Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

KHAAD Setu: A Concurrency-Safe, Role-Separated Village-Center Marketplace Platform for Smallholder Farmers

Saiprasad Zampalwad
Dept. of Computer Science & Engineering
Jawaharlal Nehru Engineering College, MGM University
Chhatrapati Sambhajinagar, Maharashtra, India

Pratik Khole
Dept. of Computer Science & Engineering
Jawaharlal Nehru Engineering College, MGM University
Chhatrapati Sambhajinagar, Maharashtra, India

Ayush Raut
Dept. of Computer Science & Engineering
Jawaharlal Nehru Engineering College, MGM University
Chhatrapati Sambhajinagar, Maharashtra, India

Sudarshan Naik
Dept. of Computer Science & Engineering
Jawaharlal Nehru Engineering College, MGM University
Chhatrapati Sambhajinagar, Maharashtra, India

Balasaheb S Sable
Dept. of Computer Science & Engineering
Jawaharlal Nehru Engineering College, MGM University
Chhatrapati Sambhajinagar, Maharashtra, India
(Project Supervisor)

***Abstract*—Smallholder farmers in India lose time and money to three recurring frictions: government schemes that are unclaimed because the tools publishing them are not usable by a digitally unfamiliar, non-English-reading farmer; a physical marketing chain that leaves one or two buyers in practical reach; and agronomic decisions made without structured data. We present an empirical engineering study of KHAAD Setu, a deployed village-center marketplace platform that addresses these frictions as one integrated system. The system comprises a 24-route, 49-service Express.js/PostgreSQL backend and a single Flutter codebase that compiles into three role-separated Android applications (farmer, village-center operator, platform administrator) sharing one design system and one API. Its core contributions are (i) a race-safe, `SAVEPOINT`-based multi-center stock-reservation design built on a single guarded `UPDATE`; (ii) a server-enforced role/flavor-separation pattern in which an app build can only narrow, never widen, an account's privileges; (iii) a self-built staged-rollout self-update mechanism with a provably monotonic rollout; (iv) a test-enforced Marathi/Hindi localisation layer; and (v) a deterministic scheme-eligibility engine over roughly fifty schemes. Verification comprises 440 backend and 604 frontend automated tests, all passing, and a direct inspection of the AWS EC2/RDS/Cloudflare-Worker deployment that found the repository documentation had drifted from reality. We report limitations with equal weight: roughly a dozen undeclared foreign keys, no CI/CD, partial localisation, an RDS connection that is encrypted but not certificate-verified, and no formal security audit or load test. This is a feasibility and engineering-verification study, not a field trial of farmer outcomes.**

***Index Terms*—agricultural marketplace, concurrency control, PostgreSQL, Flutter, role-based access control, localisation, over-the-air update, smallholder farmers**

I. Introduction

Agriculture remains the occupation of a majority of India's rural workforce, and the ease with which a farmer can reach inputs, information and markets bears directly on household income. Smallholders historically depend on physically visiting an input dealer, a government office or a town market to buy fertiliser, learn about a scheme or sell produce, and each errand costs time and travel during a cropping season.

Three frictions recur in the literature on agricultural technology in India: an *access* problem (schemes go unclaimed because the tools that publish them are unusable by many farmers) [5]; a *market-visibility* problem (few buyers in reach, no price comparison before produce perishes) [6]; and a *decision-support* problem (crop and fertiliser choices made without data) [3], [4]. We observe that these are not independent. A platform that already knows a farmer's location and purchases is well placed to also carry scheme information, local market discovery and basic advisory features.

KHAAD Setu follows from this observation. Rather than a peer-to-peer or fully centralised market, which still asks a farmer to trust a stranger, it digitises the relationship between a farmer and the nearest *village center*. The server does what a center cannot do alone: rank which of several centers can actually fulfil an order, hold stock against a reservation so two farmers cannot be sold the same sack of fertiliser, and settle payment and delivery behind the scenes.

This paper is an engineering and verification study of the deployed system, not a field trial of farmer outcomes. The primary contributions are:

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

- *Race-safe multi-center reservation:* an all-or-nothing cart hold built on one guarded SQL `UPDATE` inside per-center `SAVEPOINT`s, verified under concurrent demand.
- *Server-driven role/flavor separation:* one Flutter codebase safely serves three privilege levels because the server resolves the true role on every request.
- *Staged-rollout self-update:* a resumable, hash-verified over-the-air mechanism whose monotonic-rollout guarantee is proven by test.
- *Test-enforced Marathi/Hindi localisation:* a build fails if a phrase is added to one language file but not the others.
- *Verified deployment account:* the AWS deployment was inspected directly over SSH, exposing a documentation-versus-reality gap, and every limitation is reported.

II. Related Work

Disease detection and sensing. Nalwanga *et al.* [1] review computer-vision crop-disease detection for resource-constrained multi-crop farms, and Nguyen *et al.* [2] fuse a wireless sensor array, a ground robot and a handheld probe to map soil moisture. KHAAD Setu implements neither; these works are context for why its soil-scan feature is a deliberately simple deterministic heuristic.

Recommendation models. Gayathiri *et al.* [3] use a Random Forest and K-means on an 11-year, two-crop Tamil Nadu dataset, and Shetty *et al.* [4] offer five separate Random Forest/Decision-Tree tools, noting a possible overfitting dip in their own testing curve. Both single-region, dataset-bound models generalise poorly to a national, many-crop catalogue, so KHAAD Setu's fertiliser-review and profit features use structured data entry and heuristics instead of a trained classifier.

Scheme discovery. Khandare *et al.* [5] propose a bilingual platform using Naive Bayes, Levenshtein-distance matching and an ID3 tree for query-free scheme recommendation, with no deployed results. KHAAD Setu addresses the same access-and-language problem with a deterministic rule evaluator over a curated catalogue and adopts the language-access motivation directly.

Farmer marketplaces. Bhende *et al.* [6] match farmers to nearby merchants by K-nearest-neighbour search over Haversine distance, without field deployment. KHAAD Setu independently uses Haversine ranking, but over a small set of registered centers and fused with live stock data, which [6] does not do.

Taken together, the literature is rich in predictive models but thin on platform concerns such as concurrent stock integrity, server-enforced access control, verifiable localisation and deployed infrastructure. KHAAD Setu trains no ML model; its contribution is a concurrency-tested, deployed implementation of platform ideas that two of these papers leave at the proposed stage.

TABLE I
Comparison with Reviewed Work

Source	Core technique	In KHAAD Setu
Nalwanga [1]	CV disease detection survey	Context only
Nguyen [2]	Multi-sensor GPR fusion	Design analogy only
Gayathiri [3]	Random Forest + K-means	Scope boundary
Shetty [4]	5 RF/DT models	Predict/recommend split echoed
Khandare [5]	NB + Levenshtein + ID3	Simplified to rule engine
Bhende [6]	KNN + Haversine	Partial: ranking + stock

TABLE II
User Roles and Server-Side Resolution

Role	Resolved by	Scope
Farmer	Default for a signed-in user	Marketplace, orders, schemes, assistant
Operator	`village_center.operator_id` matches caller	`/v1/operator/*` (own center)
Admin	Verified email in server-only allowlist	`/v1/admin/*` (all centers)

III. System Design and Methodology

A. Requirements and Roles

Requirements were elicited from the three literature-identified frictions, from refinements visible in the project's git history, and from direct mapping onto two reviewed papers' problems. Three roles are enforced by the server (Table II): the *farmer*, who buys inputs, sells surplus and uses advisory features; the *operator*, who manages one center's stock and fulfils orders; and the *administrator*, who oversees all centers and accounts. Non-functional requirements include concurrency safety, role integrity, graceful degradation of optional integrations, low-bandwidth tolerance (idempotent offline sales, resumable updates), localisation correctness, auditability, and four separately sized rate limiters.

B. Architecture

The system is a three-tier design whose first tier is split into three role-separated mobile shells (Fig. 1). The Flutter app talks over HTTPS to a Cloudflare Worker, the only public edge, which forwards to an Express.js API on AWS EC2 under pm2; the API talks to PostgreSQL on AWS RDS over SSL. Five optional external services sit alongside: Supabase (authentication), Razorpay (payments), Google Gemini (assistant), Firebase Cloud Messaging (push) and OpenStreetMap/Open-Meteo (villages, weather). A route file only validates input and shapes the response; a service file holds business logic and every transaction boundary. The frontend mirrors this split (screen, Riverpod provider, abstract repository, API-backed implementation) so tests can substitute a fake repository without touching network code.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

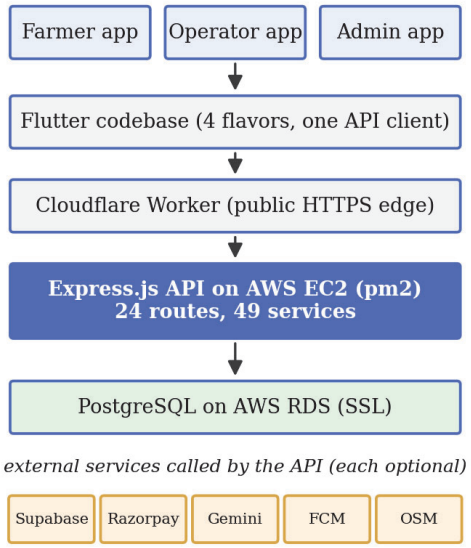


Fig. 1. System architecture of KHAAD Setu.

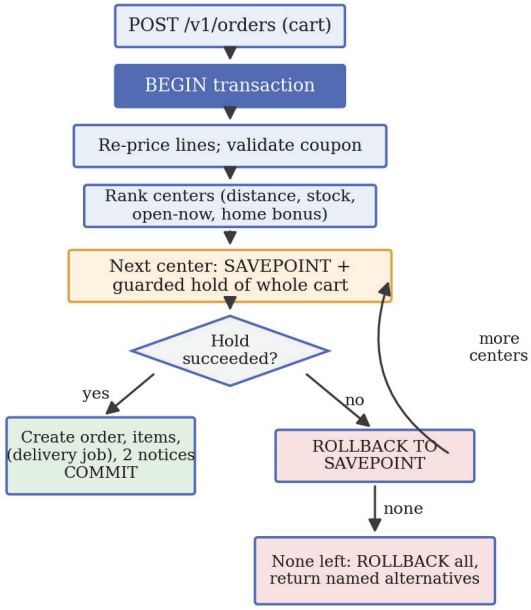


Fig. 2. Order-placement lifecycle with per-center `SAVEPOINT`.

C. Center Ranking and Race-Safe Stock Reservation

An order may be served by several centers, so assignment is a pure, no-I/O scoring function combining a Haversine distance score with adaptive radius, an inventory-match score, an operational (open-now) score and a home-center bonus. Placement then runs in one database transaction (Fig. 2): every line is re-priced from the catalogue, a coupon is validated, candidate centers are ranked, and the whole cart's stock is held at the first-ranked center under a SQL `SAVEPOINT`. A failed hold rolls back only that attempt and the next-ranked center is tried; if none can hold the full cart, the transaction rolls back and named alternatives are returned.

The underlying primitive is a single guarded update,

$$\text{reserved} \leftarrow \text{reserved} + q \quad \text{where on_hand} - \text{reserved} \geq q,$$

issued as one SQL `UPDATE` with a `WHERE` clause on that condition, where q is the requested quantity. Row-level locking serialises concurrent updates on the same row, so there is no window in which two requests both read "enough stock" and both proceed. The same primitive is reused by the resale and surplus marketplaces.

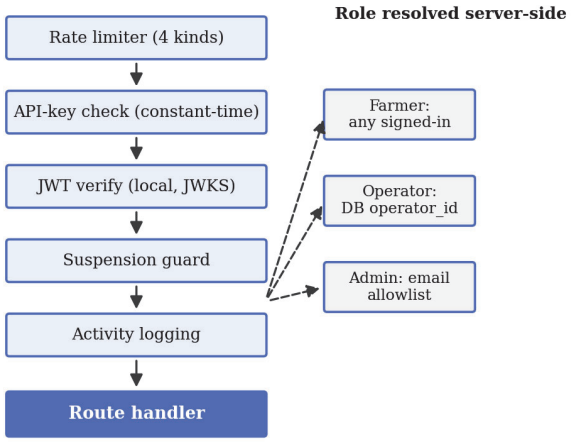


Fig. 3. Middleware chain and server-side role resolution.

D. Authentication and Role/Flavor Separation

Authentication is delegated to Supabase so that the project's security effort concentrates on authorization. The API verifies the issued JWT locally against Supabase's cached JWKS, with no per-request round trip. Every `/v1` request then passes a fixed chain (Fig. 3). The API key is compared with `crypto.timingSafeEqual`, length-checked first because that function throws on a mismatch. Operator status is a database lookup and admin status is a server-only email allowlist, never a client-supplied claim.

On the client, the installed flavor (farmer, center, admin or dev) is a compile-time constant. A single router redirect callback combines it with the server-reported role from `GET /v1/me` and confines the session to one of three route trees. A flavor can narrow who may use an APK but never widen what the server has decided: installing the admin APK and signing in as a farmer still resolves as a farmer. A suspended account receives a specific error rather than a bare 401, so a farmer can learn why they are locked out.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

E. Payments, Delivery and Wallet

Every payment is verified at two independent points, an app-reported checkout and an asynchronous Razorpay webhook, both funnelled through one function so that whichever arrives first changes state, using row locks (`FOR UPDATE`). The webhook signature is an HMAC-SHA256 over the raw request body. A database trigger rejects deletion of an issued invoice and any update to its financial fields, so corrections are always a new credit-note row. A cancelled wallet-paid order refunds synchronously, while a gateway-paid order is queued for a 30-second sweep.

Home delivery is an offer-and-accept marketplace rather than nearest-partner dispatch. Partners are ranked by existing-trip match, batchability, proximity and rating; a 30-second sweep expires stale offers under a Postgres advisory lock; long-distance jobs go to a request/approve board. Pickup and drop use separate four-digit codes compared in constant time, and five wrong attempts lock that step for fifteen minutes. The farmer wallet is an append-only ledger rather than a stored balance column.

F. Staged-Rollout Self-Update

The apps are distributed as directly installed APKs, so update delivery is the project's own responsibility. An admin publishes a release with a version code, SHA-256 and rollout percentage. The server hashes each device to a stable bucket in 0–99 and offers the update if a newer release exists and the bucket is below the rollout percentage. Because a device always lands in the same bucket, raising the percentage can only add devices. Downloads use HTTP Range requests and are SHA-256-verified before being handed to Android's installer.

G. Localisation and Advisory Features

Translatable text is keyed by its literal English string against per-language maps, with fallback to English. A dedicated test fails the build when a phrase exists in one language file but not the others, converting "do not let translations drift" into an enforced property. Coverage is scoped to farmer-facing screens and a shared staff package.

Advisory features are deliberately deterministic. The soil scan is an in-process colour heuristic over a 256×256 RGB thumbnail with no outbound call; moisture, health and NPK estimates are clipped linear combinations of channel averages, and the "disease" flag is a fixed dark-pixel threshold. Yield prediction is a two-branch heuristic: a percentile range from other farmers' results when at least three comparable reviews exist, otherwise a lookup keyed on the soil-scan flag. Scheme eligibility is a pure rule engine over about fifty schemes and a profile entered once, returning the matched rule or the specific unmet condition. The assistant first matches a message against a built-in multilingual directory of screens and answers instantly without Gemini; only non-matching messages reach the model, which removes hallucinated navigation answers for the most common question class.

TABLE III
SYSTEM SCALE AND VERIFICATION

Metric	Value
API route files / service files	24 / 49
SQL migrations / tables	28 / over 60
Backend tests (36 files)	440 passed, 0 failed
Frontend tests (40 files)	604 passed
`flutter analyze`	No issues found
Rate limiters	4 (general, OTP, scan, assistant)
Code-coverage tooling	Not configured; none reported

TABLE IV
VERIFIED BEHAVIOURS

Area	What is verified
Stock concurrency	All-or-nothing holds under contention; race loser falls back to next center
Authorization	Admin routes closed to non-admins; operator scoped to own center
Payments	Server-computed amount; forged signatures rejected; webhook idempotency
Self-update	Rollout 10%→40%→100% never un-offers an offered device
Delivery	OTP lockout; long-distance jobs never auto-dispatched
Invoicing	Database rejects edits to financial fields after issue
Offline sync	Same client reference returns the original order
Localisation	Phrase missing in one language fails the build

IV. IMPLEMENTATION AND RESULTS

A. Implementation Scale

The system was built over roughly eleven weeks in 158 commits (62 backend, 96 frontend), following a consistent backend-then-client pattern. Table III summarises the artefacts.

B. Test Evidence

A pure-logic unit layer (ranking, fees, rule evaluation) needs no database. An integration layer exercises the real HTTP surface against a disposable PostgreSQL database for the backend, and a fake repository for the frontend. Each backend test file opens its own schema in one throwaway database so files run in parallel. Table IV lists the verified areas. Two techniques account for nearly all tests, named retroactively against the actual assertions: equivalence partitioning of token classes, and boundary value analysis, for example stock equal to the requested quantity versus one short.

One test illustrates the gap between a feature working once and its guarantee being proven. The original rollout test raised the percentage once and its "consistency" check called the same endpoint twice and compared the results to themselves, an assertion that cannot fail. The replacement publishes a release at 10%, records which of 200 simulated devices are offered it, raises to 40% and then 100%, and asserts at each step that no already-offered device is dropped.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

TABLE V
SECURITY REVIEW STATUS

Area	Finding	Status
Authentication	JWT verified locally via JWKS	Implemented
API key	Constant-time, length-guarded	Implemented
Admin role	Server-side email allowlist	Implemented
Payments	Checkout and webhook both verified	Implemented
SQL injection	All observed queries parameterised	Implemented
RDS connection	Encrypted; `rejectUnauthorized` false	Partial
Foreign keys	One intentional gap, one undocumented	Partial
Penetration test	No formal audit performed	Not done

C. Security Review

D. Deployment Verification

The backend runs on an AWS EC2 Ubuntu instance under pm2, confirmed by connecting directly and reading the live process list and listening sockets. A Cloudflare Worker is the only public HTTPS edge. Deployment is a manual SSH session (`git pull`, `pm2 restart`); after one such deployment a new migration applied automatically and the live `/health` endpoint responded through the Worker URL. The repository's README describes Nginx and systemd, but neither exists on the instance. This documentation drift is reported as a finding rather than repeated.

V. DISCUSSION AND LIMITATIONS

The concurrency and role-integrity guarantees are proven by tests that exercise the failure modes they prevent. The same cannot be said of performance under load, foreign-key completeness or documentation currency, each of which was checked and found short.

Several limitations bound the conclusions. First, no dedicated load test was run, and no code-coverage figure was measured. Second, roughly a dozen columns that refer to in-database rows (for example `village_center.operator_id` and `delivery_offer.partner_id`) lack `REFERENCES` clauses, unlike their declared siblings; this is distinct from the intentional absence of foreign keys on Supabase identity columns. Third, there is no CI/CD pipeline and no verified backup schedule, and the single EC2 and RDS instances give no automatic failover. Fourth, Marathi/Hindi coverage excludes the operator and admin screens' own text. Fifth, a local Drift database is configured but unused; the offline-sale queue persists through key-value storage. Sixth, the farmer wallet has no withdrawal path. Finally, the soil-scan and yield-prediction features, despite their names, are heuristics rather than trained models, and the project is a single-developer build with a bus-factor risk. No user study of farmer outcomes was conducted.

VI. CONCLUSION

KHAAD Setu shows that a village-center marketplace for farmers can be built as a deployed, tested system rather than a prototype. Its guarded-update stock reservation, server-resolved role separation and monotonic staged rollout are each backed by tests exercising real failure modes, and its deployment was verified by direct inspection. Future work includes a CI/CD pipeline, completing foreign-key constraints after confirming cascade safety, extending Marathi/Hindi coverage to operator and admin applications, adding a wallet withdrawal path, certificate-verified database connections, a load test, and evaluating a trained crop-suitability model alongside the existing heuristics, ideally in a field study with real farmers.

ACKNOWLEDGMENT

The authors thank Mr. Balasaheb S Sable for guidance and supervision throughout the project, and the Department of Computer Science & Engineering, Jawaharlal Nehru Engineering College, MGM University, for the facilities provided.

REFERENCES

[1] B. Nalwanga *et al.*, "A review of multi-crop disease detection for resource-constrained farms," *IEEE Access*, 2025.
[2] T. Nguyen *et al.*, "A heterogeneous sensing system for high-resolution soil moisture mapping," *Comput. Electron. Agric.*, 2025.
[3] R. Gayathiri *et al.*, "Machine learning based crop suitability prediction and fertiliser recommendation," in *Proc. ICESC*, 2023.
[4] A. Shetty *et al.*, "Agrisense: A soil analysis and crop recommendation tool," *IJARCCE*, 2024.
[5] S. Khandare *et al.*, "E-Farm with S.C.H.E.M.E.: Government scheme discovery for farmers," in *Proc. RISE*, 2017.
[6] P. Bhende *et al.*, "Digital Market: An e-commerce application for farmers," in *Proc. ICCUBEA*, 2018.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)