Experiment Design (A/B Testing)

A way to know if a change actually works, instead of guessing. In an A/B test you randomly split users into two groups: group A sees the current version, group B sees the change. You then compare results. Because the split is random, the only real difference is your change, so if B does better (and it's not just luck), you've found a genuine improvement. It's the gold standard for proving cause and effect.

Best for: Testing changes · Proving cause · Conversion lifts · Data-driven decisions

01 — AT A GLANCE

Start with one hypothesis, split users randomly into A and B, run both at once, then measure which truly won.

1 — Form a hypothesis
Decide what to change and what you expect, e.g. 'a clearer button will lift sign-ups'.

▼

2 — Split randomly
Randomly place users in Group A (current) and Group B (the change), so the groups are comparable.

▼

3 — Run A vs B
Show each group its version at the same time, changing only the one thing you're testing.

▼

4 — Measure & decide
Compare results and check the difference is real (statistically significant), not just luck, then act.

02 — THE CORE COMPONENTS

A trustworthy A/B test rests on a few essentials.

Control vs Treatment	Group A (control) keeps the current experience; Group B (treatment) gets the one change. Comparing them isolates the change's effect.
Randomisation	Assigning users randomly makes the two groups alike in every other way, so any difference in results comes from your change, not from who's in each group.
One Variable	Change just one thing at a time. If you change several, you won't know which one caused the result.
Significance	A statistical test checks whether the difference is big enough to be real, not random noise. Run it to completion; don't stop the moment it looks good.

Randomness is the secret ingredient

Why split randomly? Because it makes the two groups nearly identical in every way except your change, age, device, mood, everything evens out. That's what lets you claim the change caused the result, not some hidden difference. Without randomisation, you only have a correlation; with it, you have proof.

03 — WHEN TO USE IT & WHEN NOT TO

This framework isn't right for every situation. Used at the wrong moment, it can point you in the wrong direction.

✓ USE IT WHEN…	✗ DON'T USE WHEN…
• Testing a specific change before rolling it out • Proving a change actually causes an improvement • Comparing two versions of a page, email or feature • Settling debates with data instead of opinions	• With too little traffic to reach a clear result • When you can't randomly assign users • Testing many things at once in one simple A/B • For big, one-off decisions you can't split-test

04 — IN ACTION

Swift — Does the New Button Really Work?

Swift, a SaaS tool, thinks a clearer sign-up button will boost registrations. Instead of just switching it, they run an A/B test to be sure.

How the framework was applied:

Step 1	**State the hypothesis —** 'A bigger, clearer button will increase sign-ups.'
Step 2	**Split the traffic —** Visitors are randomly shown the old button (A) or the new one (B).
Step 3	**Run both at once —** Each version runs at the same time, with only the button differing.
Step 4	**Measure the result —** Group B signs up 12% more often, and the test shows it's significant.
Step 5	**Roll it out —** Confident the button (not luck) caused the lift, Swift ships it to everyone.

The numbers at a glance

Group A	Group B	Result	Action
Baseline	**+12%**	**Significant**	**Ship it**
old button	new button	not luck	with proof

Why the test beat a guess

Swift could have just swapped the button and hoped. The A/B test split users randomly so the only difference was the button, proving the 12% lift was real, not seasonality or chance. Now Swift ships with evidence, and knows the change actually caused the win.

05 — KEEP IN MIND

Even experienced teams slip up here. Keep these in mind to keep your analysis honest and useful.

1. Stopping the test early the moment results look good.
2. Changing several things at once, so the cause is unclear.
3. Running with too little traffic to trust the result.
4. Ignoring statistical significance and chasing noise.
5. Skipping randomisation, which breaks the whole logic.