ISSN: 2455 - 1597

ijergs
Open Access

International Journal of Engineering Research and Generic Science (IJERGS)
Available online at: https://www.ijergs.in

Volume - 12, Issue - 3, May – June – 2026, Page No. 66 – 75

MediDrop: Design and Implementation of an AI-Integrated Online Medicine Delivery Platform Using the MERN Stack with Real-Time Order Tracking and Prescription Verification

[1]Charu Sharma, Department of MCA, IIMT College of Engineering, Greater Noida

[2]Khushboo Koyal, Department of MCA, IIMT College of Engineering, Greater Noida

[3]Dr. Naveen Kumar Sharma, Department of MCA, IIMT College of Engineering, Greater Noida

Abstract

The healthcare and pharmaceutical sectors are undergoing rapid digital transformation propelled by the proliferation of e-commerce, mobile computing, and artificial intelligence. This paper presents MediDrop, a full-stack online medicine delivery platform developed using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack. The proposed system bridges the critical gap between patients' need for prompt pharmaceutical access and the inherent limitations of traditional brick-and-mortar pharmacy models. MediDrop integrates a prescription upload and administrative verification module, real-time GPS-based order tracking powered by Socket.io, an AI-driven health chatbot built on the OpenAI API, multi-gateway payment processing via Stripe and Razorpay, and role-based access control (RBAC) for users, administrators, and delivery agents. The platform encompasses prescription medicines, over-the-counter drugs, herbal supplements, dietary foods, and medical equipment. Empirical evaluation demonstrates sub-200 ms API response latency under concurrent load, 99.7% order-status notification delivery, and successful real-time location propagation at intervals below three seconds.

Keywords: MERN Stack, Online Medicine Delivery, Prescription Verification, Real-Time Tracking, Socket.io, AI Chatbot, OpenAI, Role-Based Access Control, Cloudinary, Razorpay, Stripe, Redux Toolkit, RESTful API, MongoDB.

Introduction

The global pharmaceutical e-commerce market was valued at approximately USD 107 billion in 2023 and is projected to exceed USD 280 billion by 2030, reflecting a compound annual growth rate exceeding 14% [1]. This substantial growth is fuelled by increasing smartphone penetration, rising demand for home-delivery convenience, and the widespread adoption of telemedicine following the COVID-19 pandemic. Despite this momentum, a significant proportion of medicine delivery platforms in developing economies lack critical safety mechanisms—particularly prescription verification—leaving patients vulnerable to the misuse of controlled substances.

Traditional pharmacy management systems are constrained by monolithic architectures, poor scalability, the absence of live tracking capabilities, and an inability to integrate intelligent conversational agents for preliminary medical guidance. MediDrop is a cloud-ready, full-stack platform that manages the entire pharmaceutical supply chain from prescription upload and administrative verification through cart management, payment processing, real-time delivery tracking, and AI-powered post-purchase support.

The key contributions of this work are: (1) a secure prescription upload and multi-level administrative verification workflow; (2) an event-driven real-time tracking subsystem using Socket.io WebSocket channels; (3) an AI-powered pharmaceutical chatbot via the OpenAI API; (4) a dual-gateway payment infrastructure supporting Stripe and Razorpay

with idempotent webhook handling; and (5) a role-based access control model encompassing end-users, administrators, and delivery agents.

Literature Review

A substantial body of research has addressed aspects of online medicine delivery. This section surveys relevant prior work across four dimensions: architecture and technology stack, prescription management and safety, real-time logistics, and AI-assisted healthcare interfaces.

A. Architecture and Technology Stack

Chandra et al.[2] demonstrated that MERN-based e-commerce applications exhibit superior throughput and developer productivity compared to traditional LAMP stacks. Reddy and Vasan[3] reported a 37% reduction in query latency for nested document retrieval when using MongoDB over relational MySQL for pharmaceutical product catalogs. Kumar et al.[4] implemented a healthcare portal using React.js and Node.js with JWT-based authentication, though their implementation lacked prescription verification and real-time tracking.

B. Prescription Management and Safety

Fernandez and Okafor[5] examined electronic prescription system requirements under India's Drugs and Cosmetics Act, 1940, proposing a multi-tier approval workflow. MediDrop simplifies this to a single admin-tier verification appropriate for B2C delivery while retaining the recommended audit trail. Singh et al.[6] analyzed CNN-based prescription image forgery detection, achieving 94.3% classification accuracy on 12,000 prescriptions; these findings inform MediDrop's roadmap for automated forgery detection.

C. Real-Time Logistics and Tracking

Mehta and Pillai[7] evaluated Socket.io against short-polling and Server-Sent Events for logistics tracking and concluded that Socket.io achieves 60% lower end-to-end latency and 45% lower server CPU utilization. Liu et al.[8] proposed a geofence-based push notification scheme for last-mile delivery, a concept identified as a near-term enhancement for MediDrop's tracking module.

D. AI-Assisted Healthcare Interfaces

Brown et al.[9] demonstrated that GPT-series models achieve near-pharmacist-level accuracy on standardized drug interaction queries when supplied with structured prompting. MediDrop leverages the OpenAI API with carefully engineered system prompts to restrict responses to the pharmaceutical domain, mitigating hallucination risks inherent to generative language models.

Problem Statement

Existing online pharmacy platforms fall short across multiple dimensions. The absence of prescription verification permits unchecked purchase of controlled medicines, violating pharmaceutical regulations. Poor real-time delivery visibility results in degraded user experience and elevated support ticket volumes. Standard FAQ chatbots cannot address drug-specific queries outside business hours. Single-gateway payment systems exclude users preferring UPI and net-banking. Legacy relational schemas and server-rendered frontends cannot efficiently accommodate concurrent pharmacy

delivery loads. Prescription images stored locally without cloud redundancy expose patient data to loss and unauthorized access. MediDrop addresses each of these gaps through deliberate architectural decisions.

Proposed System

MediDrop is proposed as a comprehensive, cloud-ready pharmaceutical delivery ecosystem comprising seven core functional modules.

A. User Authentication and Profile Management

The authentication subsystem implements JWT-based stateless authentication. Upon successful credential validation, the server signs a token with a 30-day expiry using the HS256 HMAC algorithm. Tokens are validated by the protect middleware on all secured routes. User documents in MongoDB store bcrypt-hashed passwords with a salt factor of ten, preventing plaintext exposure in the event of database compromise. Embedded address sub-documents allow users to maintain multiple labeled delivery addresses.

B. Prescription Upload and Administrative Verification

Users requiring prescription medicines upload images through the platform. The frontend accepts JPEG, PNG, and PDF formats up to 5 MB. Images are transmitted via multipart/form-data, processed through Multer middleware, and stored in Cloudinary's secure cloud storage. The administrative dashboard provides image preview, approval, and rejection controls. Upon approval, the corresponding medicine SKU is programmatically added to the user's active cart, enforcing the pharmaceutical safety guarantee.

C. Product Catalogue and Category Management

The product catalogue is organized into four primary categories: prescription medicines; over-the-counter drugs and dietary foods; herbal and Ayurvedic products; and medical tools and devices. Each product document stores name, description, category, price, stock quantity, image URL, and a boolean requiresPrescription flag. The frontend search and filter functionality operates via parameterized API queries supporting full-text search and category-based filtering.

D. Cart and Order Management

The cart is persisted server-side in MongoDB, associating cart documents with user identifiers to enable cross-device continuity. Order creation atomically deducts stock quantities, ensuring data consistency under concurrent requests. Orders progress through a defined state machine: Pending → Confirmed → Dispatched → Out for Delivery → Delivered, with each transition emitting Socket.io events and triggering multi-channel notifications via SMTP email and Twilio SMS.

E. Real-Time Delivery Tracking

The tracking subsystem employs Socket.io with room-based channel isolation. Each order constitutes a distinct room identified by its MongoDB ObjectId. When a delivery agent updates their GPS coordinates, the server emits an orderLocation event to the order room. The React frontend renders the agent's position on a Leaflet.js map, updating the marker at each event receipt, achieving sub-3-second location propagation without polling overhead.

F. AI Health Chatbot

The chatbot queries the OpenAI Chat Completions API (gpt-3.5-turbo) with a constrained system prompt restricting responses to pharmaceutical and general health information while advising consultation with a licensed physician for specific medical decisions. The backend acts as a secure proxy, maintaining the API key server-side and preventing client-side credential exposure.

G. Payment Gateway Integration

Stripe handles international card payments through PaymentIntents with idempotency keys and webhook validation via stripe-signature header verification. Razorpay processes domestic Indian payments including UPI and net-banking through the Orders API with HMAC-SHA256 signature validation. Both gateways update order status upon receiving payment confirmation webhooks, independent of client-side confirmation messages.

Methodology

A. Requirements Engineering

Functional requirements were gathered through structured interviews with end-users, pharmacists, and delivery operations personnel. Non-functional requirements included API response times below 300 ms at the 95th percentile, 99.5% availability, and OWASP Top 10 compliance.

B. System Design

The backend follows the Model-View-Controller (MVC) pattern; the frontend follows the Flux unidirectional data-flow pattern via Redux Toolkit. Data modeling employed entity-relationship analysis translated to MongoDB document schemas. RESTful API contracts were documented using the OpenAPI 3.0 specification.

C. Implementation

Sprint 1 delivered user authentication, product catalogue, and basic cart functionality. Sprint 2 implemented prescription upload, Cloudinary integration, and the admin verification dashboard. Sprint 3 integrated payment gateways and multi-channel notifications. Sprint 4 implemented Socket.io real-time tracking, the AI chatbot proxy, and the delivery agent dashboard.

D. Testing Strategy

Unit tests were authored for all controller functions using Jest and Supertest, achieving 78% code coverage. Integration tests exercised the prescription upload-verify-cart pipeline end-to-end. Load testing with k6 simulated 500 concurrent virtual users executing the checkout flow.

E. Deployment

The backend was containerized using Docker and deployed on a cloud VPS with Nginx as a reverse proxy and SSL termination via Let’s Encrypt. The frontend was deployed as a static build on Vercel with environment-variable-driven API URL configuration.

VI. System Architecture

The MediDrop architecture is structured across three tiers: the Client Tier, the Application Tier, and the Data Tier.

Table 1: Three-Tier System Architecture of Medidrop

Tier	Components
Client Tier	React.js 18, Redux Toolkit, Tailwind CSS, Vite; Pages: Home, Products, Cart, Checkout, Orders, Admin, Delivery, Profile
Application Tier	Node.js 18 + Express.js (MVC); Controllers: Auth, Admin, Product, Cart, Order, Payment, AI, Prescription; Middleware: JWT, RBAC, Multer, Error Handler
Data Tier	MongoDB Atlas (users, products, carts, orders, prescriptions, reviews); Cloudinary CDN (images)

A. Client Tier

The client tier is a Single-Page Application (SPA) built with React.js 18 using Vite as the build tool. Global state is managed by Redux Toolkit through five feature slices: authSlice, cartSlice, orderSlice, prescriptionSlice, and productSlice. Axios is the HTTP client, configured with a base URL interceptor injecting the Authorization header from the Redux store on every outbound request.

B. Application Tier

The application tier is a RESTful API server built on Node.js 18 LTS and Express.js 4.x. Routes are modularized by domain and mounted under versioned prefixes. The protect middleware validates JWTs; the authorise('admin') middleware enforces role-based restrictions on administrative endpoints. An asyncHandler higher-order function wraps all controller functions, propagating errors to the centralized error-handling middleware.

C. Data Tier

MongoDB Atlas hosts six primary collections. The User collection embeds an addresses array for O(1) address retrieval. The Order collection embeds an items array capturing a price snapshot at creation time, maintaining historical pricing accuracy. The Prescription collection stores Cloudinary public_id, secure_url, a status field (pending | approved | rejected), and an approvedBy administrator reference.

Table 2: illustrates the prescription upload and verification data flow.

User Uploads Prescription Image
↓
Multer Middleware (size/format validation)
↓
Cloudinary Secure Cloud Storage
↓
Prescription Document Created (status: pending)
↓
Administrator Reviews via Dashboard

[Approved] → Medicine Added to User Cart [Rejected] → User Notified via Email / SMS

Table 2: Prescription Upload and Administrative Verification Data Flow

Technologies Used

Table II summarizes the technologies and frameworks employed in MediDrop. MongoDB was selected for its document-oriented model, which naturally accommodates pharmaceutical product data with nested attributes such as dosage forms and drug interaction records. Socket.io was chosen for its automatic fallback to HTTP long-polling in network environments where WebSocket connections are restricted, ensuring tracking functionality across constrained mobile networks.

Table 3: Technology Stack Summary

Layer	Technology	Version	Purpose
Frontend	React.js	18.x	UI component framework (SPA)
Frontend	Redux Toolkit	1.9.x	Global state management
Frontend	Tailwind CSS	3.x	Utility-first responsive styling
Frontend	Vite	4.x	Build tool and dev server
Frontend	Axios	1.x	HTTP client with interceptors
Frontend	Leaflet.js	1.9.x	Interactive maps for live tracking
Frontend	Socket.io Client	4.x	WebSocket real-time communication
Backend	Node.js	18 LTS	Server-side JS runtime
Backend	Express.js	4.x	RESTful API framework
Backend	Socket.io	4.x	WebSocket server for live events
Backend	Mongoose	7.x	MongoDB ODM with validation
Database	MongoDB Atlas	6.x	Cloud NoSQL document store
Auth	jsonwebtoken	9.x	JWT generation and validation
Auth	bcryptjs	2.x	Password hashing (rounds=10)
Storage	Cloudinary	1.x	Cloud image hosting
Payment	Stripe	12.x	International card payments
Payment	Razorpay	2.x	Domestic UPI/Net-banking
AI	OpenAI API	GPT-3.5	AI pharmaceutical chatbot

Notify	Nodemailer	6.x	SMTP transactional email
Notify	Twilio	4.x	SMS order notifications
Upload	Multer	1.x	Multipart file upload middleware

Results and Discussion

A. Performance Benchmarks

Load testing with k6 employing 500 concurrent virtual users over a five-minute duration yielded the results presented in Table III. The 95th-percentile response time of 194 ms satisfies the sub-300 ms requirement. The 0.12% error rate was attributable to Razorpay API timeout in the sandbox environment. Following compound index additions on the orders and carts collections, MongoDB query time improved by 58% over the unindexed baseline.

Table 4: API Load Test Results (500 Concurrent Users)

Metric	Value
Median (p50) Response Time	87 ms
95th Percentile (p95) Response Time	194 ms
99th Percentile (p99) Response Time	312 ms
Sustained Throughput	487 req/s
Error Rate	0.12%
MongoDB Query Time (p95)	43 ms

B. Real-Time Tracking Evaluation

The Socket.io subsystem was evaluated by simulating GPS coordinate emission at two-second intervals over a 3G-equivalent throttled network (10 Mbps downlink, 50 ms RTT). Median end-to-end latency from coordinate emission to map marker update was 67 ms, with a 99th-percentile latency of 218 ms. Concurrent tracking of 50 order rooms showed linear memory growth of approximately 4 KB per room, confirming feasibility of horizontal scaling via Redis-backed Socket.io adapters.

C. Notification Delivery

Order lifecycle email notifications were delivered with a 99.7% success rate across 500 test orders. SMS notifications via Twilio exhibited a 99.2% delivery rate, with failures attributed to unverified trial-account phone numbers. Production deployment with verified numbers is expected to achieve delivery rates within Twilio's published service-level guarantees.

D. AI Chatbot Accuracy

The AI chatbot was evaluated against 100 pharmaceutical queries across five categories: drug dosage, contraindications, drug-drug interactions, storage instructions, and general wellness. A qualified pharmacy professional rated responses on a

five-point scale, yielding a mean score of 4.1. The system prompt constraint effectively prevented diagnostic recommendations in 97 of 100 evaluated interactions.

E. Security Assessment

An OWASP Top 10 inspection confirmed: all MongoDB queries use Mongoose schema validation with no raw string concatenation (A03); JWT expiry and bcrypt hashing are implemented (A07); all order queries are scoped to the authenticated user's identifier preventing horizontal privilege escalation (A01); CORS is configured with an explicit allowlist and Helmet.js sets security headers (A05); Cloudinary prescription URLs use signed delivery with expiring tokens and payment credentials are excluded from logs (A02).

Advantages

MediDrop delivers several consequential advantages. The prescription verification gate prevents unauthorized sale of Schedule H and H1 drugs in alignment with the Drugs and Cosmetics Act, 1940. Real-time GPS tracking eliminates delivery status uncertainty. The stateless JWT model and horizontally scalable MongoDB Atlas cluster accommodate growth from hundreds to millions of users without architectural changes. Dual-gateway payment support (Stripe and Razorpay) accommodates both international users and domestic Indian users preferring UPI. The AI chatbot provides round-the-clock pharmaceutical query responses. Cloudinary's globally distributed CDN ensures prescription image availability with a 99.99% uptime SLA. A unified JavaScript ecosystem across frontend and backend reduces development context-switching, while the modular Express.js router architecture permits incremental feature additions without modifying existing modules.

Future Scope

A. Automated Prescription Verification via Computer Vision

Integration of a CNN-based prescription forgery detection model, such as EfficientNet-B4 (which achieves accuracy exceeding 94% in this domain [6]), will augment the manual review process. The pipeline would classify prescriptions as high-confidence genuine, high-confidence fraudulent, or uncertain (routed to human review), reducing administrative workload by an estimated 70%.

B. Subscription-Based Medicine Refills

A subscription module will allow patients on chronic medications to configure automatic refill intervals, triggering order creation and payment capture on a defined schedule with automated re-verification for repeat prescriptions within clinically defined validity windows.

C. Telemedicine Integration

A WebRTC-based video consultation module connecting patients with licensed physicians within the platform would enable in-app prescription generation. Physician-signed digital prescriptions would feed directly into the prescription verification workflow, creating a closed-loop digital healthcare experience encompassing consultation, prescription, and fulfillment.

D. Geofencing and Predictive ETA

Geofence triggers around delivery addresses will enable sub-one-minute customer alerts as delivery agents approach. Machine learning models trained on historical delivery time data can generate dynamic ETA predictions, substantially improving customer experience over static estimates.

E. Inventory Forecasting and Supplier Integration

Time-series forecasting models such as ARIMA and LSTM applied to order history can predict stock depletion and trigger automated purchase orders to pharmaceutical distributors via EDI or API integration, reducing stockout incidents.

F. Progressive Web Application and Native Mobile Clients

Converting the React SPA to a Progressive Web Application will enable offline browsing and background push notifications. Native iOS and Android applications built with React Native would leverage shared Redux business logic for a multi-platform release with minimal additional development effort.

G. Multi-Language Support and Accessibility

Internationalization via react-i18next will enable deployment across regional Indian markets (Hindi, Tamil, Kannada, Bengali) and international markets. WCAG 2.1 Level AA compliance will ensure accessibility for users with visual and motor impairments.

Conclusion

This paper presented MediDrop, a full-stack online medicine delivery platform engineered on the MERN technology stack. The system addresses the dual imperatives of pharmaceutical safety and operational efficiency through a prescription upload and multi-tier administrative verification workflow, real-time Socket.io-based delivery tracking, AI-powered pharmaceutical chatbot support, and dual-gateway payment processing. The platform's modular MVC backend, Redux Toolkit frontend state management, and cloud-native integrations collectively deliver a scalable, maintainable, and user-centric pharmaceutical e-commerce solution. Empirical evaluation confirmed sub-200 ms 95th-percentile API response times under 500 concurrent users, sub-70 ms median real-time tracking latency, 99.7% email notification delivery, and an AI chatbot accuracy rating of 4.1 out of 5.0. The system satisfies all functional and non-functional requirements established during requirements engineering and represents a production-ready foundation for planned enhancements including automated prescription verification, telemedicine integration, and predictive inventory management. As digital healthcare ecosystems continue to converge with artificial intelligence, IoT, and cloud computing, platforms such as MediDrop are positioned to transform pharmaceutical service delivery—improving medication adherence, reducing prescription errors, and expanding access to medicines in underserved geographies. The contributions documented in this paper provide a reproducible reference architecture for researchers and practitioners in healthcare informatics, full-stack web engineering, and pharmaceutical technology.

References

1. Grand View Research, "Pharmaceutical E-Commerce Market Size, Share and Trends Analysis Report," Market Research Report, San Francisco, CA, 2024.

2. R. Chandra, P. Sharma, and A. Verma, “Comparative Analysis of MERN and LAMP Stack for E-Commerce Web Applications,” Int. J. Comput. Appl., vol. 183, no. 12, pp. 14–21, Jun. 2021.
3. K. Reddy and S. Vasan, “Performance Benchmarking of MongoDB Versus MySQL for Pharmaceutical Product Catalogs,” in Proc. IEEE ICCCBDA, Chengdu, China, 2022, pp. 87–93.
4. A. Kumar, B. Jain, and C. Patel, “A Scalable Healthcare Portal Using Node.js and React with JWT Authentication,” Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 4, pp. 230–238, Apr. 2022.
5. M. Fernandez and E. Okafor, “Legal and Technical Framework for Electronic Prescription Systems Under India’s Drugs and Cosmetics Act,” J. Pharm. Policy Pract., vol. 15, no. 1, pp. 1–12, 2022.
6. H. Singh, D. Mehta, and R. Rao, “Prescription Forgery Detection Using Convolutional Neural Networks,” in Proc. IEEE ICIP, Kuala Lumpur, Malaysia, 2023, pp. 1245–1250.
7. P. Mehta and R. Pillai, “Real-Time Logistics Tracking: WebSocket vs. Polling Performance Analysis,” ACM SIGCOMM Comput. Commun. Rev., vol. 52, no. 3, pp. 44–51, Jul. 2022.
8. Y. Liu, W. Zhang, and X. Chen, “Geofence-Based Push Notification Architecture for Last-Mile Delivery Optimization,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 5, pp. 5201–5213, May 2023.
9. T. Brown et al., “Language Models are Few-Shot Learners,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, pp. 1877–1901, 2020.
10. S. Tilkov and S. Vinoski, “Node.js: Using JavaScript to Build High-Performance Network Programs,” IEEE Internet Comput., vol. 14, no. 6, pp. 80–83, Nov.–Dec. 2010.
11. A. Mesbah and A. van Deursen, “Migrating Multi-Page Web Applications to Single-Page Ajax Interfaces,” in Proc. 11th Eur. Conf. Softw. Maint. Reeng. (CSMR), Amsterdam, 2007, pp. 181–190.
12. D. Abramov, “Redux: Predictable State Container for JavaScript Apps,” Redux Documentation, ver. 4.2.0, 2022.
13. Stripe, Inc., “Stripe API Reference,” Stripe Developer Documentation, 2024.
14. Razorpay Software Pvt. Ltd., “Razorpay Payment APIs,” Razorpay Developer Documentation, 2024.
15. OpenAI, “GPT-3.5-Turbo Model Card,” OpenAI Technical Documentation, San Francisco, CA, 2023.
16. Socket.io, “Socket.IO Documentation,” ver. 4.7, 2024.
17. MongoDB, Inc., “MongoDB Atlas Architecture Guide,” MongoDB Documentation, 2024.
18. Cloudinary Ltd., “Cloudinary Media Management Documentation,” 2024.
19. OWASP Foundation, “OWASP Top Ten 2021,” OWASP Technical Report, Annapolis, MD, 2021.
20. L. Richardson and M. Amundsen, RESTful Web APIs, O’Reilly Media, Sebastopol, CA, 2013.