The Till They Couldn't Open

Designing Point-of-Sale Software for Low-Literacy, Cash-First Micro-Enterprise

Idarus Ali, Founder, AskBiz — August 2026

Abstract

Software for micro-enterprise in low-income markets is usually specified, designed, and tested by people who do not share their users' constraints. They can read a form without effort, they have a stable connection, and they are used to creating and remembering a password before a tool will let them do anything. This working paper reports on what changed, concretely, when a point-of-sale platform used by small retail, food, repair, and light-manufacturing businesses across East Africa and the Middle East was rebuilt around a different premise: that literacy, connectivity, and trust in a new login cannot be assumed, and so none of them can be a precondition for using the tool. Design responses are described in detail — replacing the keyboard with the camera wherever the task allows it, replacing the login form with a phone number and a short PIN, treating an offline, cash-only session as the default case rather than a degraded one, and teaching a new owner by walking them through a real first action rather than a rehearsal — alongside the operational evidence that shaped them. That evidence is reported plainly, including where it is uncomfortable: an internal review found that the substantial majority of trial accounts marked as active could not actually be logged into and used, because the till login checked a table that nothing had populated, including for the business owner. Separately, the single steepest drop-off in the entire sign-up funnel sat exactly one screen past the only place a user could ask a human for help. Both findings are used here as evidence for a broader claim: that the interfaces which exclude low-literacy, cash-first users are rarely missing a feature outright, and are far more often asking for a form of trust — a password, a document, a data connection — that the user has no reason to extend yet. The paper closes with what it cannot claim: none of this has been tested with low-literacy users under direct observation, at least one fix described here is designed but not yet deployed, and the figures reported are internal operational analysis, not independent measurement. It is offered as practitioner evidence for others building in the same conditions, not as a substitute for the research that should follow it.

Key Findings

1. **A login is only as inclusive as its fallback.** An internal review of every trial account that had activated the product — thirty-five, in early August 2026 — found that thirty-three, 94 percent, had no working staff login at all, including for the owner, because the till's sign-in screen checked a table that nothing had ever written a row into. The fix was not a new feature; it was making sure the account that already existed could be opened.
2. **The camera can do two jobs a form asks a literate user to do separately.** On the intake and quality-control screens rebuilt this year, the same photo a worker takes to log a batch is also run through the product's own vision recognition to pre-fill the item's name — the record and the data entry happen in one action, not two.
3. **Numeracy travels further than literacy.** Interaction that leads with a number — a price, a count, a percentage yield — degrades far more gracefully for a low-literacy user than one that leads with a text field, because reading and counting are different skills, and a large share of this population has the second without full confidence in the first.
4. **Help has to sit at the point of actual friction, not the point design assumes is hardest.** Internal funnel analysis found the steepest single drop-off in sign-up — 15 of 34 engaged users, 44 percent — occurred at the one step with no visible way to ask a person for help, one screen past where that help link had actually been placed.
5. **A setup step that saves a value nothing downstream ever reads is a promise the product doesn't keep.** The same failure — a picker or wizard collecting information that no later screen, calculation, or feature actually consumes — turned up independently in two unrelated parts of the same product, which suggests it is a structural risk of building one tool for many kinds of business, not a one-off bug.

1. Introduction

A previous paper from this desk, *Surviving on the Margins*, read the published evidence on Kenya's micro informal economy and reached a specific conclusion: the jua kali artisan, the mama mboga, the kiosk owner, and the boda boda operator are not failing to grow because they lack ambition. They are managing risk in a system that offers them almost no security — of savings, of trading space, of income against a bad week. That paper was a review of published evidence. It made no claim about software.

This paper does. It reports on what changed, in a real product used by small businesses across East Africa and the Middle East, when the design brief stopped assuming a user who reads comfortably, has a stable connection, and is happy to create and remember a new password before doing anything useful. Put simply: most software for small business is designed by people who do not share their users' constraints, and it shows in decisions nobody thinks to question — a login screen, a settings form, a "connect your data" step — that quietly assume infrastructure the target user does not have.

The paper is a practitioner's account, not a controlled study. It draws on the internal record of one product's build decisions between March and August 2026, including operational data collected in the ordinary course of running the product — sign-up funnels, activation logs, support patterns. It is offered in that spirit: not as proof that these design choices work in general, but as a specific, honestly reported record of what was built, why, and what happened next, for others building in similar conditions.

2. What "Surviving on the Margins" Leaves Open

Surviving on the Margins argued that survivalist enterprise needs security before it needs formality — a safe place to keep money, a trading spot that cannot be taken on a bribe, and something to fall back on when a shock hits. It was largely silent on a narrower, practical question: if that is what these businesses need, what does a piece of software have to look like before its target user can use it at all?

That question turns out to be underestimated in a specific way. It is tempting to read "software for the informal economy" as a features problem — add inventory tracking, add mobile money, add a simple invoice, and the job is close to done. Building the product this paper describes surfaced a different problem underneath the features one. Long before a feature can help anyone, the interface around it has to ask the user to trust something: a form, a password, a login, a step that shares data with the company. For a population whose relationship with formal institutions has, per the evidence reviewed in the previous paper, mostly been extractive — county enforcement, predatory lenders, agencies that only turn up to fine or evict — that trust is not a given. It has to be earned in the interface itself, one screen at a time, before the "real" features matter at all.

3. Three Constraints, Taken as Given

Three constraints shaped every decision described in this paper. None of them is a surprising finding on its own; the contribution here is treating all three as fixed and non-negotiable, rather than as edge cases to design around later.

Literacy is not evenly distributed, and it should not be assumed. Many of this product's users can read some, but not comfortably, and not under pressure — mid-transaction, with a queue forming, in a second or third language. Some cannot read at all. A design that requires reading a paragraph before the user can do the one thing they came to do — sell something — has already lost a meaningful share of its addressable market before a single feature is evaluated.

Connectivity is intermittent, not absent. Almost every user has a phone signal some of the time. Very few have it all of the time, and the moments it drops are not random — they cluster at the market, in the back of a shop with poor reception, during the exact hours a small business is busiest. Software that treats a lost connection as an error state, rather than a normal state, will fail its users at the worst possible moment: mid-sale, with a customer waiting.

Trust in a new login is not free. Every additional password, every new account, every unfamiliar form is a small ask of a user who has limited reason, going in, to believe this company will treat their information any better than the last one did. This is the constraint least often named in software design generally, and the one this paper spends the most time on, because it turned out to be the one most directly responsible for the largest failure described below.

4. The Camera in Place of the Keyboard

The clearest response to the literacy constraint was to make the camera the primary way information enters the system, not a secondary convenience bolted onto a text form. Across the screens rebuilt this year — recording raw material intake at a small factory, logging output at the end of a production run, flagging a quality defect, proving a parcel was handed over intact — the default view is a live camera, not a form. A worker opens the screen and the first thing they see is a viewfinder, not a set of fields.

This does more than remove typing. The same photo that becomes the permanent record of what happened is also run through the product's own image recognition to pre-fill the item's name wherever that is possible. One action — take the photo — produces both the evidence and the data entry. A form-first design asks a user to do these as two separate

acts: describe what you did, then prove it. A camera-first design treats the description and the proof as the same object, because for a worker who reads with effort, writing a product name from memory is a real barrier that a photograph is not.

The same principle governs numbers. Price and quantity are entered as digits on a large numeric keypad, not typed as free text, and, where an audio option is available, read back to the user before they confirm. This is a deliberate bet: that numeracy — the confidence to work with digits — holds up for far more of this population than reading confidence does, because counting money and haggling over a price are skills every trader already uses daily, independent of whether they read.

5. The Phone Number and the PIN in Place of a Login Form

The second response addresses the trust constraint directly. Every account on this product authenticates with a phone number and a short, self-chosen PIN — never a username invented for the occasion, never a password with rules about capital letters and symbols, and, deliberately, never an SMS one-time code that depends on a paid message actually arriving. The phone number is not a new piece of identity the user has to create and remember; it is the one they already use for everything else, including, in most of these markets, the mobile money account that already handles their cash. The PIN mirrors the exact authentication pattern that mobile money already trained an entire generation of users to trust. Nothing about signing in should feel unfamiliar to someone who has used a phone-based money agent.

This choice has a second, less obvious benefit: it removes a dependency on a paid SMS provider entirely, which means authentication keeps working even in the connectivity conditions described in the previous section, and it never silently breaks because a text message got delayed or dropped — a real failure mode of one-time-code systems that this product deliberately does not have.

6. Offline as the Default, Not the Fallback

The third response treats the connectivity constraint as the normal case rather than an exception. A cashier who has once logged in on a device can go on selling for cash with no network connection at all: the login itself checks a value already saved on the device before it tries a network call, the product catalogue is mirrored locally and kept in sync whenever

a connection is available, and every sale made offline is queued and written to the server the moment a signal returns, each one tagged so that a connection dropping mid-write can never turn into a duplicated or lost sale.

This was scoped deliberately, not built as a blanket promise. Card and mobile-money payments still need a live connection, because they depend on a third party's servers, not this product's own; what works offline is the transaction type that does not need anyone else's permission to complete — a cash sale. That is also, not coincidentally, the transaction type this population relies on most.

7. The Channel Already Trusted, Not a New One

The fourth response is about where the product shows up, not just how its own screens behave. A receipt, a repair quote, a note that an order is ready — these are sent over the messaging app the customer already has open, not through a new notification system the product would have to convince someone to trust and check. The same applies to support: when a user is stuck, the path to a real person runs through that same familiar channel, not through a ticket system or an email address they may never check.

Put simply: almost every design choice described so far reduces to the same instruction — meet the user where their trust already is, and do not ask them to build a new kind of trust just to complete the task they came to do.

8. Teaching by Doing the Real Task, Not Simulating It

An earlier version of this product's setup flow asked a new owner to walk through a simulation before they could start selling: take a practice photo, enter a practice price, add a few sample items, as a rehearsal for the real thing. It was replaced, this year, with a different approach — the new owner is taken directly to the real, live dashboard and walked through adding their first real member of staff, one small step at a time, with the interface highlighting exactly where to tap next and only advancing once that real action has actually happened, not on a timer or a "next" click.

The lesson behind the change is simple to state and easy to skip in practice: a rehearsal teaches someone what the product would feel like; doing the real task the first time teaches them what it actually does, and it produces something useful — a real staff account — instead of throwaway practice data that has to be found and deleted later.

9. The Till Nobody Could Open

Two findings from this product's own operational data motivated real changes, and are reported here in full, including where they reflect badly on earlier decisions.

In early August 2026, an internal review checked every trial account that had been marked as activated for point-of-sale — thirty-five of them — against whether anyone could actually sign in and make a sale. Thirty-three of the thirty-five could not. The till's sign-in screen checked a single staff table for a matching phone number and PIN; nothing in the sign-up flow had ever written a row to that table for the account owner. An owner could complete sign-up, see the product marked as active on their account, and have no way at all to open their own till. This was not a case of a good feature failing to be adopted. It was a login that had never been created, on an account the product's own records called active.

The fix, now live, is small and unglamorous: the moment a trial is claimed, a staff login is generated automatically for the owner, with a PIN shown once, in the same screen, at the same moment the trial activates. It closes the gap for every new account going forward. It does not, on its own, reach the roughly three dozen accounts that were already stuck before the fix shipped — those need separate, deliberate outreach, which is acknowledged here as unfinished, not solved.

10. The Cliff One Screen Past the Help Link

A separate review of the sign-up funnel — thirty-four users who had engaged with it in its first week — found the single steepest drop-off anywhere in the sequence: fifteen of the thirty-four, 44 percent, reached the step where they add their first products and never added one. No other step lost anywhere close to that share of users. The screen immediately before it had a link to ask a real person for help, on WhatsApp. The screen where people actually got stuck did not.

The lesson generalises past this one screen: a help affordance placed where the design assumes the difficulty lives, rather than where users are actually observed to stop, does very little good. At the time of writing, moving that help link to the point of real friction is designed and awaiting deployment, not yet live — reported here as an in-progress finding, not a finished fix, in the same spirit of plain accounting as the previous paper's own unfinished items.

11. The Picker With No Consumer

A third pattern surfaced twice, independently, in two unrelated parts of the same product. In one, a new business owner was asked, during sign-up, what their small factory produces, and told this would help set up the right production stages for their business. It saved the answer, and nothing downstream ever read it — every factory got an identical set of screens regardless of what it actually made, until this was found and fixed by connecting that saved answer to per-product yield targets. In the other, a repair shop was simply missing from the list of business types at sign-up, despite repair-specific tools existing elsewhere in the product; the real owner of a real repair shop, signing up in the ordinary course of business, picked "Retail" as the closest available option because there was no better one, and was then walked through a wizard built for photographing products on a shelf.

Neither is a case of a missing feature. Both are a setup step that made a promise — this will help us tailor the experience for you — that nothing downstream was built to keep. Because the same shape of bug appeared twice, in different parts of the product, built by the same team, it is treated here as a structural risk rather than a coincidence: any product that asks a user to self-identify into one of many categories, early, before showing them anything, should be checked deliberately, category by category, for whether every category's answer actually reaches a real downstream consumer. It is an easy thing to promise in a dropdown and forget to build.

12. Discussion and Limitations

This paper should not be read as more than it is. It is a single company's account of its own product decisions, evaluated against its own operational data, without a control group, without independent replication, and without direct observation of low-literacy users actually attempting these tasks. Several limitations are worth stating plainly, in the same spirit as the caveats carried in the previous paper.

No formal usability study with low-literacy participants has been conducted against any of the screens described here. The design choices are informed by the published literature on low-literacy interface design and by direct operational evidence from this product's own users, but they have not been independently validated with users under observation, and that is the single most important piece of research this paper is missing rather than reporting.

The offline design described in Section 6 has a known, accepted gap: when the same item is sold from two different offline devices before either reconnects, the system does not reconcile the two sales against each other — the server's version simply overwrites on next sync. This is a scope decision, not an oversight, but it is a real limitation for any business selling the same stock from multiple tills.

Not every fix described in this paper is deployed. Section 10's help-relocation is designed but not yet live, and is reported as such. Readers citing this paper should not assume that every design response described here is currently running in production; where that distinction matters, it has been marked.

The operational figures in Sections 9 and 10 are internal analysis on small samples — thirty-five and thirty-four accounts respectively — not independently audited statistics, and they describe one product's early users in one period, not a general claim about micro-enterprise software adoption. They are reported because they are true and because they were the direct, specific cause of the design decisions described above, not because they are statistically powerful on their own.

13. Conclusion

The previous paper from this desk argued that Kenya's micro informal enterprises need security before they need formality — a safe place for savings, a trading spot nobody can take on a bribe, income that survives a bad week. This paper has argued a narrower, adjacent point: a tool meant to serve that same population has to earn a comparable kind of security in its own interface, one screen at a time, before any feature inside it can help anyone. It does that by asking for less trust than software conventionally does — no new password, no form that has to be read before it can be filled in, no assumption that a connection will hold — and by putting the camera, the phone number, and the messaging app the user already relies on at the centre of the design, rather than treating them as accessibility add-ons around a conventional form-based product.

The clearest evidence for this argument is also the least flattering: for most of this product's early life, a meaningful share of the businesses it claimed to serve could not actually log in and use it. That is not a story about a missing feature. It is a story about designing for a user this project only partly understood, finding out plainly where that understanding fell short, and fixing the specific thing that was broken. The honest version of this paper's contribution is not a set of design principles proven to work — it is a record

of what one product got wrong, what changed as a result, and what is still unverified, offered so that the next person building in the same conditions starts a little further along than this one did.