

2ndJ: Design Plan

A rough plan for the worker and reviewer flows in the 2ndJ prototype.

Prototype: <https://2ndj-prototype.vercel.app/>

Hi Andrea,

Here's a rough plan for both sides. I also built a small working prototype (linked above) to test the flows in practice. Think of it as one way the plan could look, not the plan itself.

How I'd approach the worker side

Before someone taps anything, the nearby list needs to tell them four things: what the task is, where it is, what it pays, and what proof it needs. So I'd design the task card to show all four at a glance: category, area and distance, price, and a small tag for the proof needed (how many photos, GPS or not). That way people can decide if it's worth the trip without opening it.

Once they accept a task, I wouldn't give them a form. I'd guide them through the proof one step at a time: one short instruction per step ("stand at the front door, take one photo"), one action, then on to the next. No text fields anywhere. Every proof step should be something simple and predictable, like a photo or a location check, because as soon as a worker has to think about what to write, the job stops being quick. The last screen should be the payout, not a receipt. If they get paid on proof, then sending the proof and getting paid should happen on the same screen.

How I'd approach the reviewer side

The dashboard should help reviewers sort work, not just show it. I'd split inspections into two groups: stuff that still needs a decision and stuff that's already done. "Needs review" goes first, with a filter for status (needs review or flagged). Approved work goes in its own section that's collapsed by default. It's still easy to get to, but it stays out of the way so reviewers focus on what actually needs them.

When a reviewer opens an item, it should show everything they need to trust it without going to check in person: photos, a timestamp, GPS location, and who did the work. Then they get two actions, each with the right amount of friction. Approve should ask "are you sure?" first, because approving also pays the worker, and a misclick shouldn't be able to do that. Flag should require a short reason, because the flag goes back to the worker, and just saying "no" doesn't tell them what to fix. Moving an approved item back to review, on the other hand, should be one tap with no confirmation. It's just fixing a mistake, not paying anyone, so slowing it down would only get in the

reviewer's way.

Order I'd tackle it in

- 1 The worker's task card and that first "what am I looking at" moment. It's the thing people see most in the whole product, so it needs to be right before anything else is worth building.
- 2 The full worker flow (nearby, task, accept, capture, paid) as one real path someone can actually go through, before the dashboard even exists.
- 3 The dashboard's core loop: a list of submitted proof and one popup to review it, approve it, or flag it. This is the part that actually builds trust. Everything else just supports it.
- 4 A friction pass once both flows work and can be tested one after the other: add the confirm step on Approve, move finished work out of the main list, and put any action that changes a live listing (not just reviews it) behind a menu instead of a click on the row. Looking at proof by accident is harmless, but editing or deleting a listing by accident isn't.
- 5 Everything else the dashboard will need later, like managing listings and seeing totals over time, I'd build after that core loop is solid, not at the same time.

The parts I think are genuinely hard

The main tension in this product is that everything that makes the client trust the proof more also makes the worker's job slower, and you rarely get one without the other. A single photo is quick to take and decent proof. A second angle, a GPS pin, or a specific way of framing the shot all give the reviewer more confidence, but each one means more time standing on someone's doorstep. I don't think there's one right answer. It depends on what kind of task it is and what's being checked. So I'd rather make it a setting per task type (how many photos, whether location is required) than build it into the flow for everyone.

The second hard part is getting the friction right on the review side, and making it different for each action on purpose. Approve, Flag, and Reopen all change an item's status, but they don't carry the same risk. Approve pays someone, so it shouldn't happen by accident. Flag sends feedback to a real person, so it needs a reason. Reopen just undoes a decision, so it should be easy. Making each one feel different in how much it slows you down, instead of using the same button in three colors, is a big part of what makes the product feel trustworthy day to day. It's not about how it looks. It's about the app making you stop and think exactly where a mistake would matter.

The prototype linked above is where I tried all of this out end to end, so you can see the flows in action instead of just reading about them. It's a rough version of the plan, not a finished product.