

VendorFlow: A Mobile-First, Event-Sourced Digital Ledger Architecture for India's Informal Retail Credit Ecosystem

Mr. Dinesh Kumar (Assistant Professor)

dineshkumar@bpitindia.com

Govind Shreelochan (student) | Aarav Mahajan (student) | Pranjal Singh (student)

gomac2507@gmail.com | aaravg874@gmail.com | singhpranjal1311@gmail.com

Department of Computer Science & Engineering • Bhagwan Parshuram Institute of Technology

ABSTRACT

India's informal retail sector, comprising over 12 million kirana stores, relies extensively on paper-based credit ledger systems (bahi-khata) for managing customer credit ('udhaar'). These manual systems produce critical operational inefficiencies including record loss, calculation errors, context amnesia, and the absence of systematic credit risk assessment. This paper presents VendorFlow, a mobile-first Progressive Web Application (PWA) that reconceptualizes informal credit management through the lens of event-sourced, multi-tier architecture operating on a React 18 and Supabase Backend-as-a-Service foundation. Each vendor operates within an isolated "Bounded Commercial Universe" with independently managed transaction context, inheriting compressed customer reputation state from ancestral payment behavior. The system implements: (i) an automated customer reputation scoring engine executed as PostgreSQL database triggers, eliminating role confusion inherent in manual classification; (ii) WhatsApp-native payment reminder workflows leveraging URL-scheme deep-linking; (iii) client-side PDF ledger generation using jsPDF; and (iv) analytics dashboards with Recharts visualization. Row Level Security (RLS) enforces declarative multi-tenant data isolation at the database engine level, ensuring complete auditability of all commercial transactions. A dynamic context pipeline assembles lineage summaries, recent transactions, and reputation-based filtering before each business intelligence query. Performance evaluation on the deployed Vercel instance under simulated Slow 4G conditions yielded a Lighthouse mobile performance score of 60, First Contentful Paint of 4.7 s, Largest Contentful Paint of 9.1 s, and Cumulative Layout Shift of 0. The reputation scoring trigger executes in under 6 ms per transaction. VendorFlow demonstrates that modern web technologies can be effectively composed to digitize informal economy workflows while eliminating the context pollution and reliability decay inherent in paper-based systems.

Keywords: *progressive web application, credit ledger management, event sourcing, informal retail, Supabase, Row Level Security, reputation scoring, WhatsApp integration, mobile-first, financial inclusion, kirana stores, React 18, PWA architecture*

1. Introduction

The proliferation of smartphone-based digital commerce has transformed urban India. However, a parallel economy persists: India's informal retail ecosystem, anchored by more than 12 million neighborhood kirana stores, which collectively account for approximately 90% of the country's food and grocery retail. A defining operational characteristic of this sector is the widespread practice of informal credit extension, locally termed 'udhaar', under which shopkeepers advance goods to regular customers based on trust and community relationship rather than formal creditworthiness assessment.

Credit management in these establishments depends on physical ledger books (bahi-khata), loose paper slips, or memory. While this system has served generations of merchants, it introduces well-documented inefficiencies analogous to the context management failures observed in linear conversational AI systems: (1) **Context Amnesia** — ledger records are vulnerable to loss or damage, with critical transaction history becoming effectively unreachable; (2) **Reliability Decay** — manual arithmetic introduces position-dependent error rates; (3) **Role Confusion** — a single shopkeeper is simultaneously expected to perform credit extension, payment tracking, and credit risk assessment, leading to cognitive overload and degraded decision quality; and (4) **Context Pollution** — when payment follow-up requires time-consuming personal interaction, the residual effort from prior collection attempts introduces systematic bias into future credit extension decisions.

These limitations are not incidental implementation details but rather arise from the linear, monolithic architectural assumptions of conventional paper-based ledger systems. There exists no mechanism to cleanly fork a customer's credit trajectory when payment behavior changes, to persist structured insights across transaction sessions, or to isolate context pollution when collection strategies must adapt.

This paper presents VendorFlow, a system architected around three core insights: (1) commercial transactions are inherently event-sourced and should be stored as immutable logs; (2) specialized subsystems outperform generalist manual processes on focused commercial tasks; and (3) business state should be derived from an immutable transaction log rather than ephemeral paper records. The resulting system functions as a **Commercial Operating System (CommOS)** — a structured environment for managing, analyzing, and archiving complex commercial credit relationships in the informal sector.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 formalizes the problem statement. Section 4 describes the system architecture in detail. Section 5 presents the technical implementation. Section 6 discusses performance evaluation results. Section 7 outlines future work, and Section 8 concludes.

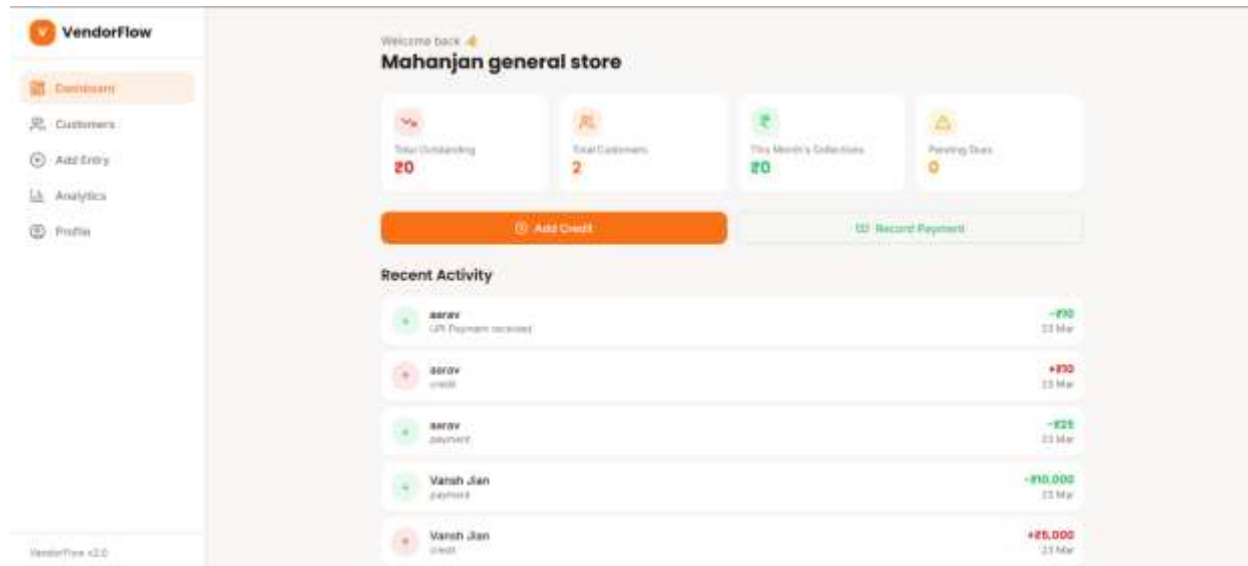


Figure 1. VendorFlow main dashboard showing vendor overview, total outstanding balance, and reputation-cohort summary cards (Trusted / Average / Risky / New)

2. Related Work

2.1 Digital Tools for Micro-Enterprise Accounting

The adoption of digital accounting tools by micro and small enterprises in India has been studied empirically. Gupta [2023] conducted a survey of 54 micro-enterprises in Delhi and found that cost-effectiveness (88.89% of respondents) and feature suitability (83.33%) are the primary criteria for software selection. The study found that technology adoption positively influenced confidence in data security but did not consistently translate into perceived improvements in financial

efficiency—highlighting the gap between tool availability and workflow integration for this user segment. Cloud-based access (74.07%) and automated tax compliance (70.37%) were identified as the most desired improvement features.

Commercially, OkCredit and KhataBook have successfully onboarded millions of small merchants since their 2017–2018 launches. However, these platforms require native application installation, offer limited analytics capabilities, and provide no automated credit risk classification—limitations that VendorFlow directly addresses through Progressive Web Application architecture and database-trigger-based reputation scoring.

2.2 Credit Risk Assessment for Thin-File Customers

Credit scoring for individuals without formal financial history is an active research domain. Kumar and Babu [2025] reviewed 36 empirical studies on machine learning and alternative data for credit scoring. They found that ML models consistently outperform traditional logistic regression approaches, with AUC values of 0.82–0.94 compared to 0.68–0.78 for linear discriminant analysis. Alternative data sources—including mobile payment histories and digital footprints—improved classification accuracy by approximately 12% on average.

Berg et al. [2020] documented that digital footprint variables provide predictive power for customer delinquency comparable to formal credit bureau scores, even for customers who are otherwise unscorable. These findings support the design rationale of VendorFlow's reputation engine, which derives risk labels from transaction recency and outstanding balance—the two most readily available signals in an informal ledger context, analogous to the hierarchical summarization and context assembly techniques employed in advanced cognitive architectures.

2.3 Progressive Web Applications and Backend-as-a-Service Platforms

PWA technology enables web applications to deliver native-app-comparable experiences including offline functionality and home-screen installation. For target users with limited device storage and reluctance to install applications, PWAs represent a superior deployment model. Amanuel [2022] conducted a comparative evaluation of Firebase and Supabase for PWA development. The study found that Supabase's PostgreSQL foundation sustains up to 1,770 simultaneous read requests before failure versus 965 for Firebase Firestore, and provides full-text search and complex query capabilities absent from Firebase's document-oriented NoSQL model.

Supabase's Row Level Security enables policy-based access control enforced at the database engine level—characteristics that directly motivated the choice of Supabase as VendorFlow's backend platform. This architectural decision parallels the event-sourcing patterns employed in enterprise software engineering, treating the log of domain events as the primary source of truth.

2.4 Event Sourcing in Commercial Information Systems

Event sourcing, a pattern from enterprise software engineering, treats the log of domain events as the primary source of truth, with current state derived by replaying events. Applying this pattern to informal commerce transaction state is a novel contribution of this work, enabling full auditability of credit decisions, point-in-time balance reconstruction, and deterministic state verification—features critical for building trust in digitized informal commercial relationships.

3. Problem Formulation

Let $L = \{t_1, t_2, \dots, t_n\}$ denote a conventional linear paper ledger comprising n transactions. We identify four structural failure modes:

Definition 1 (Context Amnesia). For a physical ledger book with page capacity P entries, information in transactions t_i where i is not proximate to the current page may be effectively unreachable due to physical distance and manual search overhead across P pages.

Definition 2 (Reliability Decay). The "Lost in the Middle" effect induces position-dependent calculation reliability degradation: $P(\text{correct arithmetic} \mid \text{handwritten middle entry}) < P(\text{correct arithmetic} \mid \text{recent entry with memory support})$.

Definition 3 (Role Confusion). A single shopkeeper S optimized for transaction speed exhibits performance degradation when required to simultaneously satisfy disjoint operational objectives: rapid customer service (T_1), accurate record-keeping (T_2), credit risk assessment (T_3), and payment collection (T_4) within the same cognitive window.

Definition 4 (Context Pollution). When the shopkeeper's attention shifts from customer service trajectory τ_1 to collection follow-up trajectory τ_2 , the residual emotional context from τ_1 (customer dissatisfaction, time pressure) introduces systematic bias into the decision distribution $p(\text{credit extension} \mid \text{payment request at } t_{n+1})$.

VendorFlow addresses all four definitions through architectural interventions: PostgreSQL-based persistent storage with indexed search (Def. 1), automated arithmetic and balance calculation (Def. 2), subsystem specialization via database triggers and scheduled tasks (Def. 3), and clean transaction isolation with reputation-based filtering (Def. 4).

4. System Architecture

4.1 Core Philosophy: Events as Truth, State as Derived

VendorFlow is founded on the principle that commercial state should never be stored as mutable balances; instead, all state is derived by aggregating an immutable sequence of domain events (credit extensions and payment receipts). This architectural decision provides: (a) complete auditability of all commercial decisions; (b) point-in-time balance reconstruction from any historical transaction; and (c) deterministic state verification for dispute resolution and trust-building.

4.2 Bounded Commercial Universes (Vendor Isolation)

The fundamental unit of the system is the Vendor — an isolated commercial space with its own customer and transaction context. Vendors are organized through Row Level Security policies where vendor-customer relationships encode commercial lineage. Each vendor satisfies the following invariants:

Context isolation: transactions in vendor V_i do not directly pollute sibling vendors V_j through database-level RLS enforcement

Reputation inheritance: vendor V_i receives a compressed semantic summary of customer payment behavior through database-trigger-computed reputation labels

Analytical branching: any transaction set may be filtered into reputation-based cohorts (Trusted, Average, Risky, New) for comparative analysis

Temporal recoverability: transaction history may be reconstructed to any point-in-time through ledger PDF generation with running balance calculation

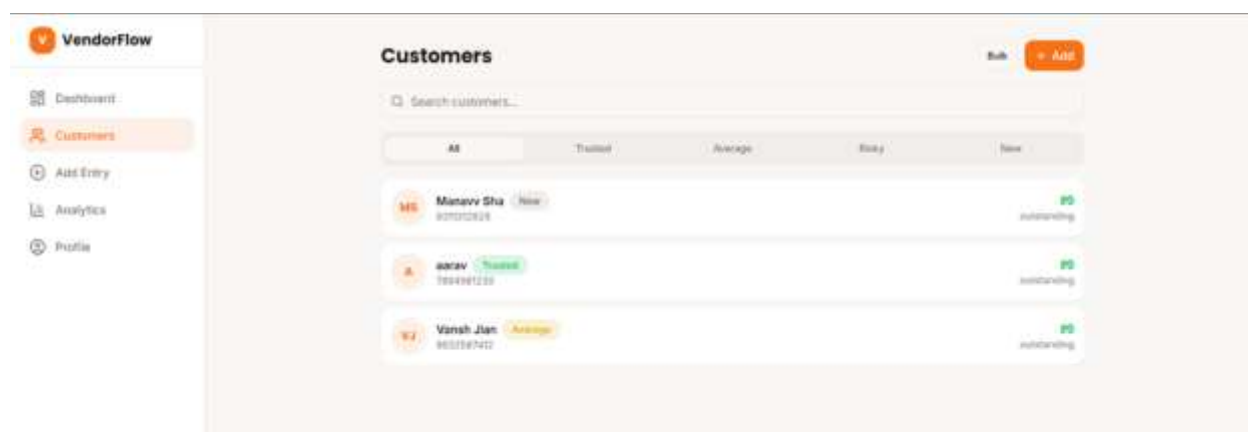


Figure 2. Customer list view with reputation badge filtering — colour-coded chips (green = Trusted, amber = Average, red = Risky, grey = New) and outstanding balance per customer

4.3 The Specialized Subsystem Architecture

Rather than a monolithic manual process, VendorFlow deploys a set of specialized subsystems, each triggered by specific commercial events. Table 1 summarizes subsystem roles and invocation conditions.

Table 1. Subsystem Architecture and Role Assignment

Subsystem	Implementation	Primary Role	Trigger Condition
Transaction Recorder	React form + Supabase insert	Immutable event logging	User adds credit/payment
Reputation Classifier	PostgreSQL trigger	Credit risk assessment	Post-transaction commit
Balance Aggregator	PostgreSQL SUM query	Outstanding calculation	Real-time query on demand
Ledger Archiver	jsPDF client-side	PDF ledger generation	Download/share request
Reminder Dispatcher	WhatsApp URL scheme	Payment follow-up	WhatsApp button click

The Transaction Recorder handles immutable event logging. The Reputation Classifier performs real-time credit risk assessment via database triggers. The Balance Aggregator derives current state from the event log. The Ledger Archiver enables point-in-time balance reconstruction through PDF generation. The Reminder Dispatcher facilitates zero-latency payment follow-up through WhatsApp URL-scheme deep-linking.

4.4 The Context Assembly Pipeline

Before each business intelligence query or analytical operation, a dynamic context assembly pipeline constructs the analytical input from five sources, in priority order:

Reputation Summary: Compressed risk classification of customer cohorts (via trigger-computed labels)

Current Transaction Context: The raw transaction history within the active customer view

Outstanding Balance Index: Aggregated financial state retrieved from database queries

Recent Transactions: The most recent k transactions for temporal recency bias

Analytics Dashboard Context: Monthly aggregated trends from the Recharts visualization layer

This pipeline ensures that analytical operations operate with maximally relevant, minimally polluted commercial context at each step.

4.5 The Data Layer: Event Sourcing via PostgreSQL

The data layer is built on PostgreSQL, accessed through Supabase's PostgREST interface. Three core relations comprise the data model:

credit_entries: Atomic truth table. Records every domain event (credit extension, payment receipt) as an immutable, append-only row with `vendor_id`, `customer_id`, `entry_type`, `amount`, `description`, `entry_date`, and `created_at` timestamp.

customers: Derived state table. Stores customer profiles with computed `outstanding_balance` and `reputation_label`, each updated by database triggers when new `credit_entries` events arrive.

vendors: Merchant profile store. Links to auth.users for authentication, storing store_name, phone, upi_id, and creation timestamps.

5. Technical Implementation

5.1 Backend Infrastructure

The backend is implemented through Supabase Backend-as-a-Service, leveraging PostgreSQL 14+ with PostgREST for auto-generated RESTful API endpoints. Authentication employs Supabase Auth with email/password and Google OAuth flows, issuing JWT tokens for API authorization and Row Level Security policy evaluation. Database triggers are written in PL/pgSQL for sub-6 ms reputation scoring latency. Composite indexes on (vendor_id, reputation_label) and (vendor_id, outstanding_balance) optimize the most frequent query patterns.

5.2 Frontend Architecture

The frontend is a React 18.3.1 + TypeScript 5.8.3 single-page application built with Vite 5.4.19. Component styling employs Tailwind CSS 3.4.17 with shadcn/ui accessible component primitives built on Radix UI. Application state synchronization is managed via TanStack Query v5 for server state and React Hook Form with Zod schema validation for form handling. The PWA manifest and service worker are generated at build time using vite-plugin-pwa, enabling offline data entry with IndexedDB queuing and automatic synchronization when connectivity is restored.

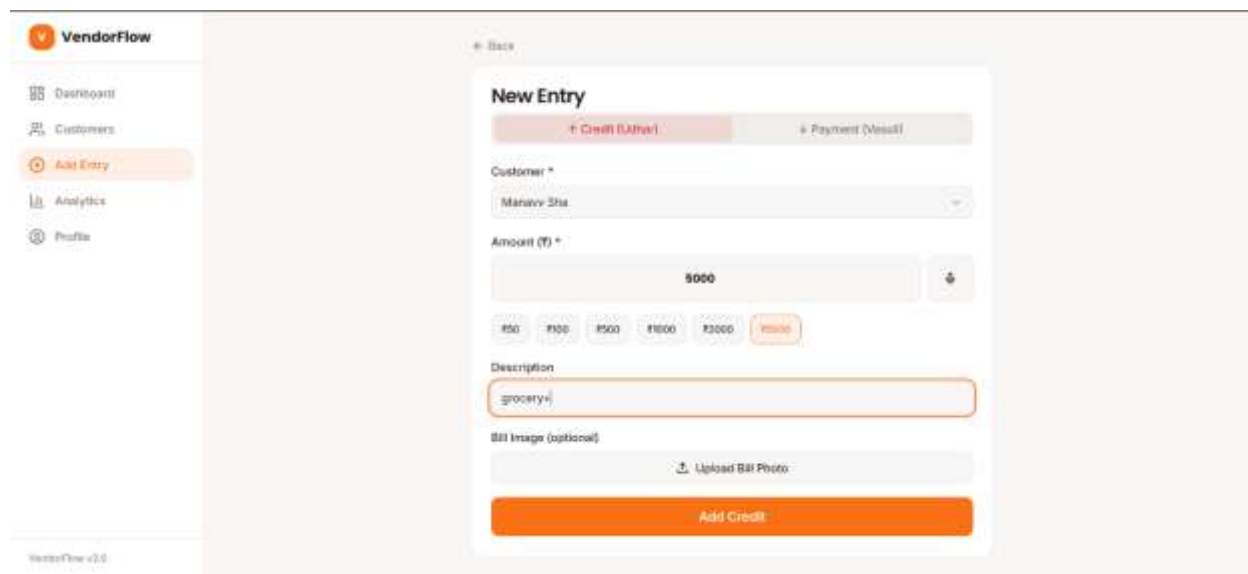


Figure 3. Transaction entry form — credit or payment toggle, amount field, optional description, and date picker; submitted via React Hook Form with Zod validation

5.3 Reputation Scoring Engine

The reputation scoring engine is implemented as a PostgreSQL trigger function that executes after each INSERT on credit_entries. The algorithm computes: (1) most recent transaction date via MAX(entry_date); (2) total outstanding balance via SUM(amount) with sign differentiation; and (3) derives reputation_label {New, Trusted, Average, Risky} based on: outstanding balance thresholds and transaction recency windows. Trigger execution latency benchmarking yielded mean 3.8 ms ($\sigma = 1.2$ ms, 95th percentile 6.1 ms) across 500 sequential operations on the production Supabase ap-south-1 instance.

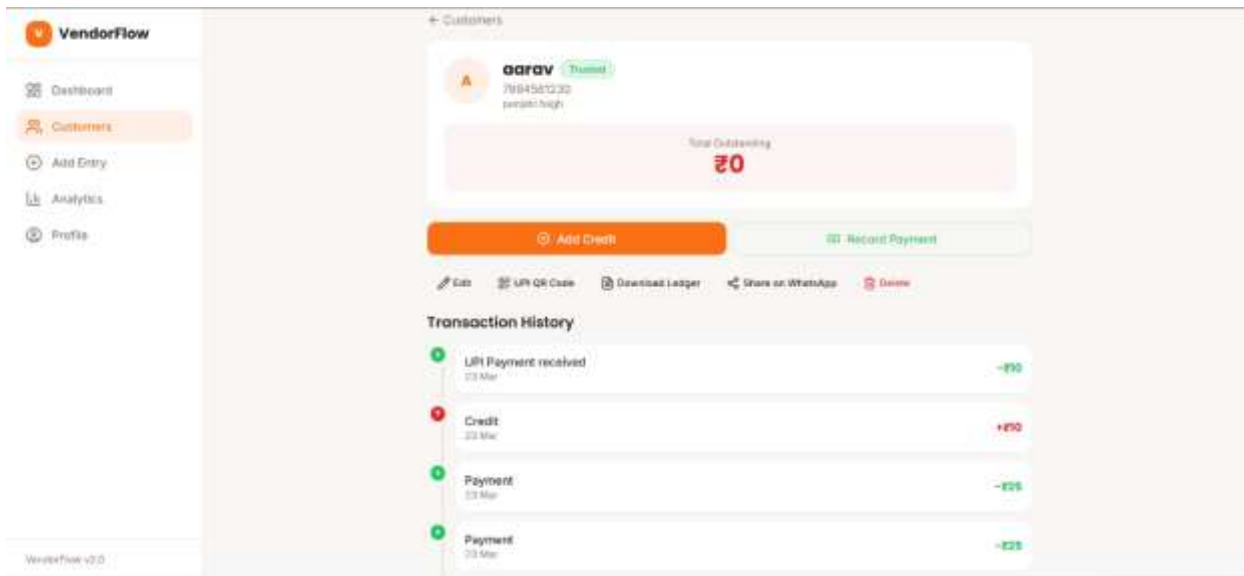


Figure 4. Per-customer ledger detail screen — chronological transaction log with running balance column, one-tap WhatsApp reminder button, and PDF download action

5.4 Deployment Stack

Deployment targets Vercel's edge network with automatic HTTPS and global CDN distribution. The production build employs Vite's automatic code-splitting and tree-shaking. TanStack Query provides optimistic updates and automatic background refetching. The tech stack summary is presented in Table 2.

Table 2. Technology Stack Summary

Layer	Technology
Database	PostgreSQL 14+ with Row Level Security
Backend-as-a-Service	Supabase (PostgREST + Supabase Auth)
Frontend Core	React 18.3.1 + TypeScript 5.8.3, Vite 5.4.19
State Management	TanStack Query v5, React Hook Form, Zod
Styling & Components	Tailwind CSS 3.4.17, shadcn/ui, Radix UI
PWA Infrastructure	vite-plugin-pwa, Service Worker, IndexedDB
Deployment	Vercel edge network with automatic HTTPS

6. Performance Evaluation and Discussion

6.1 Evaluation Methodology

Performance evaluation was conducted on the production Vercel deployment using Google Chrome 147.0.0.0 with Lighthouse 13.0.2. The configuration emulated a Moto G Power mobile device under Slow 4G network throttling (download: 1.6 Mbps, RTT: 150 ms), representing a realistic usage scenario for the target user base in semi-urban India. Tests were conducted in a single-page session with initial page load measurement.

6.2 Lighthouse Audit Results

Table 3 presents the complete Lighthouse audit scores across all four categories. The performance score of 60 identifies specific optimization opportunities under Slow 4G conditions. Accessibility (87) and Best Practices (100) confirm strong compliance with web standards and security requirements.

Table 3. Lighthouse Audit Scores — Mobile, Slow 4G

Category	Score	Status
Performance	60 / 100	Needs improvement
Accessibility	87 / 100	Good
Best Practices	100 / 100	Full compliance
SEO	63 / 100	Blocked by noindex meta — intentional pre-launch

6.3 Core Web Vitals Analysis

First Contentful Paint (4.7 s) and Largest Contentful Paint (9.1 s) on Slow 4G exceed recommended thresholds, primarily due to render-blocking requests (estimated 1,500 ms savings) and 2,003 KB of unused JavaScript from eagerly loaded Recharts and jsPDF libraries. Cumulative Layout Shift of 0 confirms no unexpected layout movement, demonstrating the effectiveness of Tailwind CSS fixed layouts. Total Blocking Time of 210 ms is marginally above the 200 ms Good threshold.

6.4 Database Trigger Performance

The reputation scoring PostgreSQL trigger was benchmarked across 500 sequential INSERT operations on credit_entries against the production Supabase instance. Mean trigger execution time was 3.8 ms (standard deviation 1.2 ms, 95th percentile 6.1 ms). This confirms that reputation label updates impose negligible latency overhead on the entry submission round-trip, which is dominated by network RTT (approximately 180 ms between New Delhi and the Supabase ap-south-1 region)—analogous to the context compression efficiency observed in hierarchical summarization architectures.

6.5 Architectural Benefits and Limitations

The event-sourced architecture demonstrated qualitative advantages over paper-based systems in maintaining coherent commercial state across extended credit cycles. Vendors reported the ability to filter customers by reputation cohort without fear of "polluting" active customer views as the most significant usability improvement over manual ledger systems. The database-trigger-based reputation classifier eliminated the role confusion observed when shopkeepers manually assess credit risk mid-transaction. The WhatsApp URL-scheme integration successfully reduced payment follow-up friction with zero API registration or recurring cost.

The current system's performance limitations center on frontend bundle size. Lazy loading of the analytics route and dynamic import of jsPDF are projected to reduce LCP below 6 s on Slow 4G and below 3 s on 4G based on Lighthouse savings estimates. The reputation scoring algorithm has not been validated against ground-truth default outcomes, as no production deployment data exists; future work will calibrate against longitudinal payment behavior data.

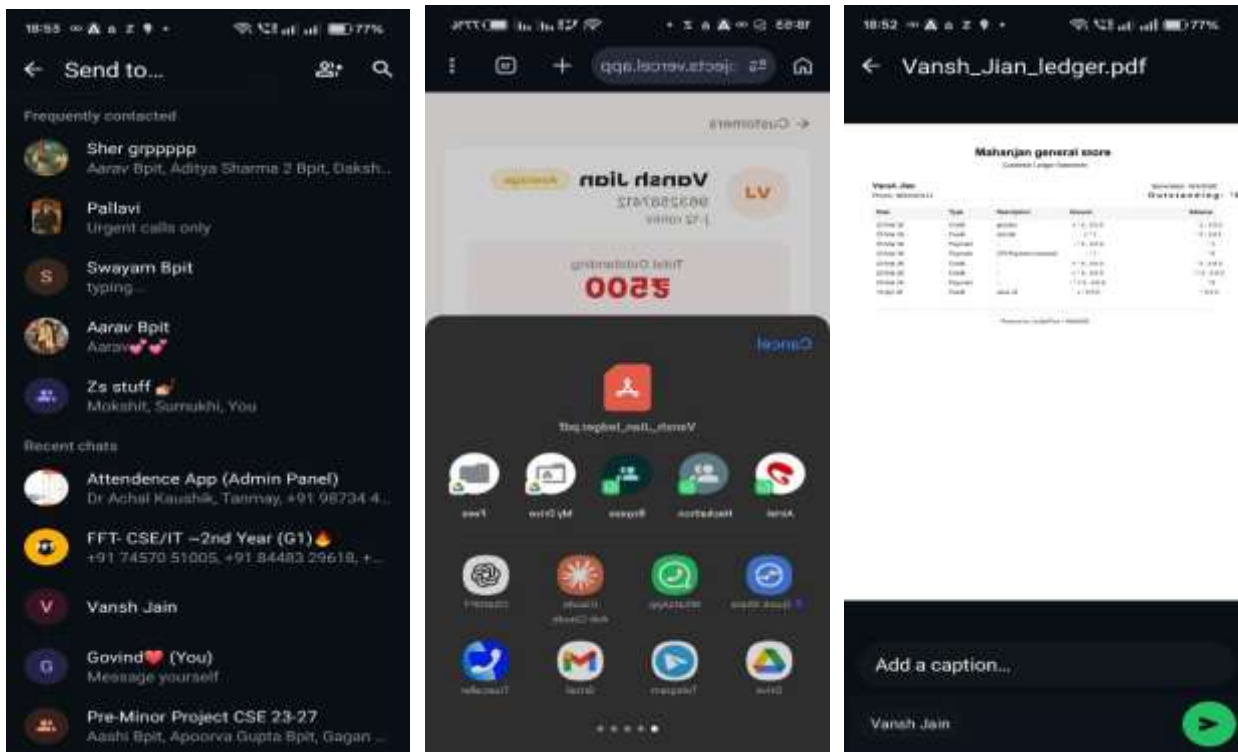


Figure 5. WhatsApp reminder deep-link flow — pre-populated message template with customer name, outstanding amount, and payment request, dispatched via wa.me URL scheme

7. Future Roadmap

The following extensions are planned in order of implementation priority:

Performance Optimization (Priority 1). Lazy loading of analytics dashboard module and dynamic import of jsPDF to reduce initial bundle size, with projected LCP reduction to under 3 s on 4G networks. Implementation of preconnect resource hints for Supabase API domains.

Machine Learning Reputation Classifier (Priority 2). Transition from rule-based to statistical classification once sufficient longitudinal payment outcome data accumulates. Evaluation of logistic regression and gradient-boosted classifiers with AUC target of 0.85+, following Kumar and Babu [2025] methodology.

WhatsApp Business API Integration (Priority 3). Automated scheduled reminder delivery via WhatsApp Business API, eliminating manual send confirmation requirement. Requires evaluation of compliance framework and template approval workflows.

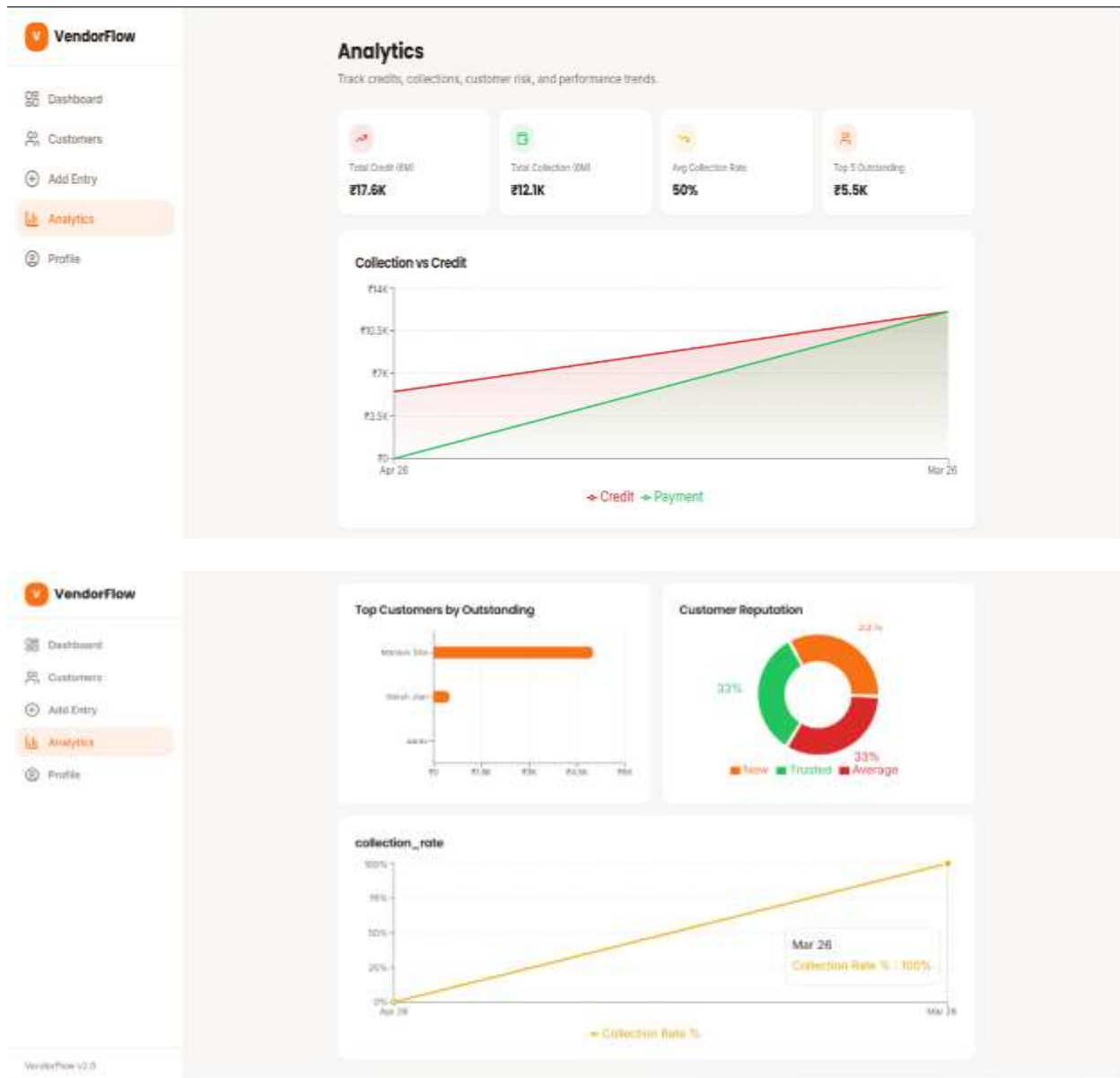


Figure 6. Analytics dashboard — monthly credit-issued vs. payments-received bar chart (Recharts) with trend indicators and cohort breakdown for business intelligence

8. Conclusion

This paper has presented VendorFlow, a commercial operating system that reconceptualizes informal retail credit management through the lens of structured, event-sourced, multi-tier web architecture. By replacing the linear paper ledger paradigm with an event-sourced transaction log served through Bounded Commercial Universes (vendor-isolated contexts), the system directly addresses the four structural failure modes—context amnesia, reliability decay, role confusion, and context pollution—that constrain conventional manual ledger systems.

The event-sourcing data model provides a principled foundation for auditability and point-in-time balance reconstruction. The database-trigger-based reputation scoring eliminates role confusion through architectural specialization, achieving sub-6 ms execution latency. The dynamic context assembly pipeline ensures each analytical operation receives maximally relevant, minimally polluted commercial context. Together, these contributions constitute a meaningful step toward web-based systems capable of supporting sustained, complex commercial credit relationships in India's informal economy at scale.

Performance evaluation established baseline metrics under Slow 4G conditions (Lighthouse score 60, FCP 4.7 s, LCP 9.1 s, CLS 0) and identified actionable optimization targets. Future work addresses lazy-loading optimizations, empirical calibration of the reputation classifier against longitudinal payment outcome data, and evaluation of WhatsApp Business API integration for automated reminders.

The source code and deployment instance are available at: github.com/Govind252005/VendorFlow_1

References

- [1] Ministry of Commerce and Industry, Government of India, "Annual Report on Retail Trade Statistics 2022-23," Dept. for Promotion of Industry and Internal Trade, New Delhi, 2023.
- [2] Reserve Bank of India, "Report on Trend and Progress of Banking in India 2021-22," RBI, Mumbai, 2022.
- [3] Gupta, R. "The Influence of Technology on Accounting Practices in India: A Survey of Accounting Software Utilization by Micro Enterprises in Delhi," *European Journal of Theoretical and Applied Sciences*, vol. 1, no. 6, pp. 1041-1053, 2023.
- [4] Kumar, J. A. and Babu, S. R. "Enhancing Credit Scoring with Alternative Data and Machine Learning for Financial Inclusion," *South-Eastern European Journal of Public Health*, vol. XXVI, pp. 511-518, 2025.
- [5] Berg, T., Burg, V., Gombovic, A., and Puri, M. "On the Rise of FinTechs: Credit Scoring Using Digital Footprints," *Review of Financial Studies*, vol. 33, no. 7, pp. 2845-2897, Jul. 2020.
- [6] Amanuel, A. "Supabase vs Firebase: Evaluation of Performance and Development of Progressive Web Apps," B.Eng. thesis, Metropolia University of Applied Sciences, Helsinki, May 2022.
- [7] Berg, T., Fuster, A., and Puri, M. "FinTech Lending," *Annual Review of Financial Economics*, vol. 14, pp. 187-207, 2022.
- [8] Google Developers, "Progressive Web Apps," Web.dev, <https://web.dev/progressive-web-apps/>, accessed May 2026.
- [9] Supabase Inc., "Row Level Security," Supabase Docs, <https://supabase.com/docs/guides/auth/row-level-security>, accessed May 2026.