

Count first, compare second: a design case of cash-session reconciliation in point-of-sale software for small businesses in Nepal

Bibhushan Saakha · BIC Technology (Tigg), Kathmandu, Nepal

bibhushansaakha@gmail.com

ABSTRACT

Small shops and restaurants in Nepal often cannot say whether the cash in a drawer is right, how much is missing, or why. This design case documents how a point-of-sale product used in Nepal moved from sessions that recorded only a time and a person to an accountable cash drawer. The author, the product's sole designer and its web frontend engineer, describes the decisions and their rationale: modelling the drawer as a ledger whose expected amount can be re-added by eye; a blind, two-phase close in which the cashier counts before the expected amount is shown, to reduce anchoring toward it; treating a blank count as unknown and zero as a known empty drawer; requiring a note only when the drawer is not balanced; a per-location choice between strict and easy modes; a single gate at the payment page; and a per-entry history of corrections. The paper traces iterations, most of which removed something, and derives five transferable principles. It compares the design with patterns documented by other POS products in a 2026 documentation scan. It is not an evaluation: there was no field study and no outcome has been measured. The paper closes with a proposed study combining shift-change observation, discrepancy logging with explicit denominators, and a controlled test of the blind count.

Keywords: point of sale · cash reconciliation · interaction design · anchoring · form design · design case study · small business · Nepal

1 Introduction

A cashier in a small shop or restaurant opens the cash drawer in the morning, takes orders all day, pays a supplier out of the till, accepts cash from a regular customer who is settling last week's credit, and hands the counter over in the evening. At the end of the day the owner has a simple question: is the right amount of money in the drawer? If not, how much is missing, since when, and why?

Point-of-sale (POS) software is usually good at the first half of that day, recording what was sold, and often weaker at the second half, accounting for the physical cash that the sales should have produced. This paper is about a product where, before the work described here, that second half was missing entirely. Tigg is a cloud accounting and POS product made by BIC Technology in Kathmandu, used by retail shops and restaurants in Nepal. Its POS already had *sessions*: a person started a session at a location, sold things, and stopped it. A session recorded a time and a person. It did not record money. There was no opening count, no record of cash taken out or put in without a sale, no closing count, and nothing to compare a count against. When a drawer came up short, the difference had no owner, no timestamp and no explanation.

Between June and July 2026 I designed, and built the web frontend for, a cash layer on top of those sessions. Each session can now carry an opening count, cash movements with reasons, a closing count that is compared with an expected amount, a required explanation when the two disagree, and a visible history of corrections. All of this sits behind a per-location setting, so that shops which do not want the control are not slowed down by it.

The paper's contribution is a documented set of design decisions and the reasoning behind each, written in the *design case* genre: a rich, situated account of a designed artefact, intended to be useful to other designers as precedent (Boling, 2010; Smith, 2010). From those decisions I draw five design principles that I believe transfer to other systems where people count something and a computer holds the expected answer. The paper is explicitly **not an evaluation**. I did not run a field study, a usability test or an experiment, and no outcome has been measured. Section 7 says what that means for the claims made here and sets out a study that could test them.

2 Background and related work

2.1 Cash handling and reconciliation practice

Reconciling a cash drawer is a practice that predates software: count the float at the start of a shift, record every movement of cash, count again at the end, and compare the count with what the records say should be there. In internal-control terms it is a *control activity*, a check that detects errors and irregularities after the fact, which depends on the records being complete and on the people involved being

accountable for their part (COSO, 2013). Two features of that framing matter for design. First, reconciliation is only as good as its inputs: if cash can leave the drawer without a record, the expected figure is wrong and the comparison is meaningless. Second, a control has to be proportionate to the organisation. A single owner-operator who is the only person touching the till has little use for a ceremony designed to separate the duties of several employees.

In HCI, Perry and Ferreira's (2017) study of *moneywork* describes the practical, social and material work people do around digital and physical money. It is a reminder that a cash drawer is not only a number in a database but a physical object handled at a busy counter by a person who may be interrupted, tired or under suspicion.

2.2 Anchoring and confirmation

The central decision in this design, hiding the expected amount until the cashier has counted, rests on a well-established finding. Tversky and Kahneman (1974) showed that numerical estimates are pulled toward an initial value, even an arbitrary one, and that adjustment away from it is typically insufficient. A later review found the effect robust across many domains, including expert judgement (Furnham & Boo, 2011). In an auditing setting close to cash reconciliation, Joyce and Biddle (1981) studied anchoring and adjustment in auditors' probabilistic judgements, an early sign that professionals who check figures are not immune to the reference values put in front of them.

Counting a drawer is not an estimate; there is a true answer. But when a count disagrees with a number on screen, the easiest resolution is to recount until the numbers agree, or to type the target. Nickerson (1998) describes this broader tendency to seek or interpret evidence in ways that confirm an existing expectation. A reconciliation interface that shows the expected figure during counting therefore risks producing counts that are partly copies of the expectation, which defeats the purpose of counting.

2.3 Numeric input and form design

Much of the integrity of a cash feature lives in small input rules. Thimbleby and Cairns (2010) document how number-entry interfaces commonly ignore, truncate or silently transform erroneous input, and argue that such behaviour causes serious errors in safety-critical settings. Their analysis applies directly to the question of what a blank field should mean. In data modelling the distinction between a known value and a missing one is long established: Codd's (1979) extension of the relational model treats a null as "value unknown", distinct from every actual value, including zero. Nielsen's (1994) heuristic of error prevention frames the choice between blocking, warning and silently accepting.

2.4 POS, small businesses and emerging markets

I found little published HCI work on POS reconciliation in small businesses. The closest work concerns micro and small enterprises in developing economies, and HCI for development. Donner and Escobari

(2010) review evidence on mobile phone use by micro and small enterprises and find that benefits accrue mostly through communication and coordination rather than through changes to internal business processes. Toyama (2010) and Dell and Kumar (2016) survey HCI for development and call for designs grounded in the specific conditions of the settings they serve, and for honesty about how much designers know about the people they design for. The Global Findex database documents the spread of digital payments alongside cash in many economies (Demirgüç-Kunt et al., 2022). In Nepal that coexistence is visible at the counter, where QR and wallet payments sit alongside cash, so a reconciliation design has to separate the cash physically in the drawer from everything else.

2.5 Design cases as a research genre

This paper follows the design-case genre as developed in design and learning-design research. Boling (2010) argues that designers learn from rich accounts of specific designs, as precedent, and that such accounts are a legitimate form of design knowledge without generalisable claims. Smith (2010) sets out what makes a design case rigorous: a clear account of the designer's situation and constraints, transparency about sources, attention to decisions and to the alternatives set aside, and honesty about failures. In HCI, design knowledge has been framed as intermediate-level concepts, more abstract than an instance but less than a theory (Höök & Löwgren, 2012), and research through design as producing accounts whose value lies in what they make thinkable rather than in proof (Zimmerman et al., 2007; Gaver, 2012). Schön's (1983) reflective practice, reasoning through a conversation with the situation, describes how most decisions in this case were made. The principles in Section 6 are offered in that spirit: as claims to be tested.

3 Context and method

3.1 Setting

Tigg serves small and medium businesses in Nepal: shops, restaurants and similar counters. Its POS runs as a web application on counter desktops and tablets, and as a mobile app. Amounts are in Nepalese rupees. The default denominations a cashier counts are Rs 1,000, 500, 100, 50, 20, 10, 5, 2 and 1, in descending order, which is the order in which a drawer is usually counted. Less common notes such as Rs 25 and Rs 250 are not in the default list, but a shop can add them, which is why the list is editable rather than fixed. Customers commonly run a tab and pay it later, and cash, QR codes and digital wallets are used side by side. All amounts in this paper and its figures are fictional.

3.2 My role

The request for cash sessions came from sales and went to the CEO, who planned the feature and asked me to ideate and design it. I am the only UI/UX designer at Tigg, and on this feature I was also the web frontend engineer. Table 1 summarises the division of labour.

AREA	WHO
Plan and priority	CEO
Interaction and visual design, web and mobile	Author
Web frontend (the POS web application)	Author
Backend: storage, the expected-cash calculation, history	Backend engineer
Mobile app build	Mobile developer, from the author's designs
Review of the author's frontend work	Senior frontend engineer
Testing	QA

Table 1. Roles on the feature. Teammates are identified by role.

Being both designer and frontend engineer meant I could try a decision in the running product within hours. My first working version ran on 2 June 2026, was connected to the real backend on 8 June, and reached production in the second half of June after QA and user acceptance testing. I continued refining it into mid-July.

3.3 Sources of evidence

The account rests on four kinds of evidence:

1. **Product briefs and conversations** with the CEO during ideation. There was no written requirements document.
2. **Design files:** the web and mobile screens for the feature, marked ready for development.
3. **The shipped behaviour** of the web product, which I built. Interface messages quoted in this paper are the product's own copy.

4. **My own recollection**, recorded as written answers in September 2026, about why particular decisions were made. Where I give a reason that I did not state at the time, I say that it is a retrospective reading.

There was one walkthrough with real equipment: the team ran the flow in the office with actual cash drawers and counter hardware. It was a hardware demonstration, not user research; nobody outside the team took part. I also hear about user problems indirectly, through QA, support and the CEO, and QA's follow-up list during the build was the most concrete feedback I had. Before designing, I looked at the cash-session features of a number of other POS products, most often Zoho's, but I did not record which ones. The competitor comparison in Section 6 is therefore based on a separate documentation scan done in September 2026, while writing this paper, and is labelled as such.

3.4 Method and its limits

This is a retrospective, reflective account by a single practitioner (Schön, 1983), structured to meet Smith's (2010) criteria for a rigorous design case: situation and constraints, decisions and alternatives, iterations and failures, and transparent sources. It can describe what was designed and why, but not whether the design works for the people who use it. Section 7 returns to these limits.

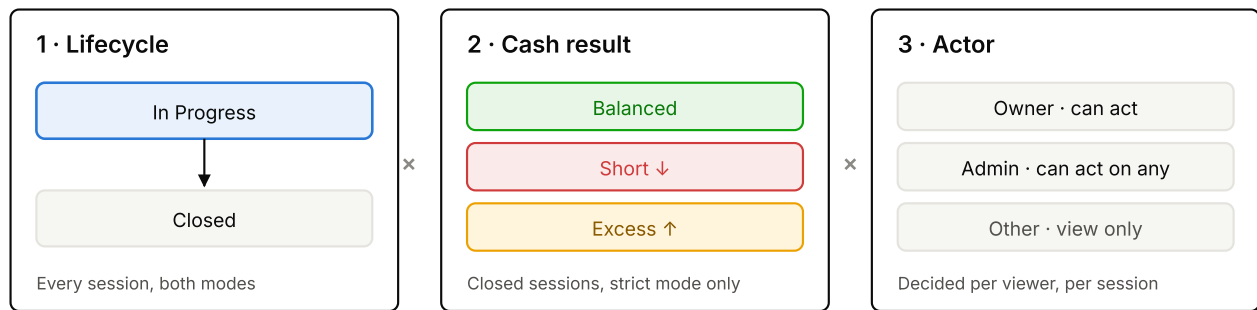
4 The design

4.1 Objects and states

Before drawing screens I separated three properties of a session that are easy to merge:

1. **Lifecycle**: in progress, or closed.
2. **Cash result**: balanced, short or excess. It exists only for closed sessions at locations that verify cash.
3. **Viewer**: the session's owner, an admin, or another member of staff.

It is tempting to let "closed" imply "fine". But a closed session is not necessarily a balanced one, and an owner reviewing last week wants exactly the closed sessions that came up short. Keeping the three dimensions independent made that question a single filter in the session list, where Balanced, Short and Excess are nested under Closed and appear only when the location verifies cash.



In the list filter: Short → closed, and the drawer came up short

The Balanced / Short / Excess options appear nested under Closed, and only when the location is strict.

DIAGRAM Figure 1. Three dimensions of a session that vary independently: lifecycle, cash result and viewer. Keeping them apart is what lets an owner ask for every short session last week in one step. Scroll sideways to see the whole diagram.

With the dimensions separated, the states followed (Figure 2).

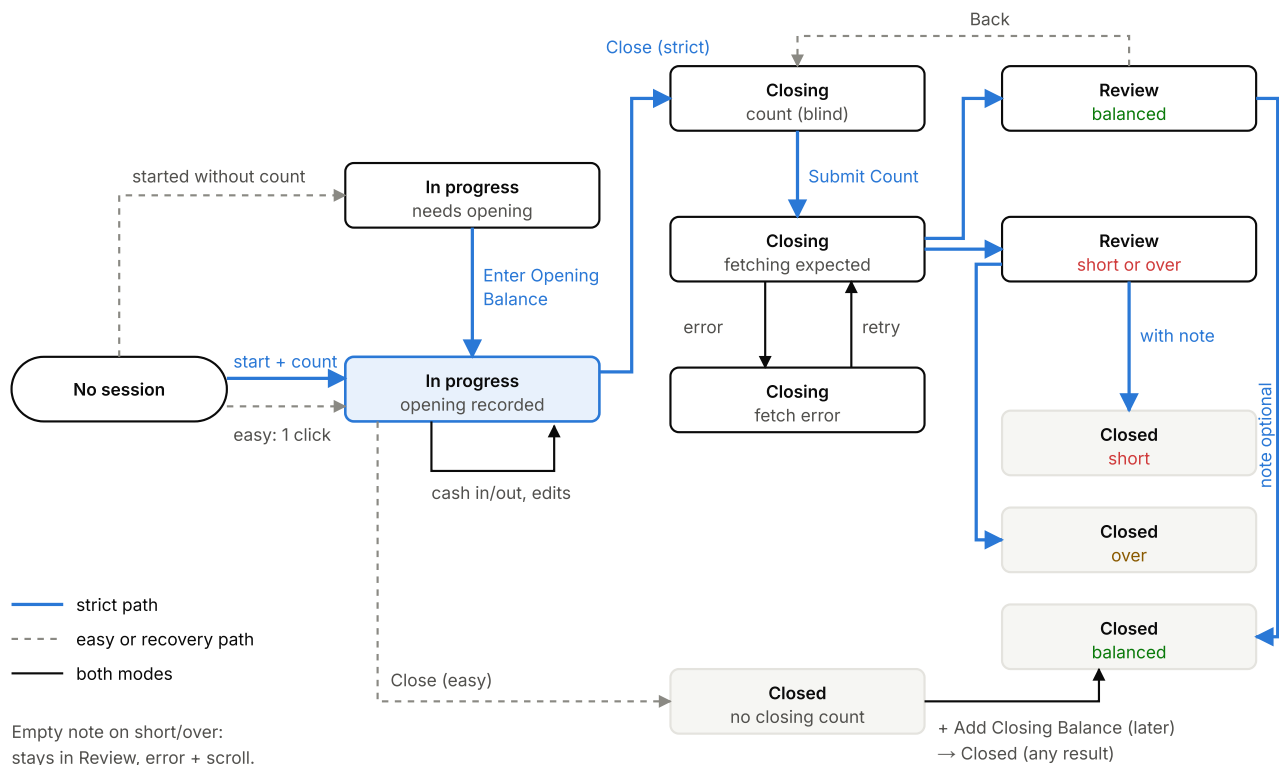


DIAGRAM Figure 2. The session and cash state machine. The blind count, the separate 'fetching expected' state with its own error and retry, and the note rule on the imbalanced branch are explicit.

The dashed path is easy mode, or recovery when a session exists without an opening count. Scroll sideways to see the whole diagram.

4.2 The drawer as a ledger

The expected amount is not a number the system should simply assert. It is a sum that a manager should be able to re-add from top to bottom:

Expected cash = opening count + cash sales - cash refunds + credit collected in cash + cash in - cash out

Difference = counted cash - expected cash

Writing the drawer as a ledger exposed three gaps that a sales report hides. *Cash that moves without a sale*, such as a change top-up, a supplier paid from the till or an owner taking money out, had no record, so cash in and cash out became entries of their own, each with a reason. *Credit settled in cash* is real cash but not a new sale, so it needed its own row; the backend engineer made sure credit collections reached the session report. And *cash versus everything else*: the drawer should only count what is physically in it, so the drawer view uses the cash share of sales, refunds and collections, and other payment methods stay in a separate transaction summary.

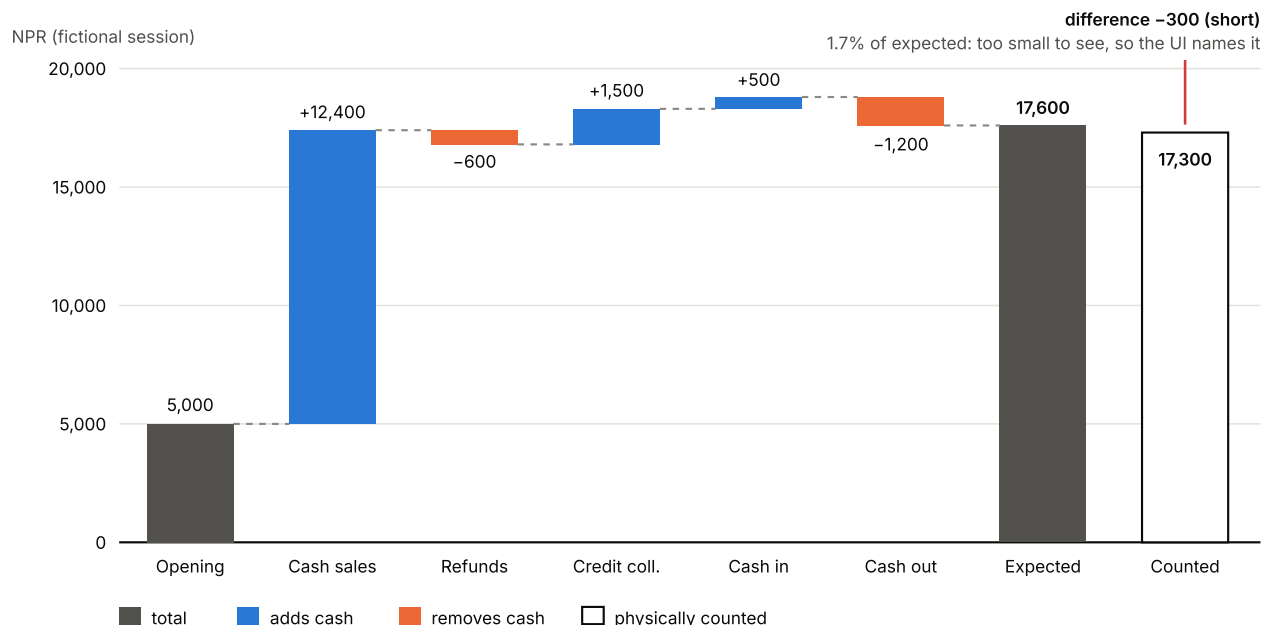


Figure 3. One fictional session as a waterfall. The movements add up to an expected Rs 17,600; the cashier counts Rs 17,300. A Rs 300 gap is under 2% of the total and almost invisible in a chart, so the interface names it in words: short by Rs 300. Scroll sideways to see the whole diagram.

The ledger became the layout. The session detail has a single Cash Drawer card that lists opening, cash sales, refunds, credit collected, cash in, cash out, expected, closing and difference, in that order (Figure 4). The expected amount is computed by the backend, not in the browser. Doing the sum in two places would eventually produce two answers, and in a cash feature two answers are worse than none. The interface's job is to show the parts of the sum so that the total can be checked by eye.

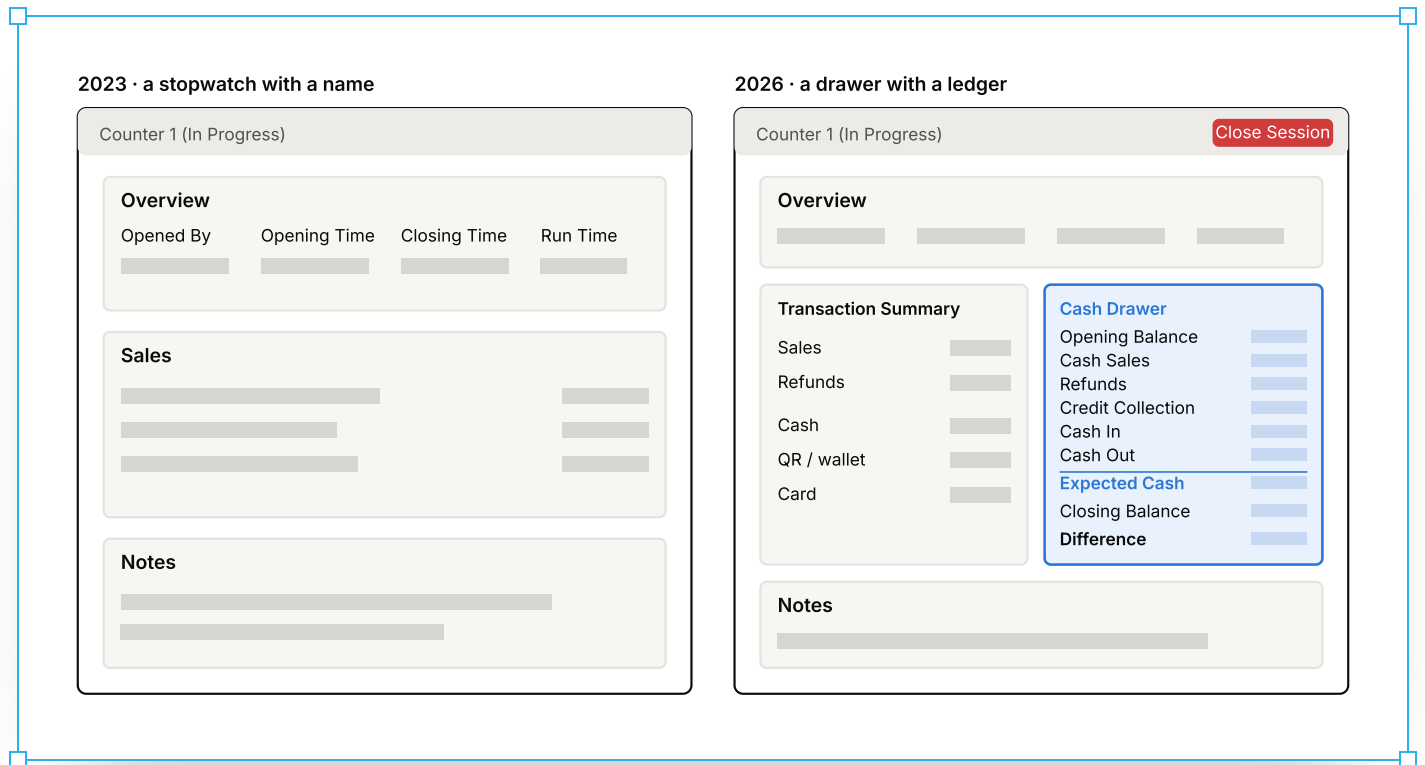
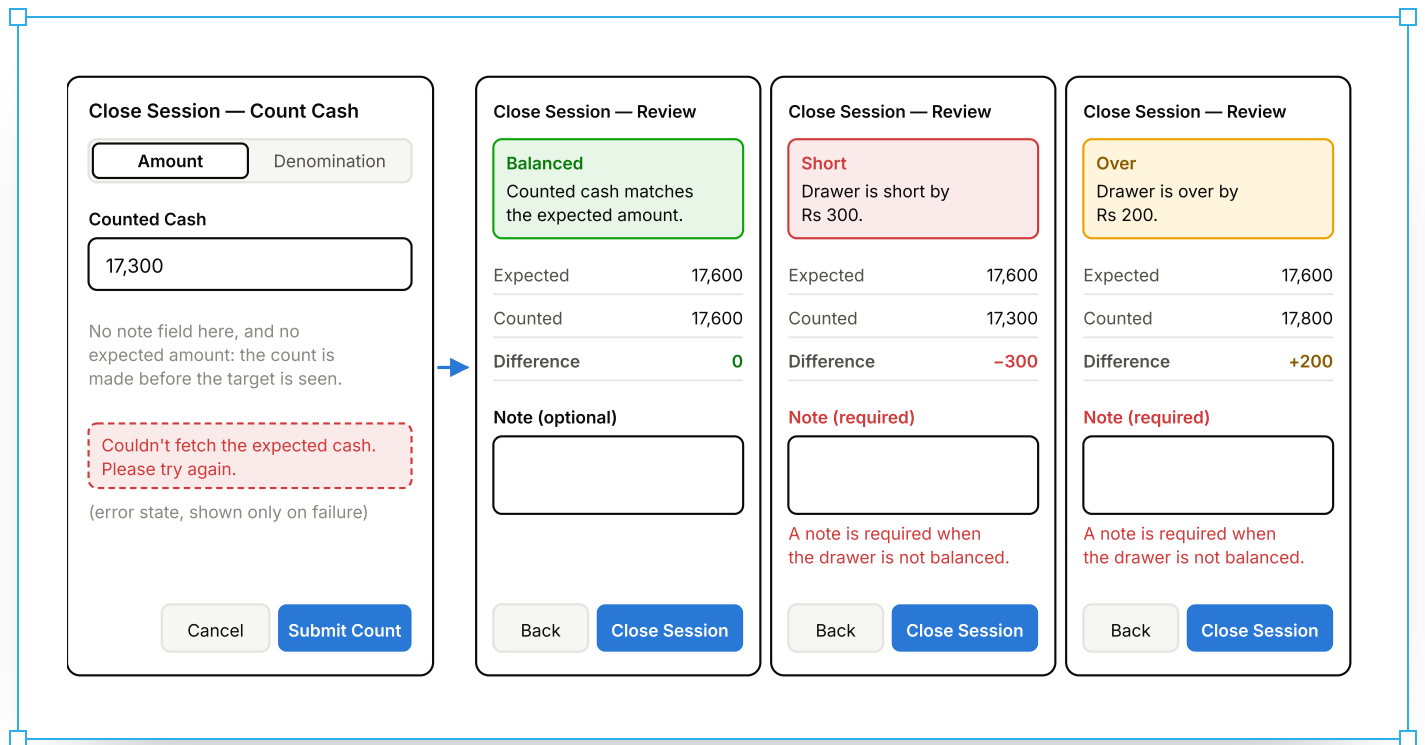


Figure 4. The session detail before and after. The earlier page showed an overview, sales and notes. The new page keeps the overview and notes, moves payments into a transaction summary, and adds a Cash Drawer card that reads like a ledger. Redrawn from the shipped screens with placeholder text. Scroll sideways to see the whole diagram.

4.3 Blind count: count first, compare second

The most important decision in the closing flow was deliberate from the start: **the cashier counts before seeing the expected amount**. The reason was to reduce bias toward matching the expected figure. If the expected amount sits next to the input, the count is no longer independent. A cashier who counts Rs 17,280 and sees Rs 17,600 has every reason to recount until the numbers agree, or to type the target (Section 2.2).

Closing therefore has two phases (Figure 5). In the *count* phase, the cashier enters a total or counts by denomination, and nothing on screen says what the answer should be. There is no note field yet either, because a note before the comparison would only invite a guess. Only after the cashier presses **Submit Count** does the system fetch the expected cash and show the *review*: expected, counted, the difference, and a plain-language status such as "Short by Rs 300". The status is carried by words as well as colour.



RECONSTRUCTION, FICTIONAL DATA *Figure 5. The two-phase close. The count phase has no expected figure and no note field. After Submit Count, the review shows one of three results; when the drawer is not balanced, the note becomes required. Redrawn from the shipped modal; fictional amounts. Scroll sideways to see the whole diagram.*

If the expected amount fails to load, the cashier stays on the count, sees "Couldn't fetch the expected cash. Please try again." and can resubmit; the count is not lost.

The blind count has one honest boundary. While a session is open, the Cash Drawer card shows the running expected cash, because staff and owners want to see the drawer build up during the day. A cashier who wants the target can find it. The two-phase close removes the target from the *moment of counting*; it does not make the whole product blind. Section 6.2 compares this with products that hide the expected amount by permission.

4.4 Empty is not zero

A blank opening count and an opening count of zero look alike on screen but mean different things. Blank means someone forgot to enter it: the value is unknown. Zero means someone checked and the drawer really was empty: the value is known. If the interface quietly turned a blank into 0, a forgotten

count would be indistinguishable from an honest empty drawer, and every difference computed after it would be wrong. This is Codd's (1979) distinction between a null and a value, applied at the input field, and a refusal of the silent transformation of input that Thimbleby and Cairns (2010) warn against.

The rules differ by kind of entry (Table 2). A *balance* is a state, and zero is a legitimate state, so a typed 0 is accepted and stays visible in the field. A *movement* is an event, and a movement of zero did not happen, so cash in and cash out need an amount above zero.

INPUT	OUTCOME	MESSAGE
Opening count left blank	Blocked; nothing saved	"Enter an opening amount."
Opening count of 0	Accepted; the 0 stays visible	None
Closing count left blank	Blocked; expected amount not requested	"Enter a counted amount."
Cash in or out of 0, or blank	Blocked	"Enter an amount greater than zero."
Cash in or out without a note	Blocked; the form scrolls to the note	"Note is required."
Denomination count with every row at 0	Valid as a balance, not as a movement	As above

Table 2. Input rules for balances and movements, with the interface's own messages.

Opening: blank

0.00

Blocked · unknown
Toast "Enter an opening amount."

Opening: 0

0

Accepted · known
The drawer really started empty

Opening: 500

500

Accepted · known
Normal float

Cash Out: 0

0

Blocked · meaningless
"Enter an amount greater than zero."

A balance is a statement of fact, so 0 is a valid fact. A movement is an event, so a movement of 0 did not happen.

The same rule holds in denomination mode: all-zero counts are a valid balance (sums to 0) but an invalid movement.

Figure 6. Four input states and their outcomes. The status is carried by the words ('Blocked', 'Accepted'), not by colour alone. Fictional data; the rules and messages are the shipped ones. Scroll sideways to see the whole diagram.

4.5 A required note when the drawer is not balanced

When the drawer is balanced, the closing note is optional. When it is short or in excess, the note becomes required: its label changes from optional to required, and pressing Close Session with the note empty shows "A note is required when the drawer is not balanced." and scrolls to the field. A mismatch means something has gone wrong, and the record should say what. The same rule applies wherever money changes without a sale: every cash in and cash out needs a note, and so does every edit to an opening or closing balance.

I did not make the note required on every close. A note that is always mandatory tends to become a single word typed forty times a week, and then it carries no information. Asking only when the numbers disagree keeps the request rare, which I expect makes a real answer more likely. That expectation is untested (Section 7).

4.6 Strict and easy modes

Some shops want strict control over cash. Others, such as an owner who is the only person at the till, would experience forced counts as friction. The design answers this with a per-location setting labelled **Require Cash Verification**, with the helper text "Require users to record opening and closing counted cash for each session." The label was chosen by the CEO. Internally we called the two states strict and easy mode (Table 3).

	STRICT (VERIFICATION ON)	EASY (VERIFICATION OFF)
Start a session	Opening-count modal	One click
Take payment without an opening count	Asked for the count at payment	Allowed
Close a session	Two-phase blind count and review	One click
Record counts	Required	Optional; can be added later from the drawer card

	STRICT (VERIFICATION ON)	EASY (VERIFICATION OFF)
Cash result in the session list	Balanced, Short, Excess	Not shown

Table 3. What the location setting changes.

The setting lives on the location, on the server, so every device at a counter behaves the same way; the editable denomination list appears beneath it.

4.7 The gate at payment

At a location that verifies cash, something has to stop a cashier from taking money before the drawer has an opening count. The question was where. My first version placed the check at every entry point to an order: the restaurant order page, the retail order page, quick orders and the payment calculator. It worked, but it could stop a cashier while they were still building an order with a customer waiting, and it had to be kept consistent in four places.

On 18 June I moved it to one place, **the payment page**, for ease of use and a better experience at the counter. The page asks for the opening count when it loads, and again if the cashier presses Complete Order without one. Order-building is never interrupted; cash is asked about at the moment cash is about to be taken (Figure 7).

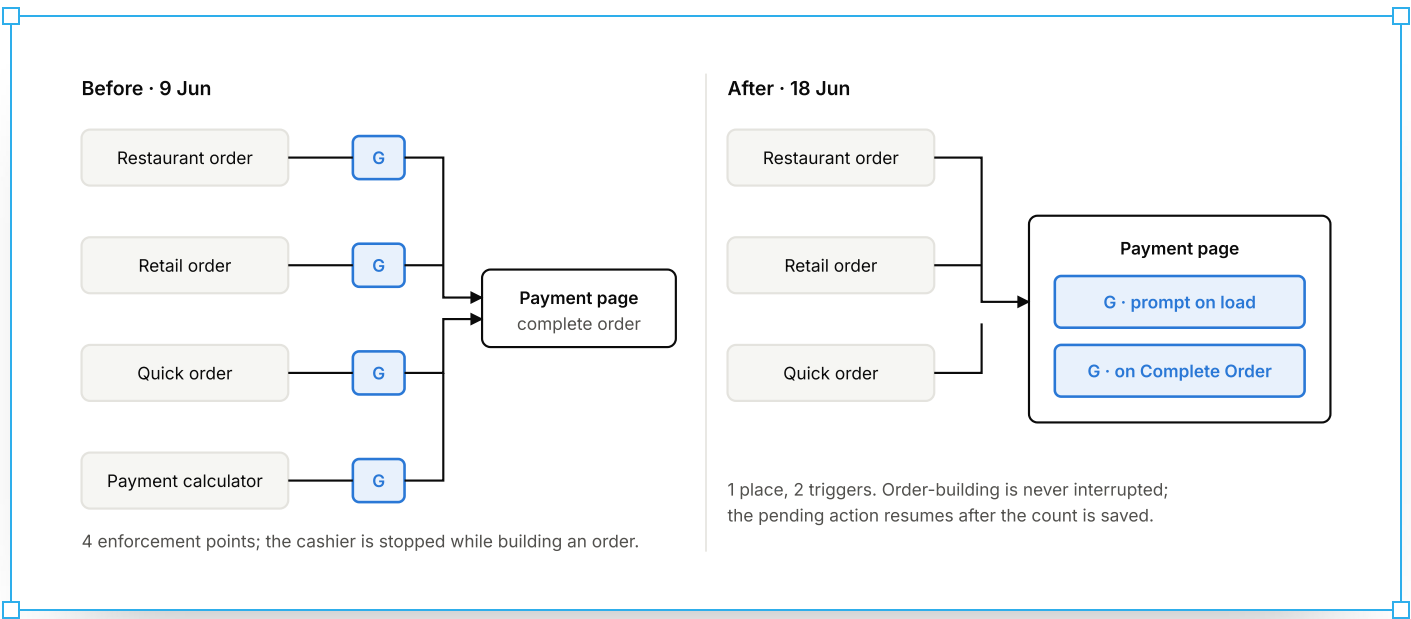


DIAGRAM *Figure 7. Gate placement before and after the 18 June change. Four enforcement points became one place with two triggers. Scroll sideways to see the whole diagram.*

Moving the gate meant the interrupted action had to survive the interruption: saving the opening count completes the order the cashier was trying to complete, and cancelling returns to the order without a spurious "unsaved changes" warning. Running against a real server also exposed a timing problem. Just after a count was saved, the page's copy of the session state had not refreshed, so it briefly believed there was no session and asked again. The implemented gate remembers the interrupted action and waits while session data is loading instead of deciding on stale information (Figure 8). Such behaviours decide whether a gate feels like a checkpoint or a trap.

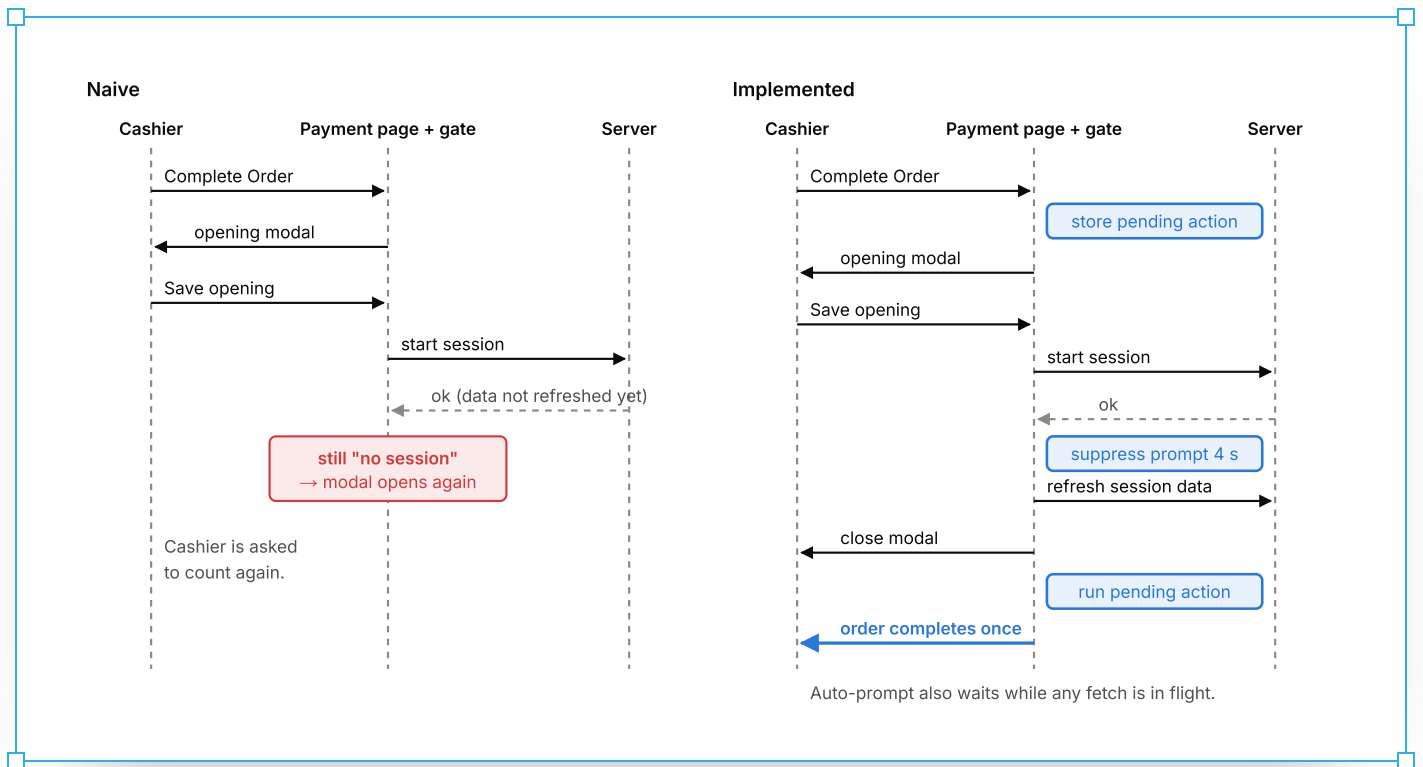


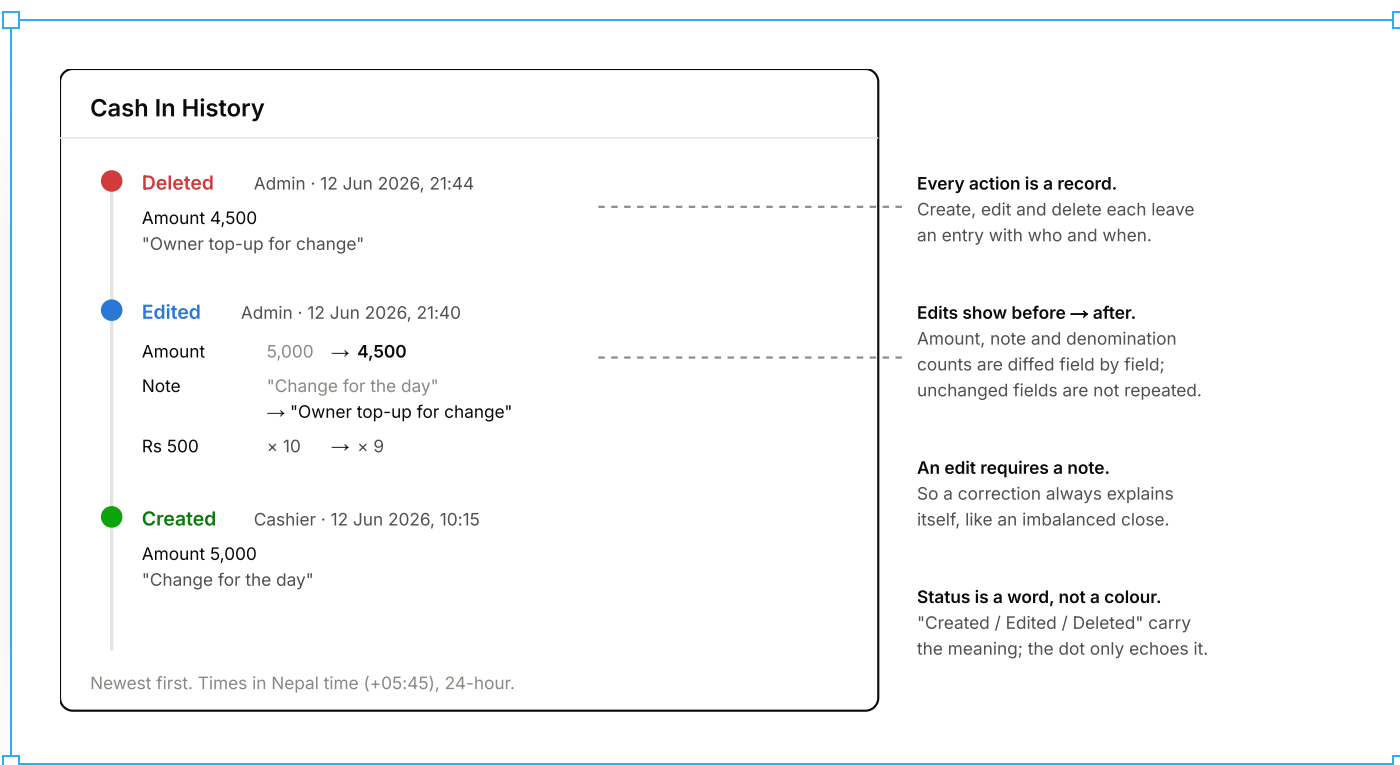
DIAGRAM Figure 8. The naive and the implemented gate. In the naive version, stale session data re-opens the modal. The implemented version remembers the action that was interrupted, holds the prompt back while session data refreshes, and then completes the order once. Scroll sideways to see the whole diagram.

4.8 Corrections and history

A cash record that cannot be corrected gets worked around; one that can be corrected silently cannot be trusted. Cashiers mistype. An owner who notices that a morning count was entered as Rs 50,000 instead of Rs 5,000 needs to fix it, and anyone reading the session later needs to see that it was fixed, by whom and why.

Every cash entry (opening balance, closing balance, each cash in and each cash out) therefore keeps its own history. Creating, editing and deleting each leave an entry with the person and the time. Edits show before and after values for each field that changed: the amount, the note and individual denomination counts. An edit requires a note, for the same reason an imbalanced close does. Deleted entries remain visible in the history (Figure 9). Because Nepal is five hours and forty-five minutes ahead of UTC, a

quarter-hour slip in reading timestamps would make corrections appear out of order, so the history handles the offset carefully and uses a 24-hour clock.



RECONSTRUCTION, FICTIONAL DATA *Figure 9. The history of one cash movement, newest first. An admin corrects an amount and its note, then deletes the entry; the cashier's original record stays visible underneath. Fictional names and amounts; times in Nepal time, 24-hour. Scroll sideways to see the whole diagram.*

4.9 Permissions

Who may change a drawer follows from the viewer dimension in Section 4.1 (Table 4).

ACTION	SESSION OWNER	ADMIN	OTHER STAFF
See the drawer and its history	Yes	Yes	Yes
Add a missing opening balance	Yes	Yes	No
Edit an opening or closing balance (note required)	Yes	Yes	No
Add, edit or delete cash in and cash out	Yes	Yes	No

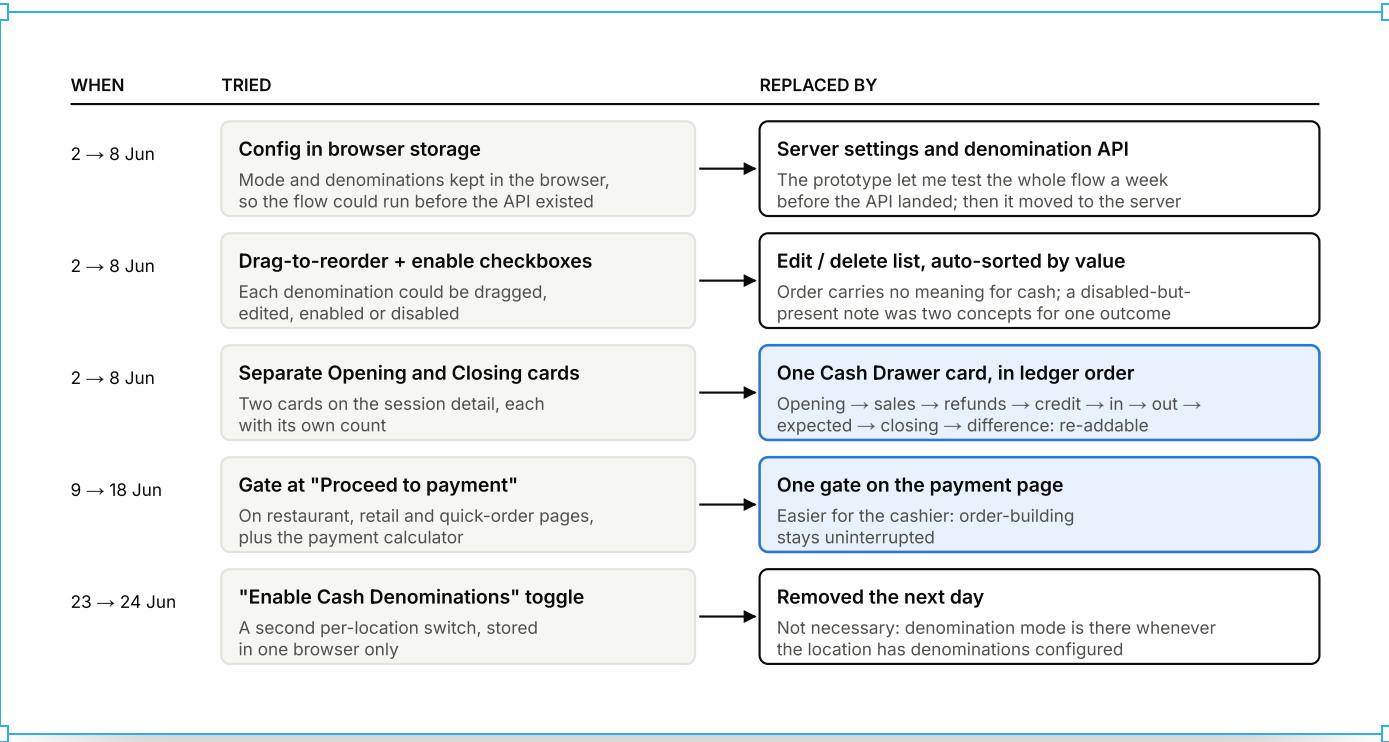
ACTION	SESSION OWNER	ADMIN	OTHER STAFF
Close the session	Yes	Yes	No
Add a closing balance after an easy-mode close	Yes	Yes	No

Table 4. Actions by viewer. The interface shows or hides actions by role; the backend enforces its own rules, which I did not design.

Other staff see the drawer and its history, but the controls that change it are hidden rather than disabled, which keeps the card readable for a cashier looking at their own drawer.

5 Iterations

Most of the decisive changes in this project removed something. Each item below was built, used and replaced. The reasons for the gate move and the toggle removal are the ones I gave at the time; the others are my retrospective reading of the changes.



DIAGRAM

Figure 10. Things that were built, tried and replaced, with dates. Scroll sideways to see the whole diagram.

DATE (2026)	BEFORE	AFTER	REASON
2–8 June	Settings and denomination list kept in the browser, so the whole flow could run before the backend existed	Settings stored per location on the server	The prototype had served its purpose; a location policy must be shared by every device at the counter
8 June	Separate Opening and Closing cards	One Cash Drawer card in ledger order	Two cards gave two numbers and no explanation; one ledger lets a manager re-add the column (retrospective)
8 June	Drag-to-reorder and per-denomination enable checkboxes	A plain list sorted by value, with edit and delete	Order carries no meaning for cash, and a present-but-disabled denomination is two concepts for one outcome (retrospective)
10–11 June	Setting labelled "Strict Session Cash", then "Session Cash Management"	"Require Cash Verification", chosen by the CEO	The final label names what staff will have to do, not the mode's internal name
12 June	Counts recordable only in strict mode	Counts optional in easy mode too	Easy-mode shops may still want a record, without a gate
9–18 June	Gate at four order entry points	Gate on the payment page only	Ease of use; order-building is never interrupted
23–24 June	An "Enable Cash Denominations" switch	Removed the next day	Not necessary: denomination counting is

DATE (2026)	BEFORE	AFTER	REASON
			available whenever the location has denominations configured

Table 5. Iterations, with dates and reasons.

Two patterns run through these changes. The first is consolidation: two cards became one ledger, four gates one, two switches one; every extra place had been somewhere for behaviour to diverge. The second is that the most consequential problems concerned timing, not appearance: what the interface shows while the server catches up, and whether an interrupted action survives. These appeared only when the flow ran against a real server, which argues for drawing in-between states in the first sketch.

Three things did not change. The blind count, the empty-versus-zero rule and the note-on-mismatch rule were present in the first working version on 2 June and survived every later revision.

6 Discussion

6.1 Transferable design principles

I offer five principles drawn from this case. They are intermediate-level design knowledge in Höök and Löwgren's (2012) sense: more general than this feature, less than a theory, and untested outside it.

P1. Model the money before the screens. Once opening, movements, closing and difference were separate records with a defined sum, every screen became a view of one ledger. Where a system holds an expected value, it should show the parts of the sum, so a person can check the total rather than trust it.

P2. Collect the independent observation before revealing the reference. Where a person measures something the system already has an expectation for, sequence the interaction so that the measurement is recorded before the expectation is shown. This is a structural response to anchoring and confirmation (Tversky & Kahneman, 1974; Nickerson, 1998) that does not depend on the person's resolve. Its scope should be stated plainly: in this design it protects the moment of counting, not the whole session.

P3. Give nothing two meanings. In anything financial, blank means unknown and zero means known and empty. Keeping them apart is a data-integrity decision, not a detail of an input field (Codd, 1979; Thimbleby & Cairns, 2010). The same principle separates states from events: a balance can be zero; a movement cannot.

P4. Ask for an explanation only when something is wrong. A note required at every step becomes noise. A note required only on a mismatch, a manual movement or a correction keeps the request rare and attaches it to the moment where an explanation is most informative. The record keeps the discrepancy instead of netting it away.

P5. Interrupt where the consequence is, and let the interrupted action finish. The gate became simpler and less disruptive when it moved to the one place where cash is actually taken. Where a system asks matters as much as what it asks, and a gate is acceptable only if completing it resumes what the person was trying to do.

One further consideration underlies all five: control should be proportionate. The per-location setting acknowledges that a reconciliation ritual suited to a shop with several cashiers is friction for a sole owner, consistent with the internal-control view that controls should fit the organisation (COSO, 2013).

6.2 Comparison with documented competitor patterns

For this paper I read the public help documentation of several established POS products in September 2026 (listed after the references). This was a documentation scan, not hands-on testing, and it was not the basis of the original design; Zoho is the only product I can say I consulted at the time. Where documentation is silent I write "not documented", which does not mean a product lacks the feature.

PATTERN	DOCUMENTED ELSEWHERE	THIS DESIGN
Opening count, cash in and out, counted against expected	Common to Zoho, Loyverse, Square, Toast, Odoo and Shopify	Same baseline; table stakes, not a contribution
Reason for a cash movement	Optional notes in Zoho and Shopify; a description in Square; preconfigured payout reasons in Toast; a reason in Odoo	Note required on every movement
Hiding the expected amount	By permission: Loyverse shows cashiers without a report right only the actual-cash field. Toast's documented close screen does the opposite: it displays the expected balance and offers the pre-calculated amount as the entry. Square's behaviour could not be verified from its public documentation	By sequence: hidden from everyone during the count, shown after it

PATTERN	DOCUMENTED ELSEWHERE	THIS DESIGN
Handling a difference at close	Zoho suggests recording a cash in or out to adjust it, and offers approval workflows; Odoo can block a close outside an authorised difference; Toast writes a default comment by variance; notes optional in Shopify	Difference kept visible and filterable as short or excess; note required
Correction history	Pay in and out history in Square; otherwise not documented	Per entry: created, edited and deleted, with before and after
Policy scope	Sessions or cash management enabled per device or register	Per-location verification setting

Table 6. Documented competitor patterns (2026 documentation scan) against this design.

Two observations follow. First, the products that hide the expected amount do so with a *permission*, which makes blindness an organisational choice and can hide the figure from cashiers entirely. This design does it with a *sequence*, which applies to everyone and needs no configuration, but reveals the figure after the count. The two approaches are complementary, and an owner-configurable blind mode for the whole session would combine them. Second, the clearest difference lies in what happens to a discrepancy. Adjusting the drawer with a balancing cash entry makes the numbers agree but removes the discrepancy from view; approval workflows move the explanation to a later step and a different person. Requiring a note at the moment of closing, and keeping the result filterable, preserves the discrepancy as a fact with an owner, a time and a reason.

Among the Nepali POS products I checked, none publicly documented a denomination-level count, an expected-versus-counted comparison or a discrepancy rule. That absence is in public documentation only.

7 Limitations and proposed evaluation

7.1 Limitations

The limits of this account are substantial, and I state them once, plainly.

- **No field study.** No cashier or owner outside the team was observed, interviewed or tested with before the feature shipped. The only walkthrough with real drawers was an internal hardware demonstration.
- **No measured outcomes.** I do not know how many locations use verification, how often drawers close short or in excess, what notes say, or whether closing became faster or slower. The only evidence of use is the CEO's report to me that the feature is live and properly used by real shops, with no usage data behind it.
- **A single designer's account.** Written by the person who designed and built the feature, months afterwards, it is subject to the biases of self-report and hindsight; retrospective reasons are marked as such.
- **An untested mechanism.** The claim that a blind count reduces bias toward the expected amount rests on general findings about anchoring and confirmation (Tversky & Kahneman, 1974; Nickerson, 1998; Furnham & Boo, 2011), not on evidence from this product.
- **Accessibility.** Every status is carried by words as well as colour, but I have not tested the flow with assistive technology.

7.2 Proposed evaluation

The following study would test the principles in Section 6.1. It is a proposal; none of it has been carried out.

Phase 1: shift-change observation. Observe at least two shift changes at each of four to six shops, mixing verification on and off and restaurants and retail counters, followed by a short interview with the cashier and the owner. The protocol would record where cashiers hesitate during the count; whether they look for the expected amount before or during counting; what they write in the note when a drawer is not balanced, and how long that takes; whether the payment-page gate ever delays a waiting customer; and how corrections are made and by whom. The aim is qualitative: to find where the design's assumptions fail.

Phase 2: discrepancy logging. With shop owners' consent, log anonymised session outcomes over a fixed period, for example eight weeks. Every measure needs an explicit denominator and method (Table 7).

QUESTION	MEASURE	DENOMINATOR	METHOD
Do people count?	Sessions closed with a count	All closed sessions at verifying locations	Session records

QUESTION	MEASURE	DENOMINATOR	METHOD
Are differences visible?	Balanced, short and excess closes, and the size of each difference	Sessions closed with a count	Session records
Are they explained?	Imbalanced closes whose note gives a reason	Imbalanced closes	Two coders classify notes as reason, placeholder or empty; agreement reported
Are corrections honest?	Edits per session, and edits made after close	Closed sessions	History records
Is the gate costly?	Time from the payment page opening to the opening count being saved	Gate prompts shown	Interface timing events
How is counting done?	Counts by total versus by denomination	Counts entered	Session records
Does anyone switch off?	Locations that turn verification off	Locations that ever turned it on	Settings history

Table 7. Proposed discrepancy log. Each measure is reported with its denominator.

The last two rows are guard rails. If the gate slows the counter, owners will switch verification off, and the feature quietly stops existing; that rate is the most important single number.

Phase 3: a controlled test of the blind count. Field data cannot isolate the effect of hiding the expected amount, because there is no comparison condition. A short task-based study could. Participants with cash-handling experience would count prepared drawers of play notes whose true contents are known, in two conditions: the expected amount hidden until the count is submitted (the shipped design), and the expected amount visible during counting. Some drawers would match the displayed expectation and some would differ from it by a known amount. The primary measure is the absolute error of the submitted count against the true contents; the secondary measure is how often a count matches the displayed expectation when the true contents do not. A within-subjects design with counterbalanced order and

different drawer contents in each condition would keep the study small. If P2 holds, the visible condition should show counts pulled toward the displayed figure.

8 Conclusion

This paper has described the design of cash-session reconciliation for a POS product used by small shops and restaurants in Nepal: from a session that recorded only a time and a person, to an accountable drawer with an opening count, recorded movements, a blind closing count, a required explanation for any mismatch and a visible history of corrections, all behind a per-location policy. It has set out the reasons for each decision and the iterations, most of which removed something, and it has drawn five principles: model the money before the screens; collect the observation before revealing the reference; give nothing two meanings; ask for an explanation only when something is wrong; and interrupt where the consequence is, letting the interrupted action finish.

These are offered as precedent and as hypotheses, not findings. The feature is in production, and the CEO reports that real shops use it, but nothing about its effect has been measured. The study in Section 7 is the next step; this account makes the reasoning available for that test and for other designers of systems where people count something and a computer holds the expected answer.

References

- Boling, E. (2010). The need for design cases: Disseminating design knowledge. *International Journal of Designs for Learning*, 1(1). <https://doi.org/10.14434/ijdl.v1i1.919>
- Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4), 397–434. <https://doi.org/10.1145/320107.320109>
- Committee of Sponsoring Organizations of the Treadway Commission. (2013). *Internal control: Integrated framework*. COSO. <https://www.coso.org/guidance-on-ic>
- Dell, N., & Kumar, N. (2016). The ins and outs of HCI for development. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 2220–2232). ACM. <https://doi.org/10.1145/2858036.2858081>
- Demirgüç-Kunt, A., Klapper, L., Singer, D., & Ansar, S. (2022). *The Global Findex Database 2021: Financial inclusion, digital payments, and resilience in the age of COVID-19*. World Bank. <https://doi.org/10.1596/978-1-4648-1897-4>

Donner, J., & Escobari, M. X. (2010). A review of evidence on mobile use by micro and small enterprises in developing countries. *Journal of International Development*, 22(5), 641–658.

<https://doi.org/10.1002/jid.1717>

Furnham, A., & Boo, H. C. (2011). A literature review of the anchoring effect. *The Journal of Socio-Economics*, 40(1), 35–42. <https://doi.org/10.1016/j.socec.2010.10.008>

Gaver, W. (2012). What should we expect from research through design? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 937–946). ACM.

<https://doi.org/10.1145/2207676.2208538>

Höök, K., & Löwgren, J. (2012). Strong concepts: Intermediate-level knowledge in interaction design research. *ACM Transactions on Computer-Human Interaction*, 19(3), 1–18.

<https://doi.org/10.1145/2362364.2362371>

Joyce, E. J., & Biddle, G. C. (1981). Anchoring and adjustment in probabilistic inference in auditing. *Journal of Accounting Research*, 19(1), 120–145. <https://doi.org/10.2307/2490965>

Nickerson, R. S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2(2), 175–220. <https://doi.org/10.1037/1089-2680.2.2.175>

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 152–158). ACM.

<https://doi.org/10.1145/191666.191729>

Perry, M., & Ferreira, J. (2017). Moneywork: Practices of use and social interaction around digital and analog money. *ACM Transactions on Computer-Human Interaction*, 24(6), 1–32.

<https://doi.org/10.1145/3162082>

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. Basic Books.

Smith, K. M. (2010). Producing the rigorous design case. *International Journal of Designs for Learning*, 7(1). <https://doi.org/10.14434/ijdl.v1i1.917>

Thimbleby, H., & Cairns, P. (2010). Reducing number entry errors: Solving a widespread, serious problem. *Journal of the Royal Society Interface*, 7(51), 1429–1439. <https://doi.org/10.1098/rsif.2010.0112>

Toyama, K. (2010). Human–computer interaction and global development. *Foundations and Trends in Human–Computer Interaction*, 4(1), 1–79. <https://doi.org/10.1561/11000000021>

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157), 1124–1131. <https://doi.org/10.1126/science.185.4157.1124>

Zimmerman, J., Forlizzi, J., & Evenson, S. (2007). Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing*

Product documentation consulted (2026 documentation scan)

Read in September 2026 for Section 6.2 only. Apart from Zoho, none of these informed the original design.

- Zoho, Retail Store sessions: <https://www.zoho.com/en-in/erp/help/sales-channel/retail-store/sessions/manage-sessions.html> and <https://www.zoho.com/en-in/erp/help/sales-channel/retail-store/sessions/session-approvals.html>
- Loyverse, shift management: <https://help.loyverse.com/help/shift-management-loyverse-pos>
- Square, cash drawer management: <https://squareup.com/help/us/en/article/5152-cash-drawer-management> and <https://squareup.com/help/us/en/article/8344-start-and-end-a-cash-drawer-session>
- Toast, cash drawer operations: <https://doc.toasttab.com/doc/platformguide/adminCashDrawerPOSOOperations.html>
- Odoo 18, Point of Sale: https://www.odoo.com/documentation/18.0/applications/sales/point_of_sale.html
- Shopify POS, register sessions: <https://help.shopify.com/en/manual/sell-in-person/shopify-pos/cash-register-management/register-sessions-in-shopify-pos>

HOW TO CITE

Saakha, B. (2026). Count first, compare second: a design case of cash-session reconciliation in point-of-sale software for small businesses in Nepal. Design case study.

<https://www.bibhushansaakha.com.np/research/cash-reconciliation-design-case>

```
@misc{saakha2026cash,  
  author      = {Saakha, Bibhushan},  
  title       = {Count first, compare second: a design case of cash-session reconciliation in point-of-sale software for small businesses in Nepal},  
  year        = {2026},  
  month       = oct,  
  howpublished = {Design case study},  
  url         = {https://www.bibhushansaakha.com.np/research/cash-reconciliation-design-case},  
  note        = {BIC Technology (Tigg), Kathmandu, Nepal}  
}
```