The Boring App Build Playbook

Reverse-Engineer a $80K/Month App With Claude Code — In a Weekend

01

Pick the Boring Niche

The apps quietly clearing $60,000–$80,000 a month are not the exciting ones. Receipt scanning is the clearest example: a single-purpose tool that removes manual bookkeeping for small businesses. It works because the pain is recurring, it costs the business real money every month, and it is a necessity rather than a nice-to-have — which is exactly why the churn is low and the subscription sticks. Open the App Store, go to the Business or Finance category, and look past the top 10 for tools that do one unglamorous job for a company rather than a consumer.

> 💡 **QUICK TIPS**
>
> - Green flags: paid subscription (not ad-supported), a business buyer, a task the user repeats weekly, and boring screenshots.
> - Red flags: social features, a feed, anything that needs network effects or virality to be useful.
> - Sanity check the money before you build — cross-reference the app on Sensor Tower or Appfigures rather than trusting a founder's screenshot.

02 Screenshot the Proven App

You are not starting from a blank page — you are starting from something the market already pays for. Install the top-ranked app in your niche and capture every single screen: onboarding, the empty state, the main capture flow, the list view, the detail view, settings, and every paywall you can trigger. Twenty to thirty screenshots is normal. Then drag all of them into Claude Code in one message and paste this prompt.

PASTE THIS WITH YOUR SCREENSHOTS

```
I'm attaching screenshots of an App Store app called [APP NAME] in the [receipt scanning / expense tracking] category. Act as a senior product engineer. From these screens only — no guessing at features you can't see — produce:

1. A screen-by-screen inventory: every screen, its purpose, and the UI components on it.
2. The inferred data model: entities, fields, relationships, and which screen reads or writes each one.
3. The core user flows, written as numbered steps from first launch to the app's main value moment.
4. The monetisation model you can see evidence for (paywall placement, free limits, upgrade prompts).
5. The 20% of features that deliver 80% of the value — what I must build for v1, and what I can cut.
6. A gap list: what this app does badly or omits, based on what's visible plus the negative reviews I'm pasting below.

Output as a PRD I can hand to a build agent. Flag anything you inferred rather than observed.
```

💡 QUICK TIPS

- Capture the paywall deliberately — hit the free limit on purpose so you can see where they charge and how much.
- Include the App Store listing screenshots too; they show you the features the company thinks are worth selling.
- The line "flag anything you inferred rather than observed" is the important one — it stops Claude Code inventing features that were never on screen.

03 What Claude Code Came Back With

Here is the actual shape of the output for a receipt tracker, so you know what a good answer looks like. **Five core screens:** camera capture, receipt list, receipt detail with editable extracted fields, reports and export, and settings and billing. **The data model:** User, Receipt, LineItem, Category, Export and Subscription — with an OCR confidence score stored per extracted field, which is the detail most people miss and the thing that makes the edit screen work. **Monetisation:** the paywall sits at the monthly scan limit, not at signup — users get hooked on the capture flow first. **The 80/20 cut:** capture, extract, edit, export. Everything else is v2. **The gap list from reviews:** the OCR misreads totals on faded receipts, there is no multi-currency support, there is no clean export into accounting software, and there is no team or multi-user mode.

> 💡 **QUICK TIPS**
>
> - If Claude Code hands you 30 features, push back once: "cut this to the four screens that deliver the core job." It will.
> - Save the PRD to your repo as PRD.md before you build — it becomes the reference the build agent keeps returning to.
> - The gap list is the most valuable section. That is your product wedge, written for you by the competitor's own customers.

---

04 Feed the Bad Reviews Back In

The one-star and two-star reviews are the market telling you exactly what it will pay to have fixed. Sort the App Store reviews by most critical, copy 30–50 of them, and paste them straight back into the same Claude Code session so it can re-rank the roadmap against real complaints instead of your assumptions.

PASTE THIS WITH THE REVIEWS

```
Here are 40 one-star and two-star reviews for [APP NAME], pasted below. Cluster them
into the underlying complaints, count how many reviews hit each cluster, and rank them
by how often they appear and how badly they block the core job. Then re-order the v1
roadmap from the PRD so the biggest cluster ships as a headline feature, not a backlog
item. Tell me which PRD features I can now cut.
```

> 💡 **QUICK TIPS**
>
> - Do the same for the two apps ranked below the leader — shared complaints across all three are category-wide gaps, and those are the safest bets.
> - Ship the single biggest complaint as your headline feature, not as a backlog item. That is your entire differentiation.
> - Ignore complaints about price. Those are almost never the real reason people churn.

05

The Build Prompt

Do not ask for the whole app in one shot — phase it, and make it prove each phase before moving on. This prompt is the one that turns the PRD into working software, and the two rules at the bottom are what keep you out of the mock-data trap where everything looks finished and nothing actually runs.

PASTE THIS ONCE THE PRD EXISTS

```
Build v1 of this product from the PRD above. Stack: React Native / Expo for mobile,
Next.js for the web dashboard, Supabase for auth, database and storage. Work in this
order and stop for my review after each phase:

Phase 1 — Scaffold the repo, the database schema and auth. Show me the migration files
before you run them.
Phase 2 — The capture flow end to end: camera to upload to OCR to editable extracted
fields to saved receipt. Prove it by running it and showing me a real saved record.
Phase 3 — The receipt list, detail and category screens.
Phase 4 — The web dashboard reading the same data, plus CSV export.
Phase 5 — The paywall and subscription limits.

Rules: no mock data in committed code. After every phase, run the app and confirm the
flow works before telling me it's done. Write the smallest diff that fully solves each
step.
```

💡 QUICK TIPS

- Run **/clear** between phases and point Claude Code back at PRD.md — long sessions get vague and start forgetting your constraints.
- "Show me the migration files before you run them" is worth keeping. Schema mistakes are the expensive ones to undo.
- If a phase comes back done in suspiciously few steps, ask it to run the flow and show you the output. It is usually stubbed.

06

Ship It in a Weekend

Two days is genuinely enough for v1 if you refuse to widen the scope. **Saturday:** Phases 1 and 2 — schema, auth, and the capture flow working end to end on your own phone with a real receipt. **Sunday:** Phases 3 to 5 — the list and detail screens, the web dashboard, CSV export, and the paywall. Cut anything that is not capture, extract, edit, export. No onboarding tour, no dark mode, no settings you do not need yet. Then watch one number: the percentage of users whose first scan makes it all the way to a saved receipt. If that number is low, nothing else you build matters.

> 💡 **QUICK TIPS**
>
> - Put it in front of one real small-business owner before you polish anything. Their first 60 seconds will tell you more than a week of tweaking.
> - Price it against the hours it saves, not against other apps. Bookkeeping time is expensive.
> - Charge from day one. A free tier with no paid tier teaches you nothing about whether the problem is worth money.

Ready to go deeper?

This playbook is the short version. Inside the hub you get the full build walkthroughs, the prompt library, and the systems I use to ship apps like this in a weekend.

Join the Build →

https://hub.digicuratoragency.com/join