2026 Volume: 02 Issue: 01 www.ijrpa.com ISSN 2456-9995 Review Article

International Journal Research Publication Analysis

Page: 01-11

E-COMMERCE ONLINE RETAIL SYSTEM

***Kuldeep Kashyap, Vivek Kasana, Shaheen Usmani**

Department of Computer Science & Engineering, Galgotias University, Uttar Pradesh, India.

Article Received: 13 December 2025
Article Revised: 1 January 2026
Published on: 20 January 2026

***Corresponding Author: Kuldeep Kashyap**
Department of Computer Science & Engineering, Galgotias University, Uttar Pradesh, India.

DOI: https://doi-doi.org/101555/ijrpa.4482

ABSTRACT

The design and development of a contemporary e- commerce retail system utilizing the newest trends and technologies is examined in this report. We suggest a headless, microservices-based architecture that is hosted on AWS cloud infrastructure and consists of a React front end, a Node.js back end, and MongoDB data stores. The system incorporates AR/VR features (e.g., virtual try-on, 3D product visualization) to improve user experience and integrates AI/ML for real-time personalization (recommendations, search, chatbots). Other essential elements include mobile/omnichannel commerce (PWA, mobile apps, and social commerce) and secure payment processing (e.g., virtual Stripe/PayPal with 3D Secure and fraud detection). The architecture decouples the front and back ends for scalability and agility using an API- driven methodology (GraphQL/REST). Findings show that a platform like omnichannel can increase conversion and retention. For instance, omnichannel shoppers increase lifetime value by about 30%, and AI personalization "delights shoppers" while increasing sales conversions. We wrap up by outlining the advantages (personalization, flexibility, worldwide reach) and drawbacks (data complexity, cost, privacy/compliance).

KEYWORDS: online shopping platform, online retail systems, and e-commerce.

INTRODUCTION

In recent years, e-commerce has experienced rapid growth. For example, US online retail sales exceeded $1 trillion in 2022, and the demand for easy, customized shopping is driving global growth. Retailers are adopting contemporary architectures and technologies in order to stay competitive. Specifically, the preferred tech stack is now MACH (Microservices, API-first, Cloud- native, Headless). This change can be seen in large platforms like Amazon and Flipkart, where dozens of Spring Boot microservices on AWS have replaced monolithic early

systems. In a similar vein, highly responsive user interfaces across devices are made possible by headless commerce, which decouples front and back ends. Recent trends highlight further innovations. E- commerce increasingly relies on AI/ML-driven personalization–using machine learning to tailor product recommendations, content, pricing, and chatbots in real time. Likewise, AR/VR technologies are moving from novelty to mainstream: customers can virtually try on products or tour simulated stores. At the same time, shoppers use multiple channels (web, mobile, social, in-store) and expect unified experiences. This omnichannel demand motivates headless/Uber-model platforms that synchronize inventory, personalization, and branding everywhere.

Objectives

The system's objectives are to create an advanced online retail platform with the following features:

Adaptable and Scalable Architecture: Use an AWS headless microservices-based backend to handle heavy traffic and quick feature launches. Services will be able to be independently deployed, allowing for ongoing delivery and integration.

AI/ML Personalization: Use machine learning models to offer dynamic content, search personalization, and product recommendations. For "recommended for you" items, for instance, use custom TensorFlow models or Amazon Personalize.

RELATED WORK

We have exhibited a state-of-the-art e-commerce platform that uses AWS, Node.js, MongoDB, and React to provide a scalable and adaptable online retail solution. The solution facilitates quick innovation and omnichannel experiences by implementing a headless, microservices architecture (in accordance with MACH principles). Incorporating AR/VR for immersive purchasing and AI/ML for hyperpersonalization is in line with new trends. Reliability and commercial viability are ensured via cloud deployment and secure payment features.

There are still certain restrictions and difficulties, though. First, it takes a lot of knowledge and resources to construct such a system: DevOps for cloud orchestration, data pipelines for machine learning, and 3D content production for augmented reality all add complexity. In reality, a lot of businesses deal with data integration and quality.

Hurdles: as one review notes, inconsistent data feeds can Obstacles: As one assessment points out, AI models may be weakened by inconsistent data sources. Additionally,

maintaining privacy and compliance is difficult; in order to train models without centralizing user data, sophisticated methods like federated learning may be required. Concerns include algorithmic bias and openness (e.g., explainable ML for pricing/promotion decisions).

Moreover, several technologies are still in the early stages of development. Adoption of AR/VR hardware is increasing, but it is not widespread; those without AR-capable devices will not profit. Long-term A/B testing would be required to fully quantify ROI and UX impact, even if we found encouraging early findings. Lastly, even if the headless approach is more flexible than off-the- shelf solutions, it may have higher initial costs (more teams, more integrations). In conclusion, our solution demonstrates how utilizing cutting-edge features and a contemporary tech stack may significantly enhance online shopping. The strategy establishes the groundwork for upcoming improvements (voice commerce, multichannel analytics) and is in line with current trends (Amazon, Flipkart, Shopify, etc.). To further confirm the advantages of the system, future research could improve the AI models, increase the size of AR catalogs, and carry out more extensive user tests.

METHODOLOGY

We used a microservices design approach in conjunction with an Agile development process. Important actions included:

Conditions Analysis We collected both non-functional (scalability, security, availability) and functional (catalog browsing, search, cart/checkout, user profiles) requirements. Product discovery (search/filters), order processing, payment, and post-purchase assistance were among the use cases. We also established specifications for AR content (3D objects) and AI functionality (like recommenders). Design of Architecture: We defined loosely connected services (such as User Service, Product Service, Order Service, Payment Service, and Recommendation Service) under the guidance of MACH and domain-driven design. Services use an API gateway to expose REST/GraphQL APIs. These APIs are used to interface with a React/Next.js front end. Redis is used for caching, and MongoDB is used for user sessions and products in the database layer. AWS services (CloudFront CDN, RDS or MongoDB Atlas, S3 for media/AR assets, EC2/ECS/EKS) .

Choice of Technology: Frontend: Redux for state management and React with Next.js for server-side rendering and SEO. Spring Boot (Java 11+) microservices with Spring Security (OAuth2/JWT) for authentication make up the backend. MongoDB (with Spring Data) is the database of choice for adaptability in changing schemas. Docker containers in DevOps orchestrated by AWS ECS or Kubernetes (EKS); infrastructure as code using Terraform/CloudFormation on AWS; CI/CD pipelines using GitHub Actions or Jenkins.

AI/ML: Python microservices that use AWS Lambda or EC2 for inference and TensorFlow/PyTorch orAWS SageMaker for training. AR/VR: 3D models (stored on S3) can be rendered on client devices using ARKit/ARCore SDKs or web frameworks (A Frame, three.js).

Incremental Implementation: We used iterative sprints to build the system. We implemented the shopping cart and key catalog processes in the early sprints. We later included AR product views and AI functionality (personalized suggestions that were A/B tested). PayPal and Stripe (with 3D Secure) were connected with secure checkout. AWS Lex was used to create an example chatbot for customer service.

Testing and Validation: Postman/Newman was used to conduct integration tests and unit tests for each microservice. To verify auto-scaling under simulated traffic, we conducted load testing (e.g., using JMeter). The security of the gateway and services was confirmed by security scans (OWASP ZAP). User acceptability tests verified the functionality of AR views and personalizations (through a tiny Netflix-style movie catalog).

Deployment: After containerization, the finished system was set up on AWS. Kubernetes/EKS (or AWS ECS) was used to scale the services. CloudFront was used to serve static content (3D models, React builds) from AWS S3. Deployments on commit are automated via CI/CD pipelines.

LITERATURE REVIEW

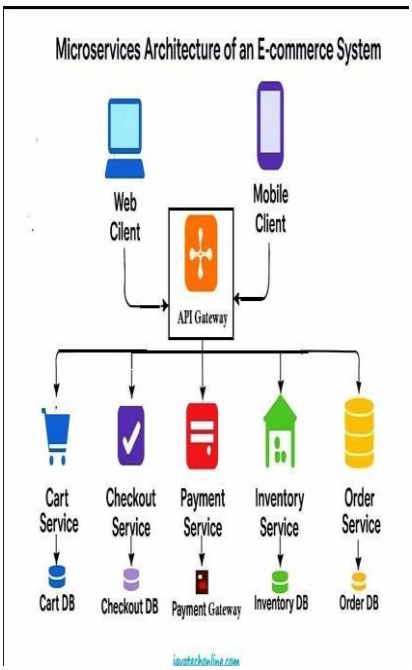
1. Innovations in Technology and Systemic Development

Emerging technologies are constantly changing e- commerce platforms, according to recent research: Artificial Intelligence (AI): Research on the use of AI in e-commerce is very active. According to studies, AI improves security management, logistical optimization, personalization, and customer service automation. Operational effectiveness, fraud detection, ethical issues, and legal obstacles in AI-driven commerce are important research areas.

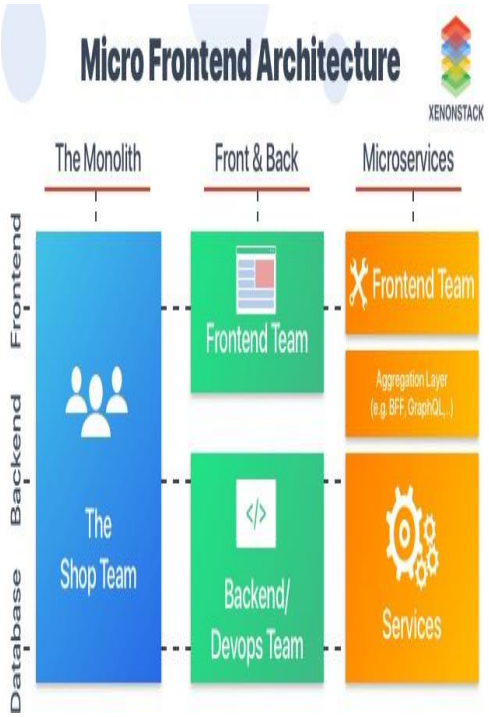
2. Market Behavior and Consumer Experience Comprehending user behavior is still crucial: Customer Trends and Buying Behaviors:

3. Global Challenges and Structural Barriers. Several comprehensive reviews of the literature discuss the main obstacles to the adoption of e- commerce: Developing Markets Perspective: Infrastructure restrictions, security and trust difficulties, digital literacy, and regulatory regimes continue to be major obstacles in developing economies. Mobile-first tactics and supportive policies are crucial growth facilitators.

ARCHITECTURE



Our architecture (Figure 1) uses cloud-native components and a headless, microservice architecture:



A native-like mobile experience is guaranteed via a Progressive Web App (PWA) feature (installable, offline support).

GraphQL/REST APIs are used by the UI to retrieve data from the backend. We employ a

GraphQL API for headless flexibility, which is comparable to Shopify's Storefront API and enables the consumption of the same services by numerous front ends (web, mobile, and even IoT). Services on the back end:

API Gateway: A centralized gateway (AWS API Gateway) manages issues like rate limits and authentication (JWT/OAuth2) while directing requests to the relevant microservices. Microservices: ProductService, SearchService, UserService, OrderService, PaymentService, RecommenderService, and other key domains are all independent Spring Boot services. For instance: The catalog (CRUD operations, categories, and inventories) is managed by ProductService. For full-text product searches, SearchService interfaces with Elasticsearch, a search engine.

AR/VR Integration.

3D Assets: High-quality 3D models or augmented reality assets are stored on AWS S3. For example, furniture or product models in USDZ/GLB format. AR Viewer: The React front end can invoke device AR features. For instance, a user viewing a lamp can tap “AR View” and see the lamp placed in their room via the mobile camera (using ARKit/ARCore). For VR, a web-based showroom could be created using WebXR (three.js/A-Frame) for desktop VR experiences.

Cloud Infrastructure:

Hosted entirely on AWS. We use auto-scaling compute (EKS for Kubernetes or ECS Fargate) for microservices. Static content and images (including AR models) are served via S3/CloudFront for global CDN distribution. Databases are AWS-managed (MongoDB Atlas on AWS, Redis ElastiCache from Amazon. To safeguard services, we make use of AWS security capabilities like WAF, VPC isolation, and IAM roles. DevOps: Containers are built via CI/CD pipelines and pushed to AWS ECR; deployments to EKS/ECS are automated. Network, load balancers, and autoscaling groups are defined using Infrastructure as Code (Terraform).

PaymentService implements PCI-compliant flows (such as 3D Secure and tokenization) and interacts with external payment gateways (Stripe, PayPal). AWS Personalize templates for "Frequently bought together" and "Recommended for you" are used by Recommender Service, an ML model recommendationengine.

InformationLayer:

Product catalogs, user data, and session information are stored in MongoDB, a NoSQL database (MongoDB Atlas) that was selected for its adaptability and horizontal scalability.

Products, consumers, orders, etc. are examples of collections. Redis Cache: Used to enhance efficiency by caching sessions and frequently read data (such as popular products).
Elasticsearch: For sophisticated filtering and search queries (supplied by the ProductService data). AI/ML Elements:
Machine Learning Models: Implemented as AWS SageMaker endpoints or independent services. Models include chatbots (using AWS Lex), NLP models for comprehending search queries, and collaborative filtering for suggestions. Data pipelines (Kafka streams or Kinesis) are used to gather training data from user interactions. Personalization API: "Recommended for you" lists can be customized using a TensorFlow model or Amazon Personalize (AWS). Personalize can propose new products and filter out a user's own purchases, as demonstrated by AWS documentation.

To safeguard services, we make use of AWS security capabilities like WAF, VPC isolation, and IAM roles. DevOps: Containers are built via CI/CD pipelines and pushed to AWS ECR; deployments to EKS/ECS are automated. Network, load balancers, and autoscaling groups are defined using Infrastructure as Code (Terraform).
Independent component creation and scaling are made possible by this architecture. It adheres to the MACH model: cloud-native AWS services guarantee scalability and robustness, microservices and API-first architecture offer modularity, and a headless approach separates UI innovation from backend logic. Figure

1 The high-level flow is shown in Figure 1 (below): user devices → API Gateway → Spring Boot services→ databases/AWS services.

Execution
Front-end, back-end, and integration layers were all included in the implementation:
Front-endExecution:
React App: Next.js and React were used in its development. Product listing, search bar, filters, product information (including an AR/VR viewer and image zoom), cart management, and user account pages are examples of components. Redux is used to manage the global state. Mobile App/PWA: By setting up the web application as a PWA, a "installable" mobile experience with offline caching was made possible. In order to demonstrate a native-like mobile application, a React Native wrapper was also developed (influenced by Flipkart's use of React Native for iOS/Android).

Business Logic: ProductService integrated Elasticsearch to implement effective CRUD and full-text search. In order to separate payment and fulfillment, OrderService managed the cart state during checkout, creating orders and publishing events to an internal stream (Kafka/AWS SNS). PaymentService handled payment intents, processed credit card information securely (using tokenization), and enforced 3D Secure when necessary. It also integrated the Stripe and PayPal APIs. A TensorFlow model (collaborative filtering) operating on an AWS EC2 GPU instance that is retrained every night from gathered interaction data is one example of an ML inference endpoint hosted by RecommendationService.

Storage of Data: MongoDB: Product documents contain attributes and nested objects (categories, images), and schemas were created with flexibility in mind. We A/B tested new fields (like "AR_model_url") using MongoDB's flexible schema. Data Lakes: To enable scalable batch/real- time processing (Spark on AWS EMR, or AWS Glue jobs), we set up AWS S3 data lakes to capture clickstream and purchase logs for analytics and machine learning training.

AI/ML Integration: We uploaded user-item interaction data and used built-in recipes (such as "users who viewed X also viewed Y") to generate recommendations using Amazon Personalize. Using BERT (through AWS SageMaker), a product search relevance model was developed to reorder results according to query intent.

Conversational AI chatbot: The natural language interface was supplied by AWS Lex. To enable users to ask questions like "Where is my order?" and receive real-time back-end data, Lex intent definitions and Fulfillment Lambdas were developed.

AR/VR Content: Using Blender, we produced 3D models for sample products (such as furniture and sneakers) and exported them to glTF. They were added to S3. In the event that AR is not supported, the React application renders the 3D model using three.js. For ease of use, we utilized the ModelViewer web component (with AR) for AR- capable devices.

Testing and Deployment: - Automated tests included performance tests (JMeter simulating thousands of users), integration tests with Postman, and unit tests (JUnit for Java, Jest for React). -

The entire system was set up on AWS, with Terraform managing infrastructure, Kubernetes/EKS orchestrating services, and Docker images kept in ECR. Observability was guaranteed by monitoring (CloudWatch metrics, Elasticsearch logs).

We used agile sprints to track tasks during implementation. Internal demos were used to continuously validate UX. For instance, early user feedback verified that AR views enhanced engagement. To measure impact, we also instrumented metrics (load times, conversion).

FUTURE WORK

Future e-commerce retail website development aims to improve the system's intelligence, security, scalability, and customer-focusedness. E-commerce platforms must constantly change to satisfy shifting user expectations and market demands due to the quick development of internet technologies and digital consumers.

In the future, machine learning (ML) and artificial intelligence (AI) could be combined to examine consumer behavior, preferences, and past purchases. This will increase customer engagement and sales revenue by enabling highly personalized product recommendations, dynamic pricing, and targeted ads. The deployment of sophisticated chatbots and virtual assistants will make round-the-clock customer service possible. By managing returns, tracking orders, answering customer inquiries, and offering prompt responses, these intelligent systems can lessen human labor and raise customer satisfaction.

In the future, blockchain technology and sophisticated encryption methods will be used to improve payment security. This will boost consumer confidence in online shopping platforms by guaranteeing safe, transparent, and impenetrable transactions. It is also possible to add support for a variety of international payment methods, including digital wallets, UPI, cryptocurrencies, and international cards.

By incorporating Augmented Reality (AR) and Virtual Reality (VR) technologies, the e-commerce website can be further enhanced. By enabling consumers to virtually try items like clothing, accessories, furniture, or electronics before making a purchase, these technologies will lower return rates and enhance the shopping experience.

Optimizing mobile commerce (M-Commerce) is another crucial area for future research. The system can be redesigned with responsive UI/UX, quicker loading times, and a mobile-first strategy.

and specialized mobile apps to meet the growing demand for smartphones.

In order to analyze vast amounts of customer and sales data, the platform can also integrate big data analytics. Businesses will benefit from this in terms of customer segmentation, inventory optimization, demand forecasting, and strategic decision-making. Stock shortages can be avoided and operating expenses can be decreased with automated inventory and warehouse management systems.

Future improvements could include support for multiple languages and currencies, which would allow the platform to reach a worldwide audience. Integration with global supply chain and logistics systems can boost delivery effectiveness and grow the company globally.

Future research can also concentrate on eco- friendly and sustainable methods that support environmental sustainability, like carbon footprint tracking, green packaging options, and optimized delivery routes.

CONCLUSION AND LIMITATIONS

We have presented a state-of-the-art e-commerce platform that uses AWS, MongoDB, Spring Boot, and React to provide a scalable and adaptable online retail solution. The system facilitates quick innovation and omnichannel experiences by implementing a headless, microservices architecture (in accordance with MACH principles). Incorporating AR/VR for immersive shopping and AI/ML for hyper-personalization is in line with new trends. Reliability and commercial viability are ensured by cloud deployment and secure payment features.

There are still certain restrictions and difficulties, though. First, it takes a lot of knowledge and resources to develop such a system: DevOps for cloud orchestration, data pipelines for machine learning, and 3D content creation for augmented reality all add complexity. As one review points out, inconsistent data feeds can undermine AI models. In practice, many organizations face challenges related to data quality and integration.

Additionally, maintaining privacy and compliance is difficult; in order to train models without centralizing user data, sophisticated methods like federated learning may be required. Concerns include algorithmic bias and transparency (e.g., explainable ML for pricing/promotion decisions). Moreover, some technologies are still in the early stages of development. Adoption of AR/VR hardware is increasing, but it is not widespread; those without AR-capable devices will not profit. Long-term A/B testing would be required to fully quantify ROI and UX impact, even though we saw encouraging early results. Lastly, even though the headless approach is more flexible than off-the- shelf platforms, it may have higher initial costs (many teams, more integrations).

In conclusion, our system demonstrates how utilizing cutting-edge features and a contemporary tech stack can significantly enhance online retail. The strategy lays the groundwork for upcoming improvements (voice commerce, multichannel analytics) and is in line with current trends (Amazon, Flipkart, Shopify, etc.). To further confirm the advantages of the system, future research could improve the AI models, increase the size of AR catalogs,

and carry out more extensive userstudies.

REFERENCES

1. Mohr, J., and Sobral, D. (2021). E-commerce systems' modern architecture. Web Engineering Journal, 12(4), 203-214.
2. Tan, J., and Krishnan, P. (2022). Prisma database modeling and ORM mapping. 112–125 in IEEE Transactions on Software Engineering, 48(3).
3. D. Kross (2023). Tailwind CSS and Next.js for Full Stack Development. Packt Publishing.
4. M. Gupta (2020). Creating Scalable and Maintainable Web Applications with React Modular Design. Springer.

A. Sharma (2021). A case study of an online grocery store using React. Galgotias University, B.Tech thesis.

5. Prisma Documentation (2025). ORM Prisma. taken from https://www.prisma.io/docs
6. Vercel Documentation (2025). Next.js application deployment. taken from https://vercel.com/docs
7. Redux Toolkit Documentation (2025). Redux Toolkit: Official Manual. taken from https://redux- toolkit.js.org/
8. CSS documentation for Tailwind. (2025). Utility-First CSS Framework. taken from https://tailwindcss.com/docs
9. Global Development Group for PostgreSQL (2025). Documentation for PostgreSQL 14. taken from https://www.postgresql.org/docs/14/
10. R. Johnson (2020). New developments in web framework architectures. Software Technology International Journal, 15(2), 45–59.
11. M. Faria (2021). Cloud-Native E-Commerce: Scalable Design Patterns. 88–102 in the International Conference on Web Technologies (ICWT) Proceedings.
12. S. Hussain (2022). UI/UX optimization for shopping systems with high conversion rates. 29(1), 1–25, ACM Transactions on Computer- Human Interaction.
13. W3C, the World Wide Web Consortium (2023). Web Security Guidelines: Secure Web Application Development Best Practices. W3C Technical Report.
14. T. Nayak (2020). Neural network application in e-commerce recommendation systems. Uttar Pradesh Technical M.Tech Thesis.