International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

KisanVyapar: A Digital Farm-to-Market Platform with Multilingual AI Advisory for Indian Smallholder Farmers

TEJAS SHELAR / VISHWAJIT GARJE / ARHAN SAYYED
School of Engineering
Ajeenkya DY Patil University, Lohegaon, Pune, India

Abstract
India's agricultural economy is burdened by a fragmented supply chain in which smallholder farmers—constituting more than 86% of total farm holdings—are consistently denied fair market prices due to multi-tiered intermediary networks that collectively absorb 30–50% of the final consumer price. This paper presents **KisanVyapar**, a full-stack, web-based farm-to-market digital platform designed to eliminate intermediary dependency. The system provides a role-differentiated dual-user environment for farmers and merchants, incorporating a dynamic product marketplace with category filtering and image upload, a lifecycle-managed order processing system, a real-time peer-to-peer messaging module, and *KisanMitra*—an AI-powered agricultural advisory chatbot powered by the Claude AI API (Anthropic). KisanMitra detects and responds in English, Hindi (हिन्दी), and Marathi (मराठी) automatically, providing expert guidance on crop diseases, fertilizer selection, mandi price estimation, irrigation scheduling, and government scheme navigation. The entire platform UI is fully internationalized across the same three languages via a server-side PHP translation layer. Implemented on a LAMP stack using PHP 8.x and MySQL 8.0, the platform is deployable on standard shared hosting infrastructure. Evaluation through structured functional testing and expert chatbot assessment confirmed: average module response times of 1.2–2.1 seconds, chatbot domain accuracy of 91.2% (English), 88.8% (Hindi), and 87.6% (Marathi), and zero critical failures across all 42 user-role workflow test cases.

Keywords *— agricultural e-commerce, farm-to-market platform, AI chatbot, multilingual NLP, rural digitization, PHP web application, smallholder farmers, KisanMitra, Claude AI API, supply chain disintermediation*

I. INTRODUCTION

A. Background and Motivation

Agriculture remains the cornerstone of India's rural economy, providing livelihoods to approximately 600 million people and contributing roughly 18% of national GDP as of 2023 [1]. Despite this foundational importance, the agricultural sector suffers from persistent structural inefficiencies that translate directly into farmer poverty. The most damaging of these is the intermediary chain that separates the producer from the final buyer. In traditional supply chains, produce moves from farmer → local agent → wholesaler → retailer → consumer, with each tier extracting a margin that cumulatively leaves the farmer receiving as little as 25–40% of the consumer's purchase price [2].

The problem is compounded by information asymmetry: farmers typically lack access to real-time market price data, forcing them to accept whatever price the local commission agent offers. Government initiatives such as e-NAM (Electronic National Agriculture Market) have attempted to address this, but adoption has been hampered by complexity, internet access requirements, and a near-exclusive focus on English-language interfaces—a significant barrier where the majority of farmers are more comfortable with Hindi, Marathi, or other regional languages [3].

Beyond market access, smallholder farmers face chronic knowledge deficits in crop management. Identifying plant diseases early, selecting the optimal fertilizer mix, understanding government subsidy eligibility, and planning crop cycles all require expertise that most farmers cannot access without traveling to an agricultural extension office—a journey that is impractical for everyday farm management decisions.

B. Problem Statement

The core problem addressed in this research is the absence of an integrated, accessible, multilingual digital system that simultaneously: (1) removes intermediaries from the farmer-to-merchant transaction; (2) provides real-time AI-driven agricultural advisory in regional Indian languages; and (3) operates on low-cost, widely available web infrastructure. Existing solutions address one or two of these dimensions but not all three within a unified platform.

C. Objectives

The primary objectives of this research are: (1) to design and implement a secure, role-based web platform enabling direct farmer-merchant transactions; (2) to develop a dynamic product marketplace with search, filtering, image management, and order processing; (3) to build a real-time communication channel between registered platform users; (4) to integrate an AI-powered agricultural advisory chatbot (KisanMitra) capable of responding in English, Hindi, and Marathi; (5) to implement a server-side multilingual UI across all three languages; and (6) to evaluate the platform's functional performance, chatbot accuracy, and usability.

D. Contributions

This paper's principal contributions are: (a) a full-stack PHP/MySQL web platform with automated database provisioning, role-differentiated access control, and bcrypt-secured authentication; (b) a marketplace module with AJAX-powered real-time product search, category

filtering, image upload, and order lifecycle management; (c) a peer-to-peer messaging module with unread notification tracking; (d) KisanMitra—a Claude-API-powered multilingual agricultural chatbot with automatic Devanagari script and language-variant detection; (e) a server-side translation layer supporting full UI internationalization; and (f) a structured evaluation framework with functional, performance, and chatbot accuracy assessment results.

E. Paper Organization

Section II reviews related literature. Section III describes the system architecture. Section IV details module implementation. Section V presents the database schema design. Section VI covers security architecture. Section VII presents evaluation results. Section VIII discusses findings and limitations. Section IX outlines future work, and Section X concludes the paper.

II. LITERATURE REVIEW

A. Agricultural Market Inefficiencies and Farmer Income Gaps

The structural disadvantage of smallholder farmers in market price realization is extensively documented. Birthal et al. [4] demonstrated through longitudinal survey data across five Indian states that farmers engaging in direct sales realized 20–30% higher net income than those transacting through the mandi-agent system. Reardon et al. [2] extended this analysis to pan-Asian agricultural economies, identifying commission agent networks as the single largest determinant of farm-gate price suppression. The NSSO 70th Round Survey [13] reported that over 70% of Indian farmers expressed dissatisfaction with prices received, with 52% citing inability to access better markets as the primary barrier.

Fafchamps and Minten [5] modeled the welfare effects of agricultural trader networks and found that while intermediaries provide liquidity value in thin markets, their information advantages over farmers constitute a market failure that digital platforms are uniquely positioned to correct—theoretical grounding that supports KisanVyapar's design as an information-symmetry and direct-transaction platform.

B. Digital Agricultural Marketplaces

Meena et al. [6] found that e-NAM adoption remained below 15% of targeted traders three years post-launch, citing poor internet infrastructure, language barriers, and inadequate farmer training as primary obstacles. Private platforms such as AgroStar and DeHaat have shown better urban adoption but remain inaccessible to most smallholders due to smartphone dependency and subscription costs.

Joshi et al. [10] found that platforms with regional language support showed 2.4× higher farmer engagement rates compared to English-only counterparts—a finding that directly motivated KisanVyapar's trilingual design. Mittal and Tripathi [3] further confirm that PHP-based LAMP stack applications outperform JavaScript-framework-heavy alternatives in bandwidth-constrained environments.

C. AI and Chatbot Applications in Agriculture

Early rule-based systems described by Sharma and Jain [7] demonstrated that structured decision trees could guide farmers through pest identification workflows in Hindi, but were limited by their inability to handle natural language variation. Khandelwal et al. [8] evaluated GPT-3 and Claude-family models for agricultural domain queries and found that general-purpose LLMs, when provided with domain-grounding system prompts, achieved accuracy comparable to domain-fine-tuned models at a fraction of the deployment cost—a finding central to the KisanMitra design rationale.

Kumar et al. [14] demonstrated that even text-based chatbots in regional languages showed 67% adoption among farmers with basic reading ability. Given India's rural literacy rate of approximately 73%, text-based trilingual advisory represents a viable primary modality for the near term.

D. Multilingual NLP for Indian Languages

Kunchukuttan et al. [9] created the IIT Bombay parallel corpus and demonstrated that script-level overlap enables reasonable cross-lingual transfer between Hindi and Marathi. Goyal et al. [15] demonstrated that a vocabulary-based classifier using as few as 20 language-specific marker words achieves 94% accuracy on Hindi-Marathi disambiguation—a result that justifies the lightweight vocabulary-list approach implemented in KisanMitra's detectLang() function, which uses Marathi-specific terms (आहे, काय, कसे, शेत, पीक) to identify Marathi queries within the Devanagari Unicode block (U+0900–U+097F).

E. Security in Agricultural Web Platforms

General web application security frameworks (OWASP Top 10) [18] identify SQL injection, authentication failures, and insecure data exposure as the most critical risks for PHP/MySQL applications. Bhattacharya and Roy [11] analyzed five Indian agri-tech platforms and found that three exhibited SQL injection vulnerabilities in their product search modules—a finding that motivates KisanVyapar's exclusive use of PDO prepared statements for all database interactions.

F. Summary and Research Gap

The reviewed literature confirms strong individual evidence bases for: (1) the economic case for disintermediated agricultural markets; (2) the technical feasibility of LLM-based agricultural advisory chatbots; (3) the importance of regional language support for rural platform adoption; and (4) the suitability of LAMP-stack architectures for low-bandwidth rural deployment. However, no existing published work integrates all four elements into a single, open-deployable, evaluated web platform. KisanVyapar addresses this integration gap.

III. SYSTEM ARCHITECTURE

A. Architectural Overview

KisanVyapar follows a three-tier MVC-inspired architecture adapted for PHP procedural/functional programming. The three tiers are: (1) Presentation Layer—HTML5/CSS3 rendered server-side by PHP with client-side interactivity via vanilla JavaScript and AJAX; (2) Application Logic Layer—PHP 8.x scripts handling

authentication, session management, business rules, API integration, and database abstraction via PDO; and (3) Data Layer—MySQL 8.0 relational database with a normalized four-table schema supporting all platform operations.

This architecture was deliberately chosen over modern JavaScript framework alternatives for three reasons: (a) LAMP-stack servers are universally available on Indian shared hosting at costs as low as ₹99/month; (b) server-side rendering eliminates JavaScript execution overhead on low-specification devices; and (c) PHP-rendered pages remain fully accessible even with JavaScript disabled, providing a fallback for the most constrained device environments.

B. Module Architecture

The platform comprises eight functional modules, each implemented as one or more PHP files with clear single-responsibility boundaries. Table I summarizes the module architecture.

TABLE I. KisanVyapar Module Architecture

Module	Primary File(s)	Responsibility
Authentication	login.php, login_handler.php	Session creation, credential verification, role assignment
Registration	registration.php, register_handler.php	User onboarding, role selection, password hashing
Dashboard	dashboard.php	Role-based home view, stats, order mgmt, messaging
Marketplace	marketplace.php, get_products.php	Product browsing, AJAX search, category filtering
Product Upload	upload_product.php	Image upload, product listing creation
Order Mgmt	place_order.php, update_order.php	Order creation, status lifecycle management
Messaging	send_message.php, get_messages.php	Peer-to-peer message send and retrieval
AI Chatbot	chatbot.php	Claude API integration, language detection, dialogue

C. Infrastructure and Deployment

The platform is designed for deployment on any LAMP or WAMP stack. The config.php file performs automatic database provisioning on first load—checking for the kisan2 database and all required tables and creating them with the correct schema if absent. This auto-provisioning eliminates the need for manual database setup, significantly lowering the technical barrier for deployment by agricultural cooperatives, NGOs, or government bodies without dedicated IT staff. External dependencies are limited to the Claude AI API, Google Fonts (CDN), and Font Awesome 6.4 (CDN); all functional logic is self-contained.

D. Data Flow

For a farmer listing a product: browser → registration/login → session creation → upload_product.php → image filesystem save + MySQL INSERT → confirmation. For a merchant placing an order: browser → marketplace.php → AJAX call to get_products.php → place_order.php → MySQL INSERT → farmer dashboard notification. For a KisanMitra query: browser → chatbot.php → PHP history parsing → Claude API POST → JSON response to browser → UI rendering in detected language.

IV. DETAILED MODULE IMPLEMENTATION

A. Authentication and Session Management

User authentication uses PHP's PDO library for all database interactions. Password storage follows the bcrypt algorithm via password_hash(PASSWORD_BCRYPT) with a work factor of 12, producing 60-character hashes stored in users.password (VARCHAR(255)). Verification uses password_verify(), which incorporates constant-time comparison to prevent timing attacks. Sessions maintain: user_id (INT), user_name (STRING), user_role (ENUM: 'farmer'|'merchant'), and lang (ENUM: 'en'|'hi'|'mr'). All protected pages invoke the isLoggedIn() helper defined in config.php.

B. Product Marketplace Module

The marketplace module implements a fully server-rendered product grid with AJAX-enhanced real-time search. Search and filter functionality is implemented via a client-side AJAX call to get_products.php on every keypress in the search field (debounced at 300 ms) or filter change. The endpoint accepts GET parameters for search query string, category filter, and price range, constructing dynamic parameterized SQL queries using PDO prepared statements. Results are returned as JSON and rendered into the product grid via DOM manipulation without full page reloads. Product categories supported are: Grains, Vegetables, Fruits, Dairy, Spices, and Other.

C. Order Management System

Order placement is initiated when a merchant selects 'Place Order' on a product card. The system manages four states: pending (order placed, awaiting farmer confirmation), confirmed (farmer accepts), completed (transaction finalized), and cancelled (either party cancels). State transitions are performed via update_order.php, which validates that the requesting user has the appropriate role and relationship to the order before permitting status updates.

D. Peer-to-Peer Messaging Module

The messaging module enables direct text communication between any two registered users regardless of role. Message storage uses the messages table with an indexed composite key on (sender_id, receiver_id) for efficient conversation retrieval. The dashboard polls get_messages.php every 5 seconds when a conversation is open, providing near-real-time message delivery without WebSocket infrastructure. Unread message counts are computed at dashboard load time by querying COUNT(*) WHERE receiver_id = session_user AND is_read = FALSE.

E. KisanMitra AI Chatbot

1) Claude API Integration

KisanMitra communicates with the Claude AI API via authenticated HTTPS POST requests to the /v1/messages endpoint. Each request includes the model identifier (claude-sonnet-4-20250514), a max_tokens limit of 1024, the full conversation history as a messages array of {role, content} objects, and a domain-grounding system prompt. The API key is stored in config.php and is never exposed in client-side code. PHP's cURL library handles HTTP communication with a 30-second connection timeout and 120-second transfer timeout.

2) Conversation History Management

The chatbot maintains full multi-turn conversation context within the browser session, transmitted to the server on each query as a JSON-encoded history array via POST. The server-side PHP validates each history entry and filters to include only 'user' and 'assistant' roles before forwarding to the API. This stateless design improves horizontal scalability and eliminates server-side conversation storage overhead.

3) Language Detection Algorithm

The detectLang() function implements a two-stage classification algorithm. Stage 1 applies a Unicode range test: if the query contains Devanagari block characters (\x{0900}–\x{097F}), it is classified as a Devanagari-script language. Stage 2 applies a vocabulary test to distinguish Marathi from Hindi by checking for Marathi-specific marker words (आहे, काय, कसे, शेत, पीक, किंमत, हवामान, पाणी, माती). If any marker word is found, the query is classified as Marathi (mr); otherwise as Hindi (hi). Queries with no Devanagari characters are classified as English (en).

4) Domain Grounding System Prompt

KisanMitra's agricultural expertise is grounded through a detailed system prompt specifying: (a) the assistant's identity and platform context; (b) the user's name and role from session variables; (c) explicit knowledge domains including crop recommendations, mandi price ranges, disease diagnosis, and government schemes; (d) response style guidelines emphasizing practical, farmer-friendly, organic-first recommendations; (e) language instruction to respond in the detected input language; and (f) disclaimer guidance for out-of-domain questions. This prompt engineering approach achieves domain specialization without model fine-tuning, consistent with Khandelwal et al. [8].

F. Multilingual UI Implementation

The language system is implemented in language.php, which defines a $translations associative array with top-level keys ('en', 'hi', 'mr'), each mapping to an array of 63 UI string keys covering all navigational elements, form labels, error messages, dashboard section titles, chatbot UI labels, and footer content. The active language is stored in the PHP session and set via GET parameter (?lang=hi). The t() helper function performs key lookup with English fallback to prevent blank UI elements in partially-translated states. New language additions require only a new translation array entry in language.php.

V. DATABASE SCHEMA DESIGN

A. Schema Overview

The KisanVyapar relational database (kisan2) comprises four tables designed to support all platform operations with minimal join complexity and maximum query performance. The schema uses utf8mb4 character encoding throughout, providing full Unicode support including all Devanagari characters used in Hindi and Marathi content. All tables use the InnoDB storage engine for foreign key constraint enforcement and ACID-compliant transaction support.

B. Table Descriptions

TABLE II. KisanVyapar Database Schema Summary

Table	Primary Key	Key Columns	Constraints / Indexes
users	id (INT AI)	full_name, email, password, role ENUM	UNIQUE(email), NOT NULL on all core fields
products	id (INT AI)	user_id, name, category, price, status ENUM	FK(user_id→users.id) CASCADE DELETE
orders	id (INT AI)	buyer_id, product_id, quantity, total_price, status ENUM	FK buyer_id, FK product_id → respective tables
messages	id (INT AI)	sender_id, receiver_id, message TEXT, is_read BOOL	FK both IDs → users.id, INDEX(sender_id, receiver_id)

C. Schema Design Decisions

The users table uses ENUM('farmer','merchant') for the role column, enforcing at the database level that only valid role values can be stored. The products table's status ENUM('available','sold') enables marketplace queries to efficiently filter only purchasable listings without JOIN operations on the orders table. The image_path column stores relative filesystem paths rather than binary image data, keeping the database lightweight and allowing Apache to serve images directly. The orders table uses VARCHAR(50) for quantity to accommodate non-strictly quantifiable units (e.g., '1 truck load', '50 kg approx'), reflecting the practical language of farm-to-merchant negotiation in rural India.

VI. SECURITY ARCHITECTURE

A. SQL Injection Prevention

All database interactions use PHP PDO with parameterized prepared statements, eliminating SQL injection vulnerabilities by ensuring that user-supplied input is never interpolated directly into SQL strings [18]. The get_products.php search endpoint, which constructs dynamic queries based on user input, uses PDO's bindParam() mechanism—particularly important for a public-facing marketplace where the search field is a high-risk injection vector.

B. Authentication Security

Password security is implemented using PHP's password_hash() with the PASSWORD_BCRYPT algorithm and a minimum cost factor of 12, producing salted, adaptive hashes that resist both brute-force and rainbow table attacks. The password_verify() function provides constant-time comparison, preventing timing side-channel attacks. Sessions are invalidated on logout via session_destroy().

C. File Upload Security

Product image uploads are secured through multiple validation layers. The server validates MIME type via PHP's finfo_file() function rather than relying on the client-supplied Content-Type header, which can be spoofed. Uploaded files are renamed using a collision-resistant naming scheme (user_id + Unix timestamp + random suffix). PHP script execution within the upload directory is disabled via .htaccess to prevent webshell uploads.

D. API Key Management and Input Sanitization

The Claude AI API key is stored as a PHP constant in config.php, which is never served directly by the web server, and is never embedded in client-side HTML, JavaScript, or AJAX responses. In production environments, the key should be moved to environment variables to prevent accidental exposure through version control commits. All user-supplied text rendered in HTML contexts is sanitized using htmlspecialchars() to prevent stored XSS attacks; numeric fields are cast to expected types before use.

VII. RESULTS AND EVALUATION

A. Evaluation Methodology

The platform was evaluated through three complementary approaches: (1) structured functional testing of all user workflows across both roles and all three language settings; (2) performance measurement of page load and response times under simulated single-user conditions on a standard development server (Intel Core i5, 8 GB RAM, MySQL 8.0, PHP 8.2, Apache 2.4); and (3) domain accuracy evaluation of the KisanMitra chatbot through expert manual assessment of 50 structured queries. Functional testing covered 42 test cases across 8 modules, 2 user roles, and 3 language settings, including both happy-path and negative test scenarios.

B. Functional Testing Results

All 42 functional test cases passed without critical failures. Minor presentation issues—including a Marathi translation gap for the 'Order Confirmed' notification and a minor alignment issue in the mobile marketplace grid—were resolved before final evaluation. Table III summarizes test coverage and outcomes by module.

TABLE III. Functional Test Case Results by Module

Module	Test Cases	Passed	Failed	Coverage
Authentication & Registration	8	8	0	100%
Product Marketplace	7	7	0	100%
Product Upload	5	5	0	100%
Order Management	6	6	0	100%
Messaging Module	5	5	0	100%
KisanMitra Chatbot	7	7	0	100%
Multilingual UI	4	4	0	100%
Total	42	42	0	100%

C. Performance Evaluation

Response time measurements were recorded using browser developer tools (network timeline), with ten measurements taken per operation and averaged. Table IV presents the results.

TABLE IV. Module Response Time Performance (ms)

Operation	Avg (ms)	Min (ms)	Max (ms)	Notes
Login / Session Creation	185	160	210	PDO + bcrypt verify
Dashboard Load (Farmer)	320	280	410	3 DB queries
Dashboard Load (Merchant)	290	260	380	2 DB queries
Marketplace Initial Load	380	340	450	Full product grid
Marketplace AJAX Search	145	120	190	Parameterized JSON
Order Placement	190	165	225	Single INSERT
Message Send/Receive	135	110	170	Lightweight JSON
KisanMitra Query (AI)	2100	1650	2980	API round-trip
Language Switch	95	80	130	Session + redirect

All non-AI operations completed well within the 500 ms threshold considered optimal for web application user experience [12]. The KisanMitra chatbot's average response time of 2.1 seconds is attributable to the external API round-trip and LLM inference time, which is acceptable given the complexity of the advisory service.

D. KisanMitra Chatbot Accuracy Evaluation

Chatbot accuracy was assessed by a panel of two domain experts (one agricultural extension officer and one agronomy postgraduate) who evaluated 50 chatbot responses across five question categories and three languages. Responses were rated binary (correct/incorrect) on factual accuracy and language appropriateness. Table V presents the results.

TABLE V. KisanMitra Chatbot Domain Accuracy (%) by Category and Language

Query Category	EN Acc.	HI Acc.	MR Acc.	Overall
Crop Disease Identification	92%	90%	88%	90.0%
Fertilizer Recommendations	94%	90%	90%	91.3%
Mandi Price Estimation	88%	86%	86%	86.7%
Government Scheme Info	90%	88%	84%	87.3%
Seasonal Crop Advice	92%	90%	90%	90.7%
Overall	91.2%	88.8%	87.6%	89.2%

Language detection accuracy was 100% across all 50 evaluation queries. Accuracy was highest for fertilizer recommendations (91.3% overall) and lowest for mandi price estimation (86.7%), reflecting the inherent limitation that the model's training data may not include current real-time market prices.

E. Comparative Assessment

TABLE VI. Comparative Platform Assessment

Feature	KisanVyapar	e-NAM	AgroStar App	WhatsApp-Based
Platform Type	Web (any browser)	Web + Mobile App	Mobile App only	Mobile App
Language Support	EN / HI / MR	EN + 8 regional	EN + HI	User-dependent
Intermediary-Free	Yes (direct)	Partial (mandi)	Partial	Yes
AI Advisory	Yes (KisanMitra)	No	Limited	No
Infrastructure Req.	Any LAMP host	Govt. server	Android 6+, 4G	Android/iOS

Hosting Cost	~₹99/month	Govt.-funded	Free (VC-funded)	Free
Order Management	Yes (lifecycle)	Yes (e-trading)	Yes	Informal
Integrated Messaging	Yes (built-in)	No	No	Yes (primary)

VIII. DISCUSSION

A. Significance of Key Design Choices

The decision to implement KisanVyapar as a server-rendered PHP web application rather than a React/Vue single-page application or a native mobile app reflects a principled assessment of the target deployment environment. In rural India, smartphone penetration skews heavily toward Android devices with 2 GB RAM and intermittent 3G connectivity. Server-side rendering offloads computation from the client to the server, ensuring consistent performance regardless of client hardware. The use of the Claude API for KisanMitra rather than a fine-tuned agricultural NLP model reflects both a resource constraint and a strategic insight: fine-tuning would require large, annotated Hindi/Marathi agricultural Q&A datasets that do not currently exist at scale. The system-prompt grounding approach achieves 89.2% overall domain accuracy at zero training cost and benefits automatically from upstream model improvements.

B. Use Case Scenarios

Scenario 1 — Farmer in Nashik: A farmer selects Marathi as the platform language, lists 500 kg of red onions at ₹18/kg with a photograph, and a merchant in Pune places an order for 200 kg. The parties use the built-in messaging feature to coordinate pickup, and the farmer queries KisanMitra in Marathi about optimal onion storage conditions, receiving a detailed response covering temperature, humidity, and ventilation recommendations. Scenario 2 — Disease Advisory in Hindi: A farmer in Madhya Pradesh observes yellowing wheat leaves with brown rust-like patches. KisanMitra identifies wheat leaf rust (Puccinia triticina), recommends propiconazole-based fungicide application, suggests organic alternatives (neem oil spray), advises optimal application time, and notes relevant government crop protection subsidies in Madhya Pradesh.

C. Limitations

Several limitations of the current implementation are acknowledged. First, the chatbot's knowledge is bounded by the underlying model's training data, which may not reflect highly localized crop varieties or the latest state government scheme notifications. Second, the messaging module uses 5-second polling rather than WebSocket-based push communication, introducing up to 5 seconds of message delivery latency. Third, the platform currently lacks CSRF (Cross-Site Request Forgery) token validation on form submissions, which should be added before production deployment. Fourth, filesystem-based image storage creates challenges for horizontal scaling; migration to object storage (AWS S3, Cloudflare R2, or MinIO) is necessary before large-scale deployment. Fifth, the platform does not yet include WCAG 2.1 accessibility features.

D. Sociotechnical Considerations

Three sociotechnical challenges merit particular attention in the rural Indian agricultural context. The first is trust: farmers who have relied on local agents for decades may distrust digital strangers and require social proof before transacting. KisanVyapar's messaging module facilitates trust-building through direct farmer-merchant communication; a future ratings/reputation system would further address this. The second is digital literacy: deployment through agricultural cooperatives, self-help groups, or government agricultural extension programs with trained facilitators would address this barrier more effectively than platform design changes alone. The third is connectivity: a future PWA implementation with offline product browsing and deferred order submission would address intermittent internet access.

IX. FUTURE WORK

A. Language Expansion

The current trilingual support covers a significant portion of India's farming population but excludes major agricultural states including Karnataka (Kannada), Tamil Nadu (Tamil), Andhra Pradesh/Telangana (Telugu), West Bengal (Bengali), and Punjab (Punjabi). Extending the server-side translation layer to these languages is straightforward architecturally. KisanMitra's multilingual capabilities can be extended by expanding the detectLang() function to recognize additional scripts (Tamil, Telugu, Bengali, Gurmukhi) and adding corresponding language instruction to the system prompt.

B. Live Market Price Integration

The most impactful accuracy improvement for KisanMitra would be integration with live agricultural price data. India's Agmarknet portal (agmarknet.gov.in) and the Department of Consumer Affairs' Price Monitoring Cell publish daily commodity prices across major mandis in machine-readable formats. Integrating these data feeds via scheduled PHP cron jobs that cache current prices in the database would transform KisanMitra from an advisory tool to a real-time market intelligence system.

C. Mobile Progressive Web App

Converting KisanVyapar to a Progressive Web App (PWA) using service workers, a web app manifest, and cache-first strategies would enable: (a) home screen installation on Android and iOS without app store distribution; (b) offline browsing of cached product listings; (c) deferred form submission for order placement under intermittent connectivity; and (d) push notifications for order status updates and new messages. PWA technology is particularly well-suited to the rural Indian deployment context given its compatibility with entry-level Android devices.

D. Payment Gateway and Ratings System

Integrating a payment gateway (Razorpay, PayU, or the government's UPI-based BHIM API) would enable end-to-end transaction completion within the platform [20]. UPI integration is particularly appropriate given its widespread adoption across rural India and zero transaction fees for small merchants. A mutual ratings system allowing farmers to rate merchants on payment reliability and merchants to rate farmers on produce quality would further

build the trust infrastructure necessary for sustained platform adoption.

E. Image-Based Crop Disease Diagnosis

A natural extension of KisanMitra's capabilities would be image-based crop disease diagnosis. Farmers could photograph affected crop leaves and upload the image to KisanMitra, which would use Claude's vision capabilities to analyze the image alongside the text description and provide a more accurate diagnosis than text description alone—a compelling demonstration of multimodal AI applied to agricultural advisory.

X. CONCLUSION

This paper presented KisanVyapar, a full-stack web-based agricultural marketplace platform designed to directly connect Indian smallholder farmers with merchants, eliminating intermediaries and enabling fair, transparent, and communication-rich transactions. The system integrates five core capabilities within a unified, multilingual interface: secure role-based authentication, a dynamic product marketplace, lifecycle-managed order processing, peer-to-peer messaging, and KisanMitra—an AI-powered agricultural advisory chatbot operating in English, Hindi, and Marathi.

KisanMitra, implemented through Claude AI API integration with domain-grounding system prompt engineering and lightweight language detection, achieves 89.2% overall domain accuracy across five agricultural advisory categories and three languages without requiring specialized model training or annotated agricultural datasets. This approach demonstrates a practically deployable, cost-effective path to multilingual AI advisory in resource-constrained deployment environments.

The platform's LAMP stack architecture, automated database provisioning, and server-side rendering approach ensure broad deployability on standard shared hosting infrastructure—a critical factor for adoption by agricultural cooperatives, NGOs, and government agricultural extension programs in rural India. Performance evaluation confirmed all core operations complete within 400 ms, with the KisanMitra AI response averaging 2.1 seconds including external API round-trip. Functional testing across 42 test cases, two user roles, and three language settings confirmed zero critical failures and complete workflow coverage.

The agricultural digitization challenge in rural India is fundamentally a challenge of access—access to markets, access to knowledge, and access to technology. KisanVyapar addresses all three dimensions within a single, open-deployable platform. The authors hope that the design principles, implementation patterns, and evaluation results documented in this paper contribute to the growing body of applied research on technology-mediated agricultural development.

ACKNOWLEDGMENT

The authors thank the faculty of the Department of Computer Science and Engineering, Ajeenkya DY Patil University, for their guidance and support. Special thanks to the agricultural extension officers and domain experts who participated in the KisanMitra evaluation panel.

REFERENCES

[1] Ministry of Agriculture and Farmers Welfare, Government of India, "Agricultural Statistics at a Glance 2023," Directorate of Economics and Statistics, New Delhi, 2023.

[2] T. Reardon, K. Z. Chen, B. Minten, and L. Adriano, "The Quiet Revolution in Staple Food Value Chains," Asian Development Bank, Manila, 2012.

[3] S. Mittal and G. Tripathi, "Role of Mobile Phone Technology in Improving Small Farm Productivity," Agricultural Economics Research Review, vol. 22, pp. 451–459, 2009.

[4] P. S. Birthal, P. K. Joshi, and A. Gulati, "Vertical Coordination in High-Value Food Commodities: Implications for Smallholders," IFPRI Discussion Paper No. 85, Washington DC, 2005.

[5] M. Fafchamps and B. Minten, "Returns to Social Network Capital Among Traders," Oxford Economic Papers, vol. 54, no. 2, pp. 173–206, 2002.

[6] R. Meena, A. Kumar, and S. Sharma, "Digital Agriculture Platforms in India: A Review of Emerging Models and Adoption Challenges," Indian Journal of Agricultural Economics, vol. 76, no. 2, pp. 215–232, 2021.

[7] A. Sharma and P. Jain, "Conversational AI for Agricultural Advisory: A Multilingual Chatbot Approach for Rural India," in Proc. IEEE Int. Conf. Artificial Intelligence in Agriculture, 2022, pp. 112–119.

[8] S. Khandelwal, R. Mehra, and V. Narayanan, "Large Language Models for Agricultural Domain Advisory: Accuracy and Alignment in Low-Resource Settings," arXiv:2403.11287, 2024.

[9] A. Kunchukuttan, P. Mehta, and P. Bhattacharyya, "The IIT Bombay English-Hindi Parallel Corpus," in Proc. LREC, Miyazaki, Japan, 2018, pp. 3473–3479.

[10] A. Joshi, P. Kaur, and M. Singh, "Role of ICT in Agricultural Development of Rural India," Int. Journal of Agricultural Science and Research, vol. 8, no. 3, pp. 63–72, 2018.

[11] T. Bhattacharya and S. Roy, "E-Commerce Adoption in Indian Agriculture: Barriers and Enablers," Journal of Rural Development, vol. 39, no. 4, pp. 512–531, 2020.

[12] J. Nielsen, "Response Times: The 3 Important Limits," Nielsen Norman Group, 1993. [Online]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/

[13] NSSO, "Key Indicators of Situation of Agricultural Households in India: NSS 70th Round," Ministry of Statistics and Programme Implementation, Govt. of India, 2013.

[14] R. Kumar, A. Singh, and P. Sharma, "Impact of Digital Advisory Services on Smallholder Farmer Decision-Making in Madhya Pradesh," Journal of Agricultural Informatics, vol. 12, no. 1, pp. 1–15, 2021.

[15] P. Goyal, K. Sharma, and A. Jain, "Language Identification for Devanagari-Script Indian Languages Using Vocabulary-Based Classification," in Proc. ICON, 2020, pp. 88–95.

[16] Ministry of Electronics and Information Technology, "India's Trillion-Dollar Digital Opportunity," McKinsey Global Institute Report, Govt. of India, 2019.

[17] S. Nakasumi, "Information Sharing for Supply Chain Management Based on Block Chain Technology," in Proc. IEEE 19th Conf. Business Informatics, 2017, vol. 1, pp. 140–149.

[18] OWASP Foundation, "OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/Top10/

[19] D. Kathuria and V. S. Bashir, "Determinants of ICT Adoption in Indian Agriculture: Evidence from Farm Household Survey," Telecommunications Policy, vol. 44, no. 1, pp. 1–14, 2020.

[20] B. Ghosh and P. Mohanty, "Digital Financial Inclusion in Rural India: Role of UPI and Mobile Banking in Agricultural Communities," Economic and Political Weekly, vol. 57, no. 14, pp. 45–53, 2022.