User Acceptance Test Strategy

A template to adapt for your organization

Acceptance testing asks a different question from every other level: not "does it work as specified" but "does the business agree this is right". That means it is run by people who do the job, against realistic data, and it is the one level where a disagreement is a useful outcome rather than a defect.

> ***How to use this template.*** *Everything in italics is guidance for you and should be deleted once you have replaced it. Square brackets mark a value to fill in. Work top to bottom the first time; after that, treat sections 3 and 4 as living documents and revisit them every quarter. Keep it short. A strategy nobody reads governs nothing.*

1. Purpose and boundaries

UAT is not a second system test. If your acceptance phase is mostly business users re-running functional checks, you are paying senior people to do regression testing.

UAT is	UAT is not
Confirmation that the change does the job it was asked to do	A hunt for functional defects that testing should already have found
Run by people who do the work	Run by the test team on the business's behalf
Against realistic data and realistic volumes	Against a handful of tidy happy-path records
A decision point — accept, accept with conditions, or reject	A phase that ends when time runs out

> **With Functionize Studio:** the regression layer is automated and green **before** UAT starts, so acceptance testers spend their time on judgment rather than on finding things a machine should have found. That shift is most of the value of automating regression.

2. Entry criteria

> *Be strict. Starting UAT on an unstable build wastes the scarcest people in the organization and poisons their view of the release.*

- All higher-severity defects from system testing are closed or accepted.
- The automated regression suite is green on the UAT build.
- The UAT environment is stable and refreshed with realistic data.
- Acceptance criteria are written down for every item in scope.
- Named business participants are confirmed and have time allocated.

UAT environment

Data refreshed on

Build / version under test

Regression suite result and date

3. Participants and time

> *The most common cause of a failed UAT is participants who were volunteered rather than allocated.*

Business area	Named participant	Days allocated	Backfilled by
[Finance]	[ ]	[ ]	[ ]
[Operations]			

4. Scenarios

> *Write scenarios in the language of the job, not the system. "Process a refund for a customer who paid by card and has already returned the item" — not "test the refund screen".*

#	Scenario	Business area	Acceptance criteria	Result	Notes
1	[ ]	[ ]	[ ]	[pass/fail]	
2					
3					

Include at least one scenario per critical journey from the master strategy, and at least one deliberately awkward case per business area — the exception people actually meet on a Tuesday afternoon.

5. Defects raised during UAT

> *Separate these two. Conflating them is how a release slips for the wrong reason.*

Type	Meaning	Route
Defect	It does not do what was agreed	Fix before release
Change request	It does what was agreed, and what was agreed is wrong	Backlog — does not block this release unless it is a critical misunderstanding

ID	Scenario	Defect or change?	Severity	Decision
[ ]				

6. Exit criteria and sign-off

- Every in-scope scenario has been executed and has a result.
- No open severity 1 or 2 defects.
- Open lower-severity defects are accepted in writing, with owners.
- Change requests are logged and triaged, not silently absorbed.

Decision	Meaning
Accept	Release proceeds
Accept with conditions	Release proceeds; listed items are fixed by an agreed date
Reject	Release does not proceed; re-entry criteria stated

Decision

Business owner / date

Conditions, if any

Re-test required by

7. Feeding UAT back into automation

> *This section is what stops UAT being the same effort every release.*

After every UAT cycle, review the scenarios and ask which of them you never want to run by hand again.

UAT scenario	Ran by hand for the last [n] releases	Deterministic?	Automate for next time?
[ ]			

A scenario that has been executed by hand three releases running, with the same steps and the same expected result, is regression testing wearing a UAT badge. Move it.

With Functionize Studio: a business user can read a plain-language test and confirm it tests what they asked for — so the scenario they signed off can become the automated test they still recognize, rather than code they have to take on trust.