RESEARCH

The Aha-Moment Audit — A Prompt to Refactor Your App's Onboarding

Eight patterns from the best onboarding flows on the internet, distilled into a copy-paste prompt that audits and refactors yours in three phases.

By Oliver Signorini · 2026-05-08

RESEARCH

The Aha-Moment Audit — A Prompt to Refactor Your App's Onboarding

Eight patterns from the best onboarding flows on the internet, distilled into a copy-paste prompt that audits and refactors yours in three phases.

The thing nobody tells you about onboarding

The conventional wisdom says **keep onboarding short**. When you actually look at the apps that win — across categories, across geographies, across subscription models — the conventional wisdom turns out to be wrong.

Some of the most successful onboarding flows on the internet run *dozens* of screens before they ask you to sign up. Duolingo lets you finish a whole lesson before account creation. Some of the shortest flows on record belong to AI products that don't onboard at all — they just let you start a chat.

Length is not the variable that matters.

The variable that matters is **value velocity** — how fast a new user reaches the *aha moment*: the first time the product proves its value to them. For Airbnb, that's their first booking. For Netflix, finding a show. For Mobbin, saving a screen they love.

Every great onboarding flow follows the same shape:

> *Sign up → Set up → Aha moment*

The art is making that path as short — *or as delightful* — as possible.

This guide is the synthesis: the 8 patterns that show up across every great flow, plus a three-phase prompt you can paste into your codebase to audit and refactor your own.

The 8 patterns that recur in great onboarding

1. Sell the outcome, not the features

The first screen of a great onboarding doesn't list features. It shows the product *working*.

- **Timo** — welcome screen is the product running on mobile and desktop.
- **Alma** — lets you try the core experience before you sign up.
- **Superhuman** — turns the signup screen itself into a pitch with customer logos as social proof.

If your first screen is a form, you're asking the user to invest before you've earned it.

2. Add a human touch at the aha moment

The moment the user feels the product work is the most generous moment in the relationship. Spend it.

- **Airbnb** — plays a CEO video after you list your first space.
- **Basecamp** — drops a personal note from the founder after account creation.
- **Oneyear** — handwritten founder signature in the welcome flow.

Cost: small. Effect: makes the product feel made *by humans, on purpose*.

3. Personalize during onboarding — and *show what the answers unlocked*

Most apps don't personalize during onboarding. AI apps almost never do — they let the product learn instead. The ones that *do* personalize and do it well don't just *collect* answers; they render a payoff page that visibly uses them.

- **Endel** — answer 6 questions, see a custom soundscape concept rendered before you sign up.
- **BitePal** — answer the quiz, see a graph of how fast you'll hit your goal.
- **Grammarly** — tailored pricing plans based on quiz answers → +20% upgrades.

If you ask N questions, you owe the user a screen that proves you listened.

4. Allow multi-intent

Headspace let users pick more than one goal in onboarding. **+10% free-trial conversion.**

Most quizzes force a single answer. Most users have more than one reason they're here.

5. Make the copy conversational

Dollar Shave Club rewrote quiz copy from clinical to conversational. **+5% subscriptions.** No new questions, no new flow — just the words.

6. Make long flows not feel long

Duolingo's onboarding is famously long, and the user has already finished a full lesson by the time they're asked to create an account. Bumble animates verification screens that nobody else animates. BitePal lets you name a virtual raccoon.

The choice isn't "long or short". It's "long but delightful, or short and forgettable".

If your flow has to be long, lean *into* delight. Animate the loading states. Name the characters. Render satisfying micro-completions.

7. Replace pop-up tours with checklists

Mural swapped pop-up tours for a 6-step checklist. **+10% week-1 retention.**

Pop-ups vanish on dismissal. Checklists persist. They're a permanent invitation to discover what the product does, on the user's schedule.

8. Pre-prompt before native permission asks

Native permission dialogs (notifications, contacts, location) are one-shot. If the user taps "Don't allow", you're done — and most apps fire them cold.

The ones that don't:

- **Brilliant** — explains *why* notifications matter before triggering the OS dialog.
- **Calm** — previews the actual notification you'll receive.

Significantly higher accept rates.

Other findings worth knowing

- **Paywalls inside onboarding are common, and the best ones are a delight surface.** Focus Flight's one-time offer is shaped like a flight ticket and "prints out" with a phone vibration. The paywall doesn't have to feel like a tollbooth.
- **Splitting signup forms across multiple screens lifts conversion ~15%** (Howse case study). Counter to "minimize fields" orthodoxy. Friction redistributed isn't friction added.

- **Web onboarding tends to be shorter than iOS.** Mobile carries permission and paywall overhead the web doesn't.
- **Cultural variation matters.** Eastern markets tolerate denser interfaces that Western users read as cluttered. Don't blind-copy reference apps.
- **Sometimes the right onboarding is none.** For products where the first action *is* the value (browsing inspiration, sending a chat prompt), the best experience is no onboarding.

How to apply this to your app

The temptation after reading research like this is to redesign in your head and not in your codebase. Here's a prompt that forces the audit into the actual repo.

It runs in three phases with explicit stop points so the agent waits for your confirmation between each one. Phase 1 audits what exists. Phase 2 inventories the data your app *actually* needs (vs. what it *currently asks for*). Phase 3 proposes a refactor scaffolded on `Sign up → Set up → Aha moment`.

Paste it into a fresh Claude or Cursor session inside your app's repo:

```
# The Aha-Moment Audit

You are a product designer + senior engineer pair. Your job is to aud
existing onboarding flow in this repo, inventory the data the app act
needs from a new user, and propose a refactored flow that maximises
**value velocity** — the speed at which a new user reaches the produc
"aha moment" (the first time the product proves its value to them).

You will work in three phases. **Stop after each phase and wait for m
confirmation before moving on.**

---

## Phase 1 — Deep dive the existing onboarding

Trace the current flow end-to-end. For each step, identify:

1. **The screen / route / component** (file path + line refs).
2. **What is asked of the user** (fields, choices, permissions, payme
3. **Why it exists** — is this data required for core functionality,
   personalization, for compliance, or is it speculative ("nice to ha
4. **Friction cost** — taps, decisions, typing, context switches.
5. **Where the user currently hits the aha moment** — the first momen
   product delivers real value. If you can't identify one, say so exp
   and propose three candidates.

Output a single table:

| # | Screen | Asks for | Required for | Friction | Position vs aha |
```

```
Then summarise:
- Total screen count
- Total required fields vs optional fields actually asked as required
- Where the aha moment sits relative to signup (before / at / after, a
  by how many screens)
- The 2-3 biggest **value-velocity blockers** — steps that delay aha
  without strictly requiring it

**Stop. Wait for my confirmation before Phase 2.**

---

## Phase 2 — Data inventory

Independent of the current flow, list **every piece of data the app
genuinely needs** about a user to function. For each, classify as:

- **Critical** — app cannot work without it (e.g. account email, auth
- **Functional** — feature-specific, only needed when that feature is
  (e.g. payment method only at first paid action)
- **Personalization** — improves UX but the app works without it
  (e.g. preferred theme, goal selection)
- **Speculative** — currently collected but the app does nothing with

For each non-critical item, answer: **can we collect this lazily**
(later, in-context, when the user actually hits the relevant feature)
**rather than upfront?**

Output a single table:

| Field | Class | Currently collected at | Can defer to | Why |
```

```
Then list every **speculative** field as a deletion candidate, with
evidence (grep results showing the field is unused, or only logged).

**Stop. Wait for my confirmation before Phase 3.**

---

## Phase 3 — Refactor proposal

Using the eight onboarding patterns above as your reference set, propo
refactored flow. Use this scaffold:

`Sign up → Set up → Aha moment`

For each step in your proposal, specify:

1. **What the user sees** (rough copy + UI intent — not pixel design)
2. **What data is collected** (only Critical + Personalization-with-pa
3. **The aha-moment-shortening rationale** — why this step earns its p
4. **Files to change** (exact paths) + new files to create.
5. **A/B-testable hypothesis** if relevant ("splitting signup into 2
   screens should lift completion by ~5-15% based on Howse case study

Apply these specific patterns where they fit:

- **Sell the outcome before asking for input.** First screen shows the
  product working, not a form.
- **Try-before-signup** if the core experience can be rendered withou
  account (massive for AI / generative features).
- **Personalize and show payoff** — if you ask N questions, render a
```

```
  result page that visibly uses those answers before any paywall or
  signup ask.
- **Multi-intent** — if you ask "what's your goal?" let users pick
  multiple.
- **Lazy permission asks** — never trigger native permission dialogs
  precede them with a custom "here's why" screen.
- **Checklist > pop-up tour** for post-signup feature discovery; pers
  after dismissal.
- **Conversational microcopy** in any quiz / form — not clinical.
- **Make long feel short** — animation on loading states, named
  characters, satisfying completion micro-moments. If your flow is
  necessarily long, lean into delight rather than trimming.

End with:

- **Migration plan** — how to ship this without breaking existing use
  (feature flag? cohort cutover? new + old flow side by side?).
- **Metrics to instrument** — completion rate per step, time-to-aha,
  D7 retention, conversion to first paid action (if applicable).
- **Three open questions** you still have for me before implementatio

Do not write implementation code in this phase. The output is a writt
refactor spec I can review, edit, and then hand back to you for build
```

What to do next

1. Pick the app where onboarding hurts most — the one with the worst activation rate, or the one you've been meaning to fix.
2. Open a fresh agent session inside that repo. Paste the prompt above.

3. Sit with the Phase 1 output. Most of the value is in seeing the flow tabulated honestly for the first time. Half the things you currently ask users won't survive the "why does it exist?" column.
4. Don't skip the stop points. The temptation is to barrel into Phase 3. The discipline is to fully understand the audit and the data inventory before redesigning.

The goal isn't shorter onboarding. The goal is **fewer screens between the user and the moment your product proves itself**.

That's the lever.

Synthesised from Mobbin's published research on onboarding flows and the public A/B-test case studies referenced in their analysis. Specific company A/B outcomes (Headspace, Dollar Shave Club, Grammarly, Mural, Howse) are from those publicly available case studies, not original research.

Built by Oliver Signorini

Find more at oliversignorini.com