INTERNATIONAL JOURNAL OF PHARMACY WITH MEDICAL SCIENCES
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3068-2738
Vol.6, No. 2(1)(2026)
www.ijpams.com

Bhubaneswar Market Plex: A Hyperlocal Digital Marketplace Platform

Sumit Sourav Dash (2201298416)
Student, Dept. of CSE, GIFT Autonomous, Bhubaneswar

Manisha Mishra (2201298351)
Student, Dept. of CSE, GIFT Autonomous, Bhubaneswar

Prof. Nitu Singh
Professor/Asst. Professor, Dept. of CSE/CSEIT, GIFT Autonomous, Bhubaneswar

Abstract

Bhubaneswar Market Plex is a full-stack, hyperlocal digital marketplace platform specifically designed to serve the commerce needs of the residents of Bhubaneswar, Odisha. The platform addresses a significant gap in the digital commerce ecosystem of Tier-2 Indian cities: the absence of a single unified application that integrates three distinct commerce modes — a second-hand goods marketplace, a quick commerce grocery delivery service, and a full-featured e-commerce catalog — in a locally relevant context.

The rapid growth of India's digital economy has created strong demand for hyperlocal commerce platforms, yet large national platforms like Amazon, Flipkart, OLX, and Blinkit fail to reflect the specific geography, trade culture, and neighbourhood trust networks of individual cities like Bhubaneswar. This project is conceived to digitise and streamline all three commerce modes in one application, rooted in the geography and economy of Bhubaneswar with real locality names, local store profiles, and a trust-first marketplace design.

The platform supports three user roles: Buyer, Seller, and Administrator, each with role-appropriate access enforced via JSON Web Tokens (JWT) and bcrypt-hashed passwords. The Marketplace module provides OLX-style classified listings with condition grading (New, Like New, Good, Fair, Poor), price negotiation flags, and an administrator moderation queue that ensures all listings are reviewed before going live. The Quick Commerce module connects buyers with local Bhubaneswar grocery and essentials stores for fast delivery. The E-Commerce module provides a full product catalog with cart, wishlist, order management, star-rated reviews, and a flash sale system with countdown timers.

The platform is built using a modern, production-grade technology stack: React 18 with TypeScript for the frontend, Express 5 on Node.js 24 for the backend, PostgreSQL as the relational database accessed through Drizzle ORM, and TanStack React Query for server state management. The project is structured as a pnpm monorepo with seven shared workspace packages. An OpenAPI 3.0 specification serves as the authoritative API contract from which both React Query hooks and Zod validation schemas are auto-generated using Orval, ensuring end-to-end type safety across the entire application.

The database comprises ten relational tables supporting all commerce modules, user roles, order management, cart, wishlist, reviews, and promotional banners. The backend exposes fourteen API route groups with over sixty endpoints, and the frontend implements twenty-one screens including a comprehensive Admin Panel. The application demonstrates sound software engineering practices including monorepo architecture, contract-first API design, end-to-end TypeScript type safety, role-based access control, rate-limited authentication, and responsive UI design with Framer Motion animations.

Testing of the platform confirms that all security scenarios, functional requirements, and responsive design breakpoints are met. The system successfully processes user registration, authentication, product browsing, cart management, order placement and lifecycle tracking, marketplace listing with admin moderation, and full administrator CRUD operations. The platform represents a complete, locally relevant digital commerce solution for Bhubaneswar and a strong demonstration of modern full-stack web development practices at the undergraduate level.

Keywords: Hyperlocal commerce, Bhubaneswar Market Plex, full-stack web development, React, Express, PostgreSQL, OpenAPI, marketplace, quick commerce.

1. Introduction

The digital economy of India has expanded rapidly due to smartphone adoption, affordable mobile internet, and the widespread use of digital payment systems. E-commerce, quick commerce, and online classified marketplaces have become major digital commerce categories. However, many Tier-2 cities still depend on fragmented digital services

that are designed for national audiences rather than local communities. Bhubaneswar, the capital city of Odisha, has a growing technology ecosystem, a large student population, and active neighbourhood markets, yet the city lacks one unified digital platform that combines second-hand goods, grocery delivery, and a complete e-commerce catalog in a locally relevant manner.

Bhubaneswar Market Plex is designed to solve this gap by merging the important features of OLX, Blinkit, and Flipkart into a single full-stack application focused on Bhubaneswar. The platform uses real locality names such as Patia, Chandrasekharpur, Sahid Nagar, Jaydev Vihar, Nayapalli, Khandagiri, Infocity, Vani Vihar, Unit-4, and Rasulgarh, thereby making the experience familiar and useful for local buyers and sellers. Instead of forcing users to switch between multiple national applications, the system offers marketplace listings, quick commerce products, and e-commerce products in a unified interface.

The project is also a practical demonstration of modern software engineering. It uses a TypeScript-based monorepo, contract-first API design through OpenAPI 3.0, auto-generated React Query hooks, Zod validation schemas, JWT authentication, and a PostgreSQL database accessed through Drizzle ORM. These choices help maintain type safety, scalability, code reuse, and clean separation between the frontend, backend, and shared packages.

2. Problem Statement

Residents of Bhubaneswar lack a unified digital commerce platform that is specific to their geography, local store ecosystem, and peer-to-peer trust networks. Existing national platforms provide useful services, but they are fragmented. A buyer may use one application for second-hand goods, another for grocery delivery, and another for general e-commerce. This fragmentation creates friction and reduces the visibility of local stores and individual sellers.

- Geographic limitation: national platforms do not provide strong Bhubaneswar locality-level filtering for neighbourhood buying and selling.
- Trust limitation: classified listings often lack condition grading, moderation, and a structured workflow for quality control.
- Commerce fragmentation: users need multiple separate apps for daily commerce needs.
- Local store invisibility: small Bhubaneswar shops have limited presence on national e-commerce platforms.
- Seller empowerment gap: individual sellers and small merchants need dashboards to manage listings, products, and digital visibility.

3. Objectives

The primary objective of Bhubaneswar Market Plex is to design and implement a unified hyperlocal digital marketplace for Bhubaneswar that integrates three commerce modes: an OLX-style marketplace, a Blinkit-style quick commerce module, and a Flipkart-style e-commerce catalog. The platform is expected to support buyers, sellers, and administrators with role-appropriate access and workflows.

- To create a responsive web application where users can browse products, manage carts and wishlists, place orders, and track order status.
- To provide sellers with the ability to create marketplace listings, upload images, set item condition, and manage their listing status.
- To implement an administrator panel for approving or rejecting marketplace listings, managing users, products, categories, banners, and orders.
- To design a normalized relational database with tables for users, products, stores, categories, listings, orders, reviews, cart items, wishlist items, and banners.
- To apply secure authentication and authorization using JWT, bcrypt password hashing, rate limiting, and role-based access control.

4. Literature Survey and Gap Analysis

A study of existing commerce systems shows that each major platform solves only one part of the local commerce problem. OLX and Quikr provide classified listings, but they do not provide a structured condition grading system, reliable locality filtering, or a formal admin moderation queue for Bhubaneswar. Amazon and Flipkart provide large-scale e-commerce, but their logistics and catalog are national rather than hyperlocal. Blinkit and Zepto have shown the importance of fast delivery, but their service availability is concentrated in larger metro cities. Facebook Marketplace supports informal peer-to-peer selling, but it lacks structured order management, formal moderation, and platform-controlled trust mechanisms.

Hyperlocal commerce has become important because it combines geographic proximity with digital convenience. A locality-aware platform can connect buyers with known neighbourhood shops and nearby sellers, reducing delivery time and increasing trust. Bhubaneswar Market Plex addresses this gap by combining three major commerce models in one application while remaining focused on a specific city and its localities.

5. Comparative Analysis

Feature	OLX	Amazon	Blinkit	Facebook Mkt	BMP
Second-hand classifieds	Yes	No	No	Yes	Yes
Quick commerce groceries	No	Partial	Yes	No	Yes
Full e-commerce catalog	No	Yes	Partial	No	Yes
Local store integration	No	No	No	No	Yes
Condition grading	No	No	N/A	No	Yes
Admin moderation	No	Yes	Yes	No	Yes
Bhubaneswar locality filter	No	No	No	No	Yes

6. Proposed System

The proposed system follows a three-tier client-server architecture. The presentation tier is a React 18 single-page application. The application tier is an Express 5 REST API running on Node.js 24. The data tier is PostgreSQL, accessed through Drizzle ORM. A shared monorepo layer connects these tiers by storing common TypeScript types, database schemas, OpenAPI specifications, auto-generated API clients, and validation schemas.

The platform has three major functional modules. The Marketplace module allows users to sell second-hand items with condition grades such as New, Like New, Good, Fair, and Poor. The Quick Commerce module connects customers with local grocery and essential stores. The E-Commerce module provides a product catalog, product details, shopping cart, wishlist, review system, flash sale products, and order management. The Admin Panel provides centralized control over users, listings, products, categories, banners, and orders.

Role-based access is central to the proposed system. Buyers can browse products, manage carts, place orders, write reviews, and browse marketplace listings. Sellers can perform buyer operations and also create marketplace listings. Administrators can approve or reject listings, ban or unban users, and manage platform entities through CRUD operations.

7. Technology Stack

The technology stack was selected to provide type safety, maintainability, and production-grade application structure. TypeScript is used across the frontend and backend to catch integration errors at compile time. React 18 and Vite provide a fast and component-based frontend environment. TailwindCSS, ShadCN UI, and Framer Motion are used to build responsive and animated interfaces. Wouter provides lightweight client-side routing, while TanStack React Query manages server state, caching, loading states, and background refetching.

The backend uses Express 5 with Node.js 24. Express 5 supports improved async error handling and enables clear API route organization. PostgreSQL is used because of its reliability, ACID compliance, support for JSONB data, and suitability for relational commerce data. Drizzle ORM defines the database schema in TypeScript and generates type-safe database queries. OpenAPI 3.0 acts as the contract between the frontend and backend, and Orval generates React Query hooks and Zod schemas from that contract.

Category	Technology	Purpose
Frontend	React 18, TypeScript, Vite	SPA, component UI, fast builds
Styling	TailwindCSS, ShadCN UI	Responsive accessible interface
Animation	Framer Motion	Page and card animations
State	TanStack React Query	API caching and server state
Backend	Express 5, Node.js 24	REST API and business logic
Database	PostgreSQL, Drizzle ORM	Relational storage and type-safe queries
API Contract	OpenAPI 3.0, Orval, Zod	Generated clients and validation
Security	JWT, bcrypt, rate limit	Authentication and protection

8. Database Design

The database design contains ten main tables: users, categories, products, stores, listings, orders, reviews, cart_items, wishlist_items, and banners. The users table stores account information, role, phone number, avatar, and ban status. The products table supports e-commerce and quick commerce products, including price, discount, stock, store relation, category relation, rating, and flash sale flags. The stores table stores local Bhubaneswar store information such as area, delivery time, minimum order, and store type.

The listings table implements the second-hand marketplace. Every listing stores title, description, price, negotiable flag, condition grade, status, images, location, seller relation, and category relation. New listings start with pending status and become visible only when administrators approve them. This moderation workflow improves trust and reduces spam or low-quality listings.

The orders table uses a JSONB item snapshot instead of a separate order_items table. When an order is created, the system stores the product name, image, price, and quantity at the time of purchase. This design preserves historical order accuracy even if product prices or details are changed later. The cart_items and wishlist_items tables provide persistent user-specific shopping data across sessions.

9. System Architecture and Workflow

In the order placement workflow, the buyer first adds products to the server-side cart. During checkout, the frontend sends a POST request to the orders endpoint with the delivery address and payment method. The JWT middleware verifies the user token and extracts the user identity. The backend then retrieves all cart items for that user, fetches current product details, calculates the total, creates the order with a JSONB item snapshot, clears the cart, and returns the created order to the frontend. React Query then invalidates the cart cache and the interface displays an empty cart with the new order detail page.

The marketplace approval workflow is equally important. A seller creates a listing by entering item information, uploading images, selecting condition, setting the negotiable flag, and selecting a Bhubaneswar locality. The backend stores the listing as pending. Administrators view all pending listings in the Admin Listings screen and approve or reject them. Only active listings are visible in the public marketplace. This design creates a controlled trust layer for peer-to-peer transactions.

10. Implementation

The frontend contains twenty-one screens including Home, Products, Product Detail, Marketplace, Listing Detail, Cart, Wishlist, Orders, Order Detail, Stores, Store Detail, Sell, Login, Register, Seller Dashboard, Admin Dashboard, Admin Listings, Admin Users, Admin Products, Admin Categories, Admin Banners, and Admin Orders. Protected routes use an authentication guard that checks token presence, while admin routes additionally check the role claim.

The Home page presents a hero banner, category grid, flash deals, top stores, and featured products. Product cards show image, title, price, discount, rating, and action buttons. The Marketplace page displays listing cards with condition badges, location, price, and negotiation status. The Sell page allows sellers to create listings with up to five images and one of twelve Bhubaneswar localities. The Cart page shows a quantity stepper and checkout summary, while the Order Detail page shows an itemized order and a visual status timeline.

The backend is organized into route groups for authentication, products, marketplace, cart, wishlist, orders, reviews, stores, categories, banners, and admin functions. The authentication routes implement registration, login, and current-user retrieval. Product routes provide product listing, filtering, flash sale products, featured products, product

details, and admin product operations. The admin routes are protected using JWT verification and a requireAdmin middleware function.

11. Security Design

Security is implemented through a combination of stateless authentication, password hashing, role-based authorization, and request rate limiting. Passwords are never stored directly; they are hashed using bcrypt before being inserted into the users table. After login, the server returns a JWT containing the user's id, email, and role. The client sends this token as a Bearer token in the Authorization header for protected requests.

The backend verifies the JWT before executing protected operations. Invalid, expired, or missing tokens return HTTP 401. Admin-only routes return HTTP 403 when accessed by buyers or sellers. Login and registration endpoints are rate-limited to reduce brute-force attacks. The user ban feature allows administrators to suspend accounts, and banned users are blocked during login.

12. Testing and Results

The system was tested through functional, security, performance, and responsive design checks. Functional testing verified registration, login, product browsing, category filtering, flash sale retrieval, cart operations, wishlist toggling, order placement, order cancellation, product reviews, marketplace listing creation, listing moderation, store retrieval, and admin CRUD operations. All listed functional cases passed.

Security testing confirmed that valid users receive tokens, incorrect passwords are rejected, missing or malformed tokens are blocked, expired tokens are denied, buyers and sellers cannot access admin routes, rate limits are enforced, and banned users cannot log in. Performance testing showed acceptable local response times for common endpoints such as product listing, product detail, order placement, admin statistics, login, cart retrieval, and marketplace listing retrieval.

Area	Example Test	Expected Result	Status
Authentication	Register/login user	JWT returned	Pass
Products	Fetch and filter catalog	Correct product list	Pass
Cart	Add/update/remove item	Cart updates correctly	Pass
Orders	Place order from cart	Order created and cart cleared	Pass
Marketplace	Create seller listing	Pending status	Pass
Admin	Approve listing	Listing becomes active	Pass
Security	Buyer accesses admin route	HTTP 403	Pass
Rate Limit	21 login attempts	HTTP 429	Pass

12.1 Detailed Functional Modules

The Buyer module is designed around simple discovery and purchase workflows. A buyer can search for products, filter them by category, open product details, add products to the cart, save products to the wishlist, place orders, review purchased products, and track order status. The cart is stored on the server instead of only in browser memory, so it remains available across sessions after login. This improves reliability for users who switch devices or close the browser before completing checkout.

The Seller module extends buyer functionality with marketplace listing creation. A seller can enter the listing title, detailed description, price, negotiable status, product condition, category, images, and locality. The condition field provides structured quality information to the buyer, while the locality field makes nearby discovery easier. The seller dashboard gives the seller a basic control center for viewing personal listings and monitoring whether each listing is pending, active, sold, expired, or rejected.

The Administrator module is responsible for platform trust and data management. Administrators can approve or reject second-hand listings, manage suspicious users by banning or unbanning them, create and update products, manage categories, create promotional banners, and monitor orders. The Admin Dashboard displays major platform statistics, which helps administrators understand total users, total products, total listings, total orders, revenue, pending orders, active listings, and banned users.

12.2 API Design

The backend API is organized into route groups so that each business domain remains maintainable. The /auth routes manage registration, login, and current-user retrieval. The /products routes support product listing, search, filtering, featured products, flash sale products, product details, and administrative product operations. The /cart and /wishlist routes provide persistent shopping actions. The /orders routes handle order creation, order listing, order details, and order cancellation. The /marketplace routes manage classified listings and search. The /admin routes provide restricted administrative control over users, listings, products, categories, banners, orders, and platform statistics.

OpenAPI 3.0 is used as the source of truth for this API. Each endpoint is described with its method, path, input body, response body, response status codes, and security requirements. This contract-first design reduces ambiguity between frontend and backend development. When the API contract changes, generated TypeScript code changes as well, allowing errors to be detected during development rather than after deployment.

12.3 User Interface Design

The user interface follows a market-inspired design that is bright, responsive, and action oriented. The Home page guides users to the three main commerce experiences through category cards, flash sale products, local stores, and featured products. Product cards are designed to show the most important information quickly: image, title, price, discount, rating, and action buttons. The Marketplace listing cards highlight condition badges and localities because these two details are most important for second-hand buying decisions.

Responsive behavior is included for desktop and mobile browser sizes. Product grids reduce the number of columns on smaller screens, navigation elements adapt to available width, and form components remain readable on mobile devices. Animation is used carefully to improve user engagement without affecting clarity. Framer Motion is applied to page transitions, card hover effects, and flash sale countdown interactions.

12.4 Data Integrity and Maintainability

Data integrity is handled through relational design, foreign keys, controlled status values, and validation schemas. Products are related to stores and categories. Listings are related to sellers and categories. Orders are related to users and store item snapshots. Reviews are connected to users and products. Cart and wishlist rows are connected to users and products. This design makes the database understandable and prevents many inconsistent records.

Maintainability is improved through the monorepo approach. Shared database schemas, common API types, generated clients, and validation schemas are kept in workspace packages rather than copied manually. This structure is valuable because the same concept is often needed in multiple layers of the application. For example, a product response type is needed by the backend route, the API specification, the frontend hook, and the product card component. Sharing and generating these definitions reduces mistakes.

12.5 Deployment Considerations

For deployment, the application can be separated into three main services: a static frontend build, a Node.js backend API, and a managed PostgreSQL database. The frontend can be hosted on platforms such as Vercel, Netlify, or any static hosting service. The backend can be hosted on a Node.js-compatible platform, and the database can use a managed PostgreSQL provider. Environment variables are required for the database connection string, JWT secret, frontend URL, backend port, and storage configuration.

A production deployment should also include HTTPS, secure cookie or token handling policies, database backups, request logging, error monitoring, and centralized storage for uploaded images. Since marketplace images and product images are important to user trust, storing them in an object storage service such as AWS S3, Cloudflare R2, or similar infrastructure would be better than local file paths. Monitoring should track failed logins, API latency, database errors, and unusual admin actions.

12.6 Practical Impact

The practical value of Bhubaneswar Market Plex is that it brings digital visibility to local commerce. Small stores can present products online without building an independent website. Students and residents can sell second-hand items within known localities. Buyers can discover nearby shops and products through one interface. Administrators can maintain quality through moderation and user management. This combination makes the platform suitable not only as an academic project but also as a prototype for a real city-level commerce solution.

The platform can also help reduce dependency on national applications for every small commerce requirement. A local-first system can encourage faster delivery, better communication, higher trust, and more relevant recommendations. Since the system is designed around a modular architecture, more cities could be added later by changing locality data, store profiles, and city branding while preserving the main technical foundation.

12.7 Key Algorithms and Business Rules

Several business rules are implemented to keep the platform consistent. During registration, the system checks whether the email already exists before creating a new account. During login, the system checks the password hash and also verifies whether the account is banned. During marketplace listing creation, the system does not publish the listing immediately; it stores the listing with pending status. This ensures that administrators can review title, description, price, images, condition, and locality before public visibility.

During order placement, the system follows a transactional sequence. It loads the user's cart, validates that the cart is not empty, fetches the latest product records, calculates total price, creates the order snapshot, and clears the cart. This sequence prevents incomplete orders and ensures that order history remains stable. During order cancellation, the system checks the current status and allows cancellation only for pending orders. Orders that are already packed, picked, on the way, or delivered are protected from buyer-side cancellation.

The admin statistics feature applies parallel database queries so that dashboard values can be loaded quickly. Instead of waiting for each count sequentially, the backend requests user count, product count, listing count, order count, and revenue calculation at the same time. This improves perceived performance for administrators and keeps the dashboard responsive even when the platform data grows.

12.8 Validation and Error Handling

Validation is performed before data reaches the core business logic. Request bodies for registration, login, product creation, listing creation, review submission, and order placement are checked against schemas. Invalid inputs return clear error responses, which prevents malformed data from entering the database. This is important in a commerce system because invalid prices, missing product ids, empty addresses, or incorrect roles can create serious inconsistencies.

The backend uses a global error handler to return structured JSON error messages. Authentication failures return Unauthorized responses, permission failures return Forbidden responses, duplicate or invalid inputs return Bad Request responses, and missing resources return Not Found responses. This predictable response pattern helps the frontend show useful messages to users and also simplifies debugging during development.

12.9 Limitations of Current Version

Although the project is functionally complete for demonstration, it still has practical limitations. The payment flow is simulated and does not process real transactions. The delivery system does not include real-time GPS tracking or integration with delivery partners. Product images and listing images are not stored in a dedicated production object storage service. The platform does not yet support push notifications, live buyer-seller chat, or automatic fraud detection. The interface is English-only, so Odia language support would be important for wider adoption in Odisha.

These limitations do not reduce the value of the system as an academic project, because the core architecture already supports extension. Payment gateways, object storage, notification services, delivery tracking, and multilingual user interfaces can be added as future modules without redesigning the entire system. The current version therefore acts as a strong base prototype for further development.

13. Discussion

Bhubaneswar Market Plex shows that a city-specific commerce platform can combine local relevance with modern full-stack architecture. The system does not simply clone existing national platforms; it adapts their useful ideas to the specific needs of Bhubaneswar. The locality selection, condition grading, seller dashboard, and administrator moderation are important features for local trust. The quick commerce module supports the idea that neighbourhood stores can become digital sellers without building their own independent applications.

From a technical perspective, the monorepo structure reduces duplication and improves consistency. The OpenAPI contract ensures that frontend and backend development remain aligned. Orval-generated hooks reduce manual API code, while Zod schemas provide consistent input validation. The JWT-based authorization system and admin guard provide a secure role-based foundation for future expansion.

The current version also has limitations. Payment processing is simulated, so real financial transactions are not supported. Image storage uses file path references rather than a production CDN or object storage service. The platform supports English only and does not yet include Odia localization. Real-time GPS tracking, push notifications, and live chat are also not included in the present version.

14. Conclusion and Future Work

Bhubaneswar Market Plex successfully implements a full-stack hyperlocal marketplace that integrates classified listings, quick commerce, and e-commerce in a single application for Bhubaneswar. The platform supports buyer, seller, and administrator roles; provides secure authentication; offers server-side cart and wishlist persistence; enables order placement and tracking; and includes an admin panel for moderation and management. The project demonstrates practical use of React, Express, PostgreSQL, Drizzle ORM, OpenAPI, Orval, and TypeScript in a production-style undergraduate project.

Future enhancements may include real payment integration using Razorpay, PhonePe, or UPI; native Android and iOS applications; real-time GPS delivery tracking; WebSocket-based buyer-seller chat; Firebase push notifications; Odia language support; product recommendation systems; analytics dashboards for sellers; and production-grade cloud storage for images. These improvements would make the platform more deployable and commercially useful for Bhubaneswar's local commerce ecosystem.

References

[1] React Documentation, React 18: Building User Interfaces with Components.

[2] Express.js Documentation, Express 5 Web Framework for Node.js.

[3] PostgreSQL Documentation, The World's Most Advanced Open Source Relational Database.

[4] Drizzle ORM Documentation, TypeScript ORM for SQL Databases.

[5] OpenAPI Initiative, OpenAPI Specification Version 3.0.3.

[6] Orval Documentation, TypeScript Client and React Query Generator.

[7] TanStack Query Documentation, Server State Management for React.

[8] Bhubaneswar Market Plex Major Project Report, Department of Computer Science and Engineering, GIFT Autonomous, Bhubaneswar.

[11]Babburi, S. (2023). Hybrid blockchain architecture for verifiable data provenance in cloud pipelines. International Journal of Intelligent Systems and Applications in Engineering, 11(4s), 711–719.

[12] Gaddam, S. From Fixed Specifications to Self-Adapting Systems: A Machine Learning Perspective on Software Engineering.

[13] Immadi, S. K. (2025). Optimizing ERP for Human Capital Management. Applied Research for Growth, Innovation and Sustainable Impact, 377–384. https://doi.org/10.1201/9781003684657-63

[14] Poojari, R. INTELLIGENT SYSTEMS+B108 AND APPLICATIONS IN ENGINEERING.

[15] Poojari, R. Frameworks for Data Management and Lineage in Large-Scale Healthcare Data Systems.

[16] Poojari, R. Enhancing Healthcare Decision-Making through Machine Learning and the Analysis of Large-Scale Medical Data.

[17] Mahimalur, R. K., Vasgam, M., & Manoharan, D. (2025). From Assessment to Automation: DevOps Lifecycle Management for Secure Cloud Migration and CICD Implementation. Power System Technology, 49(3).

[18] Purmani, S. S. R. (2025). Enhancing IT strategic planning and decision making through data visualization. International Journal of Enhanced Research in Management & Computer Applications, 14(4), 75–81

[19] Purmani, S. S. R., & Kotte, G. Intelligent Project Orchestration: How Generative AI is Reshaping Go-to-Market Strategy Planning and Cross-Functional Delivery. environments, 4, 5.

[20] Cyril, H. P., & Kumara, S. Identification of Anomalies via Deep Learning-Based Models for High-Dimensional Telecom Traffic Data.

[21] Kotte, G. (2025). Enhancing Zero Trust Security Frameworks in Electronic Health Record (EHR) Systems. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283668

[22] Kotte, G. (2025). Revolutionizing Stock Market Trading with Artificial Intelligence. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5283647

[23] Viswanathan, V. (2025). Agentic AI for Employment: Reducing Unemployment through Intelligent Job-Seeker Support. LEX LOCALIS–Journal of Local Self-Government.

[24] Viswanathan, V., Shah, A. K., Kubam, C. S., Dontu, S., Gandhi, A., & Singla, P. (2025, August). Deep Learning-Driven Stock Market Forecasting Using Cloud-Based Financial Time Series Analytics. In 2025 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) (pp. 1-6). IEEE.

[25] Mudusu, S. K. (2026, March 26). A data trust scoring framework for reliable and responsible AI systems. InfoWorld (Foundry Expert Contributor Network).

[26] Mudusu, S. K. (2025, June 3). Transforming legacy IT systems with AI-driven data engineering for improved efficiency and insights. Hampton Global Business Review (HGBR).

[27] Gajula, S. (2025). Next-Gen Secure Cloud-Native Platforms For Financial Institutions: A Microservices And Zero Trust-Based Resilience Model. Journal of International Crisis & Risk Communication Research (JICRCR), 8.

[28] Gajula, S., & Margam, M. (2026, February). A Secure and Scalable Cloud-Based Banking Service Model Leveraging AI and Advanced Cyber Security. In 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC) (pp. 1-5). IEEE.

[29] Maturi, S. Y. (2025). Blockbond Hardening: Securing Pooled-Hash Protocols Against Traffic Tampering, MITM Hash-Rate Hijacking, and Template Coercion. https://doi.org/10.20944/preprints202512.2064.v1

[30] Maturi, S. Y. (2025). Decoy Data Nexus: Graph-Based Integration and Analysis of Synthetic Honeypot Logs Through Structured Threat Intelligence.

[31] Ranjbareslamloo, S., Dzukey, G. A., Islam Muhit, M. M., & Qattawi, A. (2025). Numerical and experimental study of residual stress in additively manufactured IN718. Manufacturing Letters, 44, 915–927. https://doi.org/10.1016/j.mfglet.2025.06.108

[32] Manoharan, D. (2026). Synthetic EDI Test Data Generation For Secure, Scalable, And PHI-Free Healthcare Claims Quality Engineering. Journal of International Crisis and Risk Communication Research, 9(1).

[33] Venkata Ramana, P. (2024). AI-driven predictive analytics in ERP systems for proactive supply chain optimization. International Journal of Research in Information Technology and Computing, 8(4).

[34] Pavan Kumar Adabala. (2026). IoT-Driven Digital Twins for Manufacturing Optimization: Hybrid Modelling, Reinforcement Learning and Sustainable Operations. International Journal of Computational and Experimental Science and Engineering, 12(1). https://doi.org/10.22399/ijcesen.5050

[35] Pavan Kumar Adabala. (2026). Best Practices for Enterprise System Integration in Modern Organizations. Journal of Information Systems Engineering and Management, 11(2s), 1137–1146. https://doi.org/10.52783/jisem.v11i2s.14558

[36] Srikanth Kavuri. (2024). Probabilistic Generative Modeling for Synthesizing High-Coverage Test Data in Safety-Critical Software Applications. Computer Fraud and Security, 633–642. https://doi.org/10.52710/cfs.838

[37] Srikanth Kavuri. (2022). Large Language Model (LLM)-Based Automation for Software Test Script Generation. Computer Fraud and Security. https://doi.org/10.52710/cfs.836

[38] Venkata Pavan Kumar Gummadi. (2023). MuleSoft Batch Processing: High-Volume Streaming Architecture. Computer Fraud and Security, 50–57. https://doi.org/10.52710/cfs.886

[39] Venkata Pavan Kumar Gummadi. (2026). Infrastructure Optimization Techniques for Enterprise Integration Platforms: A Comprehensive Analysis. Computer Fraud and Security, 37–44. https://doi.org/10.52710/cfs.875

[40] Venkata Pavan Kumar Gummadi. (2024). API Design and Implementation: RAML and OpenAPI Specification. Journal of Electrical Systems, 16(4), 76–85. https://doi.org/10.52783/jes.9329

[41] Venkata Pavan Kumar Gummadi. (2025). MuleSoft's Role in Advancing Sustainable Digital Infrastructure: An Enterprise Integration Perspective. Journal of Information Systems Engineering and Management, 10(62s), 1313–1321. https://doi.org/10.52783/jisem.v10i62s.13783

[42] Gummadi, V. P. K., Chilamkurthi, L. S., & Kavuri, S. (2026). Service Level Objective (SLO) Observability with Splunk and Dynatrace in Microservices. 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET), 1–4. https://doi.org/10.1109/icaiset66439.2026.11541542

[43] Pokala, H. K., & Gummadi, V. P. K. (2026). Autonomous AI-Powered Resource Management for Apache Flink on Amazon EKS. 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET), 1–4. https://doi.org/10.1109/icaiset66439.2026.11541881

[44] Gajula, S. (2025). Cloud transformation in financial services: A strategic framework for hybrid adoption and business continuity. International Journal of Scientific Research in Computer Science, Engineering and Information technology.

[45] Gajula, S. (2025). Cybersecurity in Supply Chain Management: Role of Identity and Access Management, Zero Trust, and Blockchain. Asian Journal of Computer Science Engineering (AJCSE), 10(2), 1-11.

[46] Gajula, S. (2026). Two Pillars of Banking Intelligence: A Comparative Analysis of AI Techniques for Fraud Prevention and Churn Mitigation. 2026 14th International Symposium on Digital Forensics and Security (ISDFS), 1–6. https://doi.org/10.1109/isdfs69419.2026.11458995