Requirements to Risk-Based Test Design

Lesson 4 · Test what matters first

Today’s outcome

- Turn a story into observable acceptance criteria
- Identify state, boundary, role, and failure cases
- Rank test work by harm and risk
- Use AI for ideas, not final priorities

Vague request

> “A concierge can reserve an allocation after confirming a deposit, and the customer receives an update.”

What could go wrong?

Observable acceptance criteria

Good: “A permitted concierge sees one confirmed allocation after a successful synthetic deposit, and an opted-in customer has one queued update.”

Not enough: “The allocation flow works.”

Risk lenses

- Customer trust and experience
- Money and commitments
- Security and tenant isolation
- Privacy and consent
- Operational recovery
- Accessibility and device access

AI is a brainstorming partner

Ask it for cases. Then check for missing:

- duplicate action
- wrong role
- delay/failure/retry
- boundary values
- stale state
- accessible/mobile experience

State transitions reveal bugs

Inquiry → allocation candidate → deposit pending → reserved → notified

Test allowed moves, forbidden moves, and recovery from each state.

Lab: Allocation-to-Delivery Test Charter

Deliver: risks, prioritized tests, evidence needs, synthetic data, and automation decision.

Every team must include security, accessibility, and integration coverage.

Risk poker

Defend your top three tests.

The goal is not the longest checklist. It is the best use of limited time to protect people and the business.

From charter to release evidence

Charter → manual exploration → automated checks → evidence packet → release recommendation

The same intent should survive the whole lifecycle.

Exit ticket

Write one negative acceptance criterion for a workflow you know.