	Security scan and penetration test findings	No open critical or high findings at release

4. Test approach

Overall approach (summary)	Risk-based first. Every feature gets a likelihood and impact rating at refinement, and the rating sets test depth. Testing starts at refinement, not after the build, and continues after release through monitoring.

Approach	What it means here	When it applies
Risk-based	Rate each feature high, medium or low on likelihood and impact. High-risk areas get deeper coverage, more automation and exploratory sessions.	Every feature, at refinement. Checkout, payment and account security are always high risk.
Shift-left	QA joins refinement, writes acceptance criteria with product and reviews designs. Developers write unit and API tests with the code.	Every story.
Requirements-based	Test cases trace to acceptance criteria and requirement IDs.	Payment, tax and privacy features, and anything with a contractual or legal requirement.
Exploratory	Time-boxed, charter-based sessions by QA engineers and developers.	New features before release and after major UI changes.
Automated regression	Stable, repeatable checks run in the CI pipeline.	Smoke suite on every merge; full regression nightly and on each release candidate.
Shift-right	Synthetic monitoring of key journeys, error tracking and staged rollouts behind feature flags.	After every production deploy.

5. Test levels

Level	What it covers	Owner	Where it runs	Gate
Unit	Functions, components and business rules in isolation	Developers	CI, on every commit	Must pass to merge. Coverage reviewed in code review, no fixed target.
Integration	Service calls, API contracts, database, payment provider sandbox	Developers and QA engineers	CI and QA environment	API and contract suites pass before deploy to staging
System	End-to-end journeys across web, mobile apps and API	QA engineers	Staging	P1 journeys pass before release sign-off
Acceptance	Business rules and acceptance criteria with realistic data	Product manager, supported by QA	Staging	Product sign-off on each epic

6. Test types

Test type	What it checks	When it runs	Owner
Functional	Features meet their acceptance criteria	Every story, within the sprint	QA engineers and developers
Regression	Existing features still work after a change	Smoke on every merge; full suite nightly and before each release	QA engineers (automation owners)
Performance	Response times, load and stress on browse, search and checkout	Monthly baseline; before peak-season releases; after major architecture changes	Platform team with QA
Security	Code and dependency scans, dynamic scans, penetration tests	Scans on every build; dynamic scan weekly; external penetration test yearly	Security lead
Accessibility	WCAG 2.2 AA on key journeys, screen reader use on web and mobile	Automated checks in CI; manual audit each quarter and before redesigns	QA engineers
Usability	Real users can complete key tasks without help	Before launching a major new journey	Product design team
Exploratory	Unscripted, charter-based testing for risks scripts miss	Each new feature before release; one session per squad per sprint (illustrative)	QA engineers and developers

7. Automation approach

What we automate	Stable, repeated checks: smoke tests, regression of P1 and P2 journeys, API contracts, and cross-browser and device checks of checkout.
What stays manual	New features in their first sprint, exploratory testing, usability, visual design review and one-off data migrations.
Test pyramid (guidance)	Most checks live at unit level. API tests cover business rules. UI end-to-end tests cover only the critical customer journeys. No fixed ratios.
Tools and frameworks	Unit test framework native to each codebase; Katalon Studio for web, mobile and API automation; Katalon TestOps for test management, scheduling and reporting; a cloud browser and real-device grid.
Where automated tests run	CI pipeline: smoke suite on every merge, full regression nightly and on each release candidate.
Flaky test rule	A test that fails twice in a week without a code change is quarantined, ticketed, and fixed or deleted within one sprint (illustrative rule).

8. Default entry and exit criteria

Stage	Entry criteria (default)	Exit criteria (default)
Story testing	Acceptance criteria agreed; build deployed to the QA environment; unit tests pass	All acceptance criteria tested; no open critical or high defects on the story; agreed automated checks added
System testing	Feature-complete build in staging; smoke suite passes; test data loaded	All P1 test cases run and passed; P2 pass rate at least 95% (illustrative); every open defect triaged with an owner
Release (go or no-go)	System testing exit met; performance and security checks for the release done	No open critical or high defects; open medium defects accepted by product; rollback tested; QA lead and product manager sign off

Each release test plan can tighten these gates or add its own. Full entry and exit criteria with examples: katalon.com/resources-center/blog/test-plan

9. Test environments and test data

Environment	Purpose	Test data	Managed by
Local	Developer testing and unit tests	Seeded fake data	Developers
CI	Unit, API and smoke suites on each merge	Generated per run and removed after	Platform team
QA	Story and integration testing	Synthetic data set, refreshed weekly	QA engineers
Staging	System, regression, performance and acceptance testing	Production-like configuration; masked or synthetic data only; payment provider sandbox	Platform team
Production	Post-deploy smoke checks and monitoring	Real data; test accounts flagged and kept out of business reports	Platform team and on-call

Browser and device coverage	Set each quarter from analytics: the latest two versions of the major desktop and mobile browsers, and the iOS and Android versions that cover most of our traffic. The current list lives in each test plan.
Test data rules	No real customer data outside production. Personal data is masked or synthetic. Test cards come from the payment provider's sandbox.

10. Defect management

Severity	Definition	Example
Critical	Blocks a core journey or puts money or data at risk, with no workaround	Customers can't complete payment
High	A major feature is broken and the workaround is hard	Promo codes don't apply in the iOS app
Medium	A feature is partly broken and a workaround exists	Sort by price ignores sale prices
Low	Cosmetic or minor issue	Misaligned icon in the footer

Priority	Meaning	Fix by
P1	Fix now	Hotfix, or before the current release ships
P2	Fix soon	Within the current or next sprint
P3	Fix when capacity allows	Backlog, reviewed monthly

Triage	Daily during release hardening, twice a week otherwise. The QA lead, an engineering lead and the product manager set severity, priority and owner.
Defect flow	New > Triaged > In progress > Ready for retest > Closed, or Reopened. Won't fix and Duplicate need a written reason.
Defect record	Steps to reproduce, expected and actual result, environment, build, screenshots or logs, and links to the failed test and the requirement.

11. Tools

Purpose	Tool	Owner
Test management and reporting	Katalon TestOps	QA lead
Test automation (web, mobile, API)	Katalon Studio	QA engineers
Unit testing	Framework native to each codebase	Developers
CI/CD	Company CI server	Platform team
Defect tracking	Company issue tracker	Each squad
Performance testing	Open-source load testing tool	Platform team
Security testing	Code and dependency scanners in CI; external penetration testers	Security lead
Accessibility testing	Automated accessibility checker; VoiceOver, TalkBack and NVDA	QA engineers
Browser and device coverage	Cloud browser and real-device grid	QA engineers

12. Roles and responsibilities

Role	Responsibilities
Head of QA	Owns this strategy; sets quality gates; reports quality to leadership; approves exceptions to exit criteria
QA engineer	Writes and runs tests; builds and maintains automation; joins refinement; leads exploratory sessions; triages failures
Developer	Writes unit and API tests with the code; fixes defects; keeps the pipeline green
Product manager	Agrees acceptance criteria; runs acceptance testing; accepts or rejects open defects at release
Engineering manager	Makes time for testing and defect fixing in each sprint; reviews quality metrics
Security lead	Owns security testing and penetration tests; signs off on security findings
Platform team	Runs CI and test environments; owns performance testing and monitoring

13. Metrics and reporting

Metric	Why we track it	Reviewed
Requirement coverage (acceptance criteria with a passing test)	Shows untested scope before release	Each release
Test pass rate by suite	Shows release readiness	Daily during hardening
Defect escape rate	Shows how well testing catches issues before customers	Monthly
Open defects by severity and age	Shows release risk and backlog health	Weekly
Automated suite run time and flaky rate	Shows feedback speed and trust in automation	Weekly
Time to resolve critical defects	Shows how fast the team responds	Monthly

Report	Audience	Cadence	Where
Test execution dashboard	Squads and QA	Live	Katalon TestOps
Release test summary report	Product, engineering, leadership	Each release	Linked from the release notes
Quality review	Engineering leadership	Quarterly	Review meeting

14. Risks and mitigations

Risk	Likelihood	Impact	Mitigation	Owner
Payment provider outage or unstable sandbox	Medium	High	Mock the provider in CI; run a live sandbox check nightly and alert on failure	Platform team
Flaky end-to-end tests hide real failures	High	Medium	Quarantine rule (section 7); flaky rate on the weekly dashboard; a named owner per suite	QA lead
Peak-season traffic exceeds capacity	Medium	High	Load test against the forecast peak before seasonal releases; check autoscaling	Platform team
A new mobile OS version breaks the app	Medium	Medium	Run regression on OS betas; update the device list each quarter	Mobile lead
Real customer data reaches test environments	Low	High	Masking pipeline; quarterly data audit; no production copies	Security lead
Tight deadlines squeeze testing	High	Medium	Risk-based scope cuts agreed with product; exit gates only waived with Head of QA sign-off	Head of QA

15. Review and approval

Review cadence	Every six months.
Triggers for an early review	A new platform or architecture change; new regulation; a serious production incident; exit gates missed on two releases in a row.
Change process	Propose changes as comments on this document. The Head of QA updates the version and change log, and reviewers re-approve.

Role	Name	Decision	Date
Head of QA	[Name]	Approved	2026-11-06
VP of Engineering	[Name]	Approved	2026-11-06
Product director	[Name]	Approved	2026-11-06
Security lead	[Name]	Approved	2026-11-06

More on writing a test strategy, with steps and examples: katalon.com/resources-center/blog/test-strategy

Disclaimer: this template is provided by Katalon for reference only. Adapt it to your product. Katalon makes no warranties about its completeness or suitability for your use.