



Smart AI-Powered E-Commerce Platform with Personalized Recommendation and Sales Analytics

Wadekar Gargi Prashant¹, Bole Ishwari Santosh², Patil Harshvardhan Sunil³,

Mr. S. S. Ingole⁴

UG Student, Department of Computer Science and Engineering, SKN Sinhgad College of Engineering, Pandharpur, India¹⁻³

Project Guide, Department of Computer Science and Engineering, SKN Sinhgad College of Engineering, Pandharpur, India⁴

Abstract: Online retail has become a primary shopping channel, and customers increasingly expect experiences that are tailored to them [1]. Many small and mid-sized merchants, however, still operate storefronts that show every visitor the same catalog and give owners little insight into what is selling, what is running out, and why. This paper presents a Smart AI-Powered E-Commerce Platform that combines personalized product recommendation, a database-grounded multilingual shopping assistant, and an analytics-driven administration console in a single full-stack web application. The recommendation module ranks products with an explainable score built from category affinity, popularity, customer rating, and price suitability, and derives “frequently bought together” suggestions from real co-purchase history. The shopping assistant interprets free-text queries in English, Hindi, and Marathi, retrieves products only from the store database, and phrases its replies through a language-model layer with an automatic rule-based fallback, so it never invents items or prices. For the merchant, the platform provides sales-trend, category, and regional analytics together with an inventory module that estimates sales velocity and proposes restock quantities. The system is implemented with React, Node.js/Express, and MongoDB, and includes JWT-based sessions, e-mail one-time-password two-factor login, server-side price computation, Razorpay online payment, and cash on delivery. A working prototype comprising 89 REST endpoints and 12 data collections demonstrates the design across customer and administrator interfaces. The platform is expected to improve product discovery, reduce stock-outs, and support data-driven decisions for local retailers; future work covers collaborative filtering, payment-gateway webhooks, and large-scale usability and load evaluation.

Keywords: E-Commerce, Personalized Recommendation, Sales Analytics, AI Shopping Assistant, Inventory Intelligence, MERN Stack

I. INTRODUCTION

Online retail in India has moved from a convenience to a mainstream shopping channel. A joint study by Bain & Company and Flipkart reports that shoppers from tier-2 and smaller towns make up nearly half of all online shoppers and place roughly three out of every five orders on leading platforms, and that a typical shopper views more than twenty product pages before buying [2]. For a merchant, growth is therefore driven increasingly by customers who are new to online shopping and who compare many alternatives before deciding. A store that serves them must help them reach the right product quickly and, ideally, in a language they are comfortable with.

Personalization has become the standard answer to this discovery problem. McKinsey reports that 71 percent of consumers expect companies to deliver personalized interactions and that 76 percent become frustrated when this does not happen [1]. Recommender systems, which suggest items by exploiting the behavior and opinions of a community of users, have consequently become a core component of e-commerce sites [3][4]. The large marketplaces that popularized these techniques, however, rely on extensive interaction logs and dedicated data-science teams that small and mid-sized retailers usually do not have.

Merchants also face a second, less visible problem on the administration side. Order, stock, and customer records often sit in separate spreadsheets or in generic dashboards that report what has already happened but do not indicate which products are about to run out, which categories are growing, or which regions generate demand. Missed restocking causes lost sales, while excess stock ties up working capital.



Recent progress in language models and web frameworks makes it practical to address both problems in one system. A conversational interface lets customers describe what they want in natural language, including price limits and preferences, instead of navigating nested menus, while an analytics layer built on the store's own transaction data can turn raw orders into restocking guidance.

This manuscript describes "Smart AI-Powered E-Commerce Platform with Personalized Recommendation and Sales Analytics," a full-stack web application for a stainless-steel utensil retailer, in which customers can:

- Receive recommendations based on their browsing, wishlist, and purchase behavior
- Ask a multilingual shopping assistant to find products by category, price, or capacity
- Complete a secure checkout through online payment or cash on delivery
- Track orders and receive notifications and price alerts

Administrators, in turn, obtain sales, category, and regional analytics, inventory status with restock suggestions, and tools to manage the catalog, orders, coupons, and customers.

The principal motivations for this work are the rapid spread of online retail beyond metropolitan areas, rising customer expectations of personalization, the need for explainable and affordable AI components that small retailers can operate without large data-science infrastructure, and the availability of mature open-source web technologies. The platform is designed to be modular, multilingual, and straightforward to extend with more advanced models.

II. RELATED WORK

The design of the proposed platform draws on four bodies of literature: (i) recommender systems for e-commerce, (ii) conversational shopping assistants and grounded language generation, (iii) business analytics and inventory decision support, and (iv) the security, privacy, and usability requirements of online stores.

A. Recommender Systems in E-Commerce

Resnick and Varian described recommender systems as tools that help people make choices by drawing on the opinions of a community [3], and Schafer et al. documented how e-commerce sites use them to turn browsers into buyers, encourage cross-selling, and build loyalty [4]. Adomavicius and Tuzhilin later organized the field into content-based, collaborative, and hybrid approaches and identified recurring limitations such as cold start and data sparsity [5]; a broader treatment appears in the Recommender Systems Handbook [6]. Item-based collaborative filtering improved scalability by comparing items rather than users [7], and the item-to-item method deployed at Amazon.com showed that the idea can serve millions of customers and products [8]. Matrix factorization became a dominant model-based technique because it learns compact latent factors from rating data [9], neural architectures have since replaced the inner product with learned interaction functions [10], and the deep-learning literature is surveyed in [11]. Hybrid designs, which combine several signals to compensate for the weaknesses of each, are an established way to mitigate cold start and sparsity [12]. Finally, studies of explanation interfaces indicate that showing users why an item was recommended improves transparency and acceptance [13][14]. These findings motivate the proposed engine, which uses a transparent hybrid score instead of an opaque trained model, because a young store lacks the interaction volume needed to train latent-factor models reliably.

B. Conversational Shopping Assistants and Grounded Generation

Adamopoulou and Moussiades reviewed chatbot technologies and showed that they range from hand-written rule-based systems to data-driven generative models, each with different trade-offs in coverage, controllability, and cost [15]. Classical language-processing pipelines extract an intent and its parameters, such as price limits or product attributes, from the utterance before any response is produced [16]. Large language models tolerate spelling variation and free-form phrasing far better [17], but they can generate fluent statements that no source supports, a failure known as hallucination [18]. Retrieval-augmented generation addresses this by conditioning the model on evidence fetched from an external store [19]. The assistant proposed here follows the same principle in a deliberately conservative form: product facts always come from a database query, and the language model is used only to interpret the question and to phrase the answer.

C. Sales Analytics and Inventory Intelligence

Chen et al. traced the evolution of business intelligence and analytics and highlighted its role in turning transactional data into managerial decisions [20]. In inventory management, classical replenishment theory sets reorder quantities from the demand expected during the replenishment lead time plus a desired cover period [21]. Small retailers seldom apply such methods because they require demand histories and tooling. The proposed inventory module therefore adopts the simplest transparent form of this logic: it estimates average daily demand from recent paid orders and reports insufficient data, rather than fabricating a forecast, when the history is too short.

**D. Security, Privacy, Regulation, and Usability**

Online stores handle credentials, addresses, and payments, and therefore face the risk categories catalogued by OWASP, which include broken access control, injection, and identification and authentication failures [22]. Stateless session tokens such as JSON Web Tokens are standardized in RFC 7519 [23], while adaptive password hashing such as bcrypt is a widely used defense against offline guessing [24]. Payment gateways such as Razorpay require the merchant's server to verify a signed response before an order is treated as paid [25], and MongoDB supports multi-document transactions for operations that must change several records atomically [26]. In India, the Digital Personal Data Protection Act, 2023 establishes obligations for processing digital personal data that center on consent and purpose limitation [27]. Finally, the System Usability Scale provides a quick, standardized instrument for evaluating interface usability [28], and it is the planned metric for the platform's future user evaluation.

TABLE I
MAPPING OF LITERATURE TO THE PROPOSED SYSTEM

Reference	Focus	Contribution to Proposed System
[1][2]	Consumer expectations of personalization; behavior of Indian online shoppers	Establishes the need for personalized discovery in the proposed storefront
[3][4][5][6]	Foundations and taxonomy of recommender systems	Frames the hybrid recommendation engine
[7][8]	Item-based and item-to-item collaborative filtering	Basis for the "frequently bought together" co-purchase module
[9][10][11]	Latent-factor and neural recommenders	Defines the future migration path beyond heuristic scoring
[12]	Hybrid recommendation	Supports combining affinity, popularity, rating, and price signals
[13][14]	Explanations in recommender systems	Motivates the transparent, component-wise score
[15][16]	Chatbot technologies and language-understanding pipeline	Guides the intent-extraction and retrieval design of the assistant
[17][18][19]	Grounded generation, hallucination, and large language models	Justifies database-grounded replies with an optional language-model layer
[20][21]	Business analytics and replenishment theory	Informs the sales dashboards and restock logic
[22][23][24]	Web-application security, session tokens, password hashing	Defines authentication and access-control mechanisms
[25][26]	Payment verification and database transactions	Defines the checkout and stock-consistency pipeline
[27][28]	Data protection and usability evaluation	Sets privacy requirements and the planned usability method

III. PROBLEM STATEMENT

Small and mid-sized retailers are increasingly expected to offer the same quality of online experience as large marketplaces. A conventional storefront lists products in a fixed order, supports only keyword search, and presents the same page to every visitor. Customers who are unsure about brand, size, or set composition must examine many pages, and those who prefer Hindi or Marathi are often left without assistance.

On the merchant side, order, stock, and customer information is seldom connected. Owners learn about stock-outs only after customers complain, cannot easily compare category or regional performance, and cannot support restocking decisions with data. The tools that do exist are either expensive enterprise suites or generic chatbots that may invent products and prices, which erodes customer trust.



Problem Statement: “Existing small-business e-commerce solutions present identical catalogs to all customers, offer limited multilingual guidance, rely on unverified conversational responses, and provide little actionable insight into sales and inventory. There is therefore a need for an integrated, secure, and affordable platform that personalizes product discovery, answers customer questions strictly from store data in the customer’s language, completes orders with server-verified pricing and stock control, and equips the administrator with analytics and restocking intelligence.”

Key Challenges Resolved by the Proposed Architecture:

- Generic Discovery: Replaces identical product listings with ranked, behavior-aware recommendations.
- Language Barriers: Supports English, Hindi, and Marathi across the interface, category names, and assistant replies.
- Unreliable AI Answers: Prevents invented products or prices by grounding every reply in database results.
- Limited Merchant Insight: Converts raw orders into revenue, category, product, and regional analytics.
- Stock-outs and Overstocking: Estimates sales velocity and proposes prioritized restock quantities.
- Order Integrity: Computes prices on the server and updates stock atomically to avoid mismatched totals and overselling.
- Account Security: Combines hashed passwords, e-mail one-time-password verification, role-based access, and rate limiting.
- Fragmented Tooling: Unifies the storefront, assistant, analytics, and administration in one application.

IV. PROPOSED SYSTEM AND METHODOLOGY

The proposed platform replaces a static online catalog with an intelligent, data-driven store. Its functionality is organized into four cooperating domains, described below, which share a single MongoDB database as the source of truth.

A. Customer Storefront and Transaction Layer

The storefront is a React single-page application that offers category browsing, full-text search with filters for price, material, size, rating, and availability, product pages with reviews and similar items, a wishlist, a cart with coupon support, and a multi-step checkout. Guests may browse freely; protected actions such as adding to the cart open a login prompt and resume the interrupted action after sign-in. Registration requires verification of the e-mail address through a one-time password, and login can require a second one-time-password step. At checkout, the server recomputes every price from the live catalog and never trusts totals sent by the browser. Online payments are created as Razorpay orders and confirmed by verifying the gateway’s signature on the server [25]; cash on delivery follows the same validation pipeline with a pending payment status. Order creation runs inside a database transaction that conditionally decrements stock, stores immutable item snapshots, records coupon usage, and clears the cart [26]. Orders progress through placed, confirmed, processing, shipped, out-for-delivery, and delivered states, and customers can cancel before shipment, which restores stock.

B. Personalized Recommendation Engine

For a signed-in customer, the engine builds a profile from three signals: the twenty most recently viewed products, the wishlist, and past paid orders. Candidates are active, in-stock products that the customer has not already bought, limited to a pool of 200. Each candidate p receives the score $S(u, p)$ given in Eq. (1):

$$S(u, p) = 3 \cdot C(u, p) + 2 \cdot (n_p / n_{\max}) + 2 \cdot (r_p / 5) + P(u, p) \quad (1)$$

where $C(u, p)$ equals 1 when the category of p appears among the customer’s recently viewed or wishlisted items and 0 otherwise; n_p and $n_{\max}$ are the sales count of p and the largest sales count in the candidate pool; r_p is the average customer rating on a 0–5 scale; and $P(u, p)$ equals 1 when the selling price lies within ± 40 percent of the customer’s mean past purchase price. The score therefore ranges from 0 to 8, as summarized in Table II. Customers with no history receive best-selling and highly rated products as a cold-start fallback [5]. Because every term is visible, the reason behind a recommendation can be explained and tuned [13].



TABLE II
COMPONENTS OF THE RECOMMENDATION SCORE

Component	Signal Source	Range	Purpose
Category affinity C	Last 20 viewed products and wishlist	0 or 3	Personal relevance
Popularity	Sales count normalized by the pool maximum	0 – 2	Social proof
Rating	Average review rating divided by 5	0 – 2	Product quality
Price suitability P	Within $\pm 40\%$ of mean past paid-order price	0 or 1	Budget fit
Total	Sum of the four components	0 – 8	Ranking of candidates

A second module produces “frequently bought together” suggestions. For a given product, the system scans up to 200 paid orders that contain it, counts how often every other product appears alongside it, and returns the four most frequent companions, in the spirit of item-to-item co-purchase methods [7][8]. When no co-purchase history exists, it falls back to a list curated by the administrator.

C. AI Shopping Assistant

The assistant, available as a chat widget on every page, processes each message in three stages. In the understanding stage, the message is converted into a structured intent containing a category, up to five keywords, minimum and maximum price, a capacity, a sort preference, and an intent type. When a Gemini API key is configured, a language model performs this extraction under a strict JSON schema, and its output is validated so that a category outside the store’s list or a malformed field can never reach a query [17]. Otherwise, or if the request fails or exceeds an eight-second limit, a deterministic rule-based extractor handles stop-word filtering, price phrases such as “under 500” or “between 200 and 500,” capacity expressions, sort phrases, and singular/plural category matching [16]. In the retrieval stage, MongoDB is queried through a five-step cascade: full-text search with filters; the same search without a strict capacity filter; category and price filters alone; a per-keyword pattern scan; and, if nothing matches, an empty result. At most six products are returned, and the store never substitutes unrelated items. In the generation stage, the reply is phrased in English, Hindi, or Marathi, either by the language model, which is instructed to mention only the listed products and prices [18][19], or by multilingual templates. Messages are limited to 500 characters, requests are rate-limited to 15 per minute, and the history of authenticated users is stored for continuity.

D. Administrator Analytics and Inventory Intelligence

The administration console exposes the store’s data through analytics endpoints that accept preset ranges (today, 7 days, 30 days, 3 months, one year, all time) or a custom date range. The dashboard reports revenue, orders, customers, active products, average order value, units sold, and low-stock counts; charts show revenue over time, revenue by category, order-status distribution, top-selling products, and orders and revenue by city and state, derived from shipping addresses already collected at checkout [20].

The inventory module classifies each product as in stock, low stock, or out of stock using a per-product minimum threshold, and it estimates sales velocity from paid orders in the last 30 days. When a product has at least two orders in that window, the estimated days of cover and the suggested restock quantity are computed as in Eq. (2):

$$Q_p = \max(0, \text{ceil}(d_p \cdot (L + T)) - s_p), \quad E_p = s_p / d_p \quad (2)$$

where d_p is the average number of units sold per day, s_p the current stock, $L = 7$ days the assumed supplier lead time, and $T = 14$ days the target cover period [21]. Products are given high priority when they are out of stock or have at most L days of cover, and medium priority when they are low in stock or have at most T days of cover; products selling one unit per day or more are tagged fast-moving. For products with fewer than two recent orders, the system reports that history is insufficient instead of producing a number. In addition, the system records each selling-price change, e-mails customers whose wishlists contain the product, and alerts the administrator by e-mail when stock first falls to ten units or fewer.

E. Core Technical Methodologies

- Server-Authoritative Commerce: Prices, shipping, coupon discounts, and totals are recomputed on the server for every checkout, and stock is decremented atomically [26].



- Explainable Personalization: A weighted additive score replaces opaque models, so the contribution of each signal can be inspected [13].
- Grounded Conversational AI: Database retrieval decides which products exist; the language model only interprets and phrases [19].
- Multilingual Interface: The i18next framework serves English, Hindi, and Marathi, and category names carry stored translations.
- Layered Security: bcrypt password hashing with cost factor 12 [24], JWT sessions delivered as HTTP-only cookies or bearer tokens [23], hashed one-time passwords with expiry and attempt limits, role-based access control, HTTP-header hardening, NoSQL-injection sanitization, and endpoint-specific rate limiting [22].
- Notification Services: SMTP e-mail and in-app notifications for orders, price changes, stock alerts, and administrator recommendations.
- Modular Scalability: The frontend, API, domain services, and database are separated so that components such as the recommender can be replaced independently.

F. Operational Workflow

- Customer Operations: The user registers, verifies the e-mail with a one-time password, logs in, browses or asks the assistant, adds products to the cart, applies a coupon, pays online or chooses cash on delivery, and later tracks, cancels, or reviews the order.
- Administrator Operations: The administrator signs in, monitors the dashboards, manages products, categories, coupons, and orders, reviews restock recommendations, and sends personalized recommendations to customers.
- Backend Sequence: Each request passes through security middleware and authentication, reaches a controller, invokes domain services, reads or writes MongoDB (transactionally for orders), and finally triggers notification e-mails where applicable.

V. SYSTEM DESIGN

The design is documented with Unified Modeling Language artifacts and data-flow diagrams so that both developers and store owners can follow the logic. These comprise a context-level data-flow diagram, a decomposed process-level diagram, and use-case, sequence, and entity-relationship views.

At context level, the platform is a single process that interacts with five external entities: the customer, the administrator, the payment gateway, the e-mail service, and the optional language-model service. Decomposition reveals eight processes: Authentication and Verification, Catalog and Search, Cart and Coupon Management, Order and Payment Processing, Recommendation, AI Assistant, Analytics and Inventory, and Notification Management. The use-case model has two actors: customers use the browsing, assistant, cart, checkout, and review use cases, while administrators use the catalog, order, coupon, customer, analytics, and inventory use cases.

The checkout sequence diagram traces the chronological exchange among the browser, the API, the cart service, Razorpay, and the database: the browser requests a payment order; the server recomputes the total and creates the gateway order; the customer pays in the gateway window; the browser returns the payment identifiers; the server verifies the signature, re-validates stock, and executes the order transaction; and a confirmation e-mail is dispatched. The entity-relationship view centers on the twelve collections summarized in Table III. Orders keep price snapshots of purchased items so that later catalog edits never alter historical records.

TABLE III
PRINCIPAL DATA COLLECTIONS OF THE PLATFORM

Collection	Principal Fields and Purpose
Users	Name, e-mail, hashed password, role, two-factor flag, hashed one-time passwords, saved addresses, notification preferences, preferred language
Products	Name, SKU, category, MRP, selling price, stock, minimum-stock threshold, images, rating, sales and view counters
Categories	Name, slug, display order, Hindi and Marathi translations
Carts	User, line items (product and quantity), applied coupon



Collection	Principal Fields and Purpose
Orders	Item snapshots, shipping address, totals, payment method and status, order status, tracking history
Coupons	Code, discount type and value, cap, minimum order value, validity window, global and per-user usage limits, usage log
Reviews	Product, user, 1–5 rating, comment, verified-purchase flag (one review per user per product)
Wishlists	User and list of saved products with timestamps
Notifications	User, type (order, price drop, promotion, recommendation), message, read flag
Conversations	User, assistant message history, products referenced in replies
RecentlyViewed	User, product, last-viewed timestamp (input to personalization)
ProductPriceHistory	Product, old and new price, change type and percentage, alert status

VI. ARCHITECTURE

The platform follows a layered client–server architecture, shown in Fig. 1:

- **Presentation Layer:** A React 18 single-page application built with Vite provides 15 storefront screens, 11 administration screens, the chat widget, and shared providers for authentication, cart, wishlist, and notifications, with translations in three languages.
- **API and Security Middleware:** An Express.js REST API of 89 endpoints applies HTTP-header hardening, cross-origin controls, input sanitization, rate limiting, JWT verification, role checks, and centralized error handling before a request reaches business logic.
- **Application Services:** Domain services implement authentication with one-time passwords, catalog and search, cart and coupons, order and payment processing, the recommendation engine, the assistant pipeline, analytics and inventory intelligence, and notifications with price alerts.
- **Data and External Services:** MongoDB, accessed through Mongoose, stores twelve collections; Razorpay handles online payments, an SMTP service delivers e-mail, and the Gemini API optionally supports the assistant.

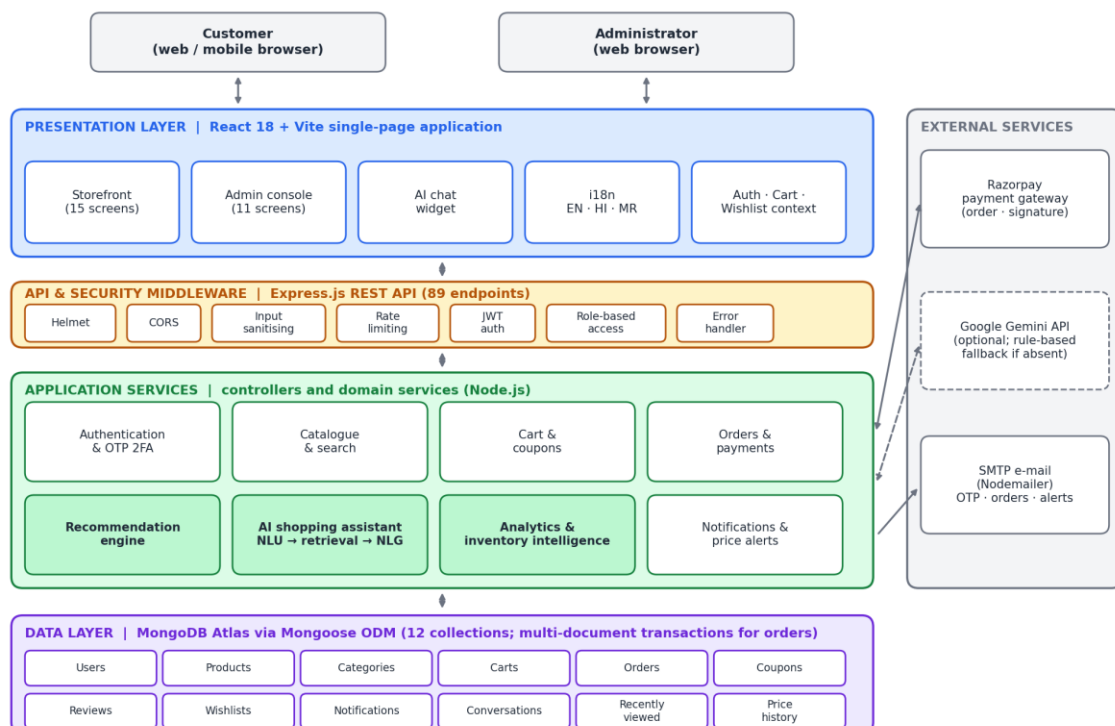


Fig. 1 System Architecture of the Smart AI-Powered E-Commerce Platform



VII. RESULTS

The architecture was implemented as a working prototype for a stainless-steel utensil retailer branded Mauli Steels, seeded with a catalog of 29 products in nine categories. Table IV summarizes the implemented scope, and the figures that follow present screenshots of the running system for the customer and administrator roles.

TABLE IV
SUMMARY OF THE IMPLEMENTED PROTOTYPE

Parameter	Value
REST API endpoints	89 (customer, administrator, assistant, and recommendation routes)
Database collections	12 (MongoDB through Mongoose)
User-interface screens	26 (15 storefront and 11 administration)
Interface languages	3 (English, Hindi, Marathi)
Seeded catalog	29 products in 9 categories
Payment options	Razorpay online payment and cash on delivery
Authentication	Password with e-mail one-time-password verification; JWT sessions; role-based access (customer, administrator)
Recommendation components	Explainable 0–8 score; frequently-bought-together co-purchase module
Administrator intelligence	Revenue, category, status, product, and regional analytics; restock recommendations

A. Customer Storefront

Fig. 2 shows the storefront home page for a signed-in customer. The header provides product search, the language selector (English, Hindi, Marathi), the account menu, the wishlist, and the cart with an item counter. Below it, a navigation bar links to the category menu and to the featured, best-seller, and new-arrival collections, followed by a rotating banner carousel, a value-proposition panel (secure payments, free shipping, easy returns, cash on delivery), and the floating button that opens the shopping assistant. For customers with browsing history, the recently viewed and “recommended for you” sections generated by the scoring engine are rendered further down the same page.

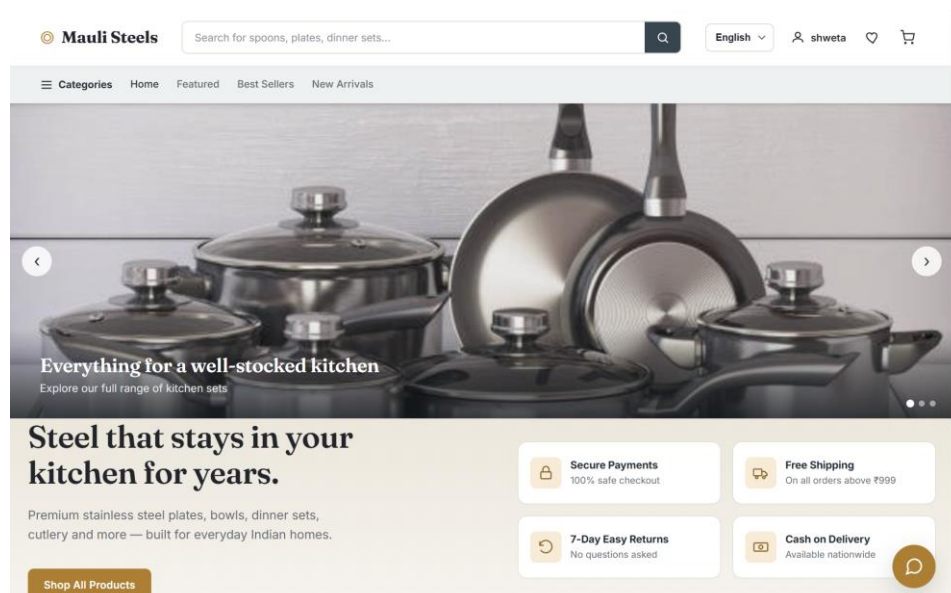


Fig. 2 Storefront Home Page for a Signed-In Customer



Fig. 3 shows a product page. It presents the product name and category breadcrumb, the selling price against the MRP with a discount badge (₹999 against ₹1,299, 23 percent off), a short description and feature list, material, size, and weight attributes, the live stock count, a quantity selector, the Add to Cart and Buy Now actions, a wishlist toggle, and delivery information. Customer reviews and the “frequently bought together” and “similar products” panels follow below the fold, so that recommendations appear at the point of decision.

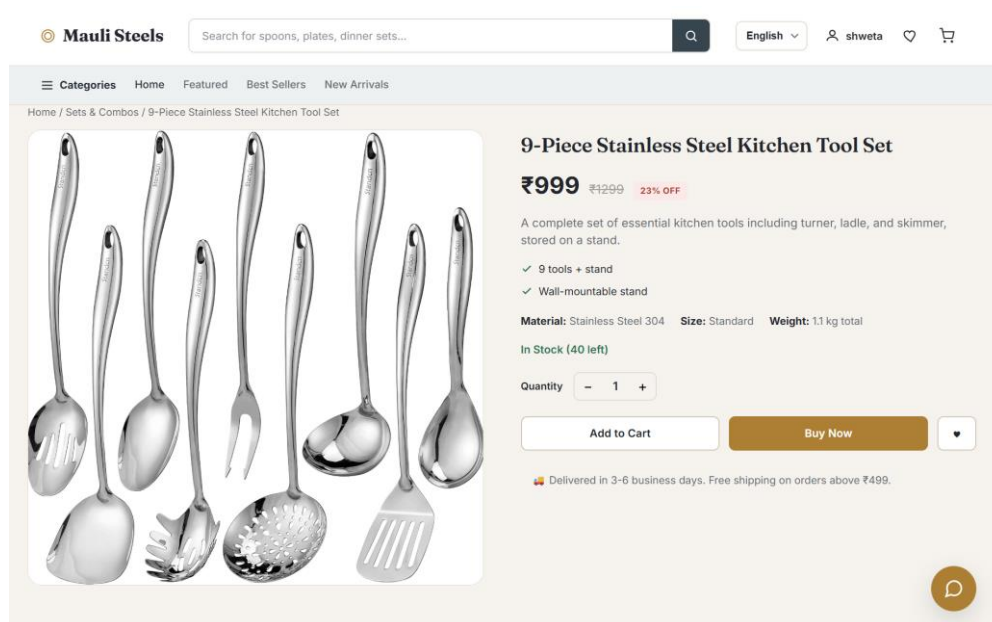


Fig. 3 Product Detail Page with Pricing, Stock Status, and Purchase Actions

B. AI Shopping Assistant

Fig. 4 presents the chat widget handling the natural-language request “i want gift sets under 1500.” The assistant resolved the category (Gift Sets) and the price ceiling, queried the database, and answered with a short message together with a product card showing the item’s image, name, and price (Steel Tea Set Gift Box, 13-piece, ₹1,149, 23 percent off). The card above it belongs to an earlier query in the same conversation. Every name and price in the reply comes from the product collection, and the same flow accepts requests in Hindi or Marathi.

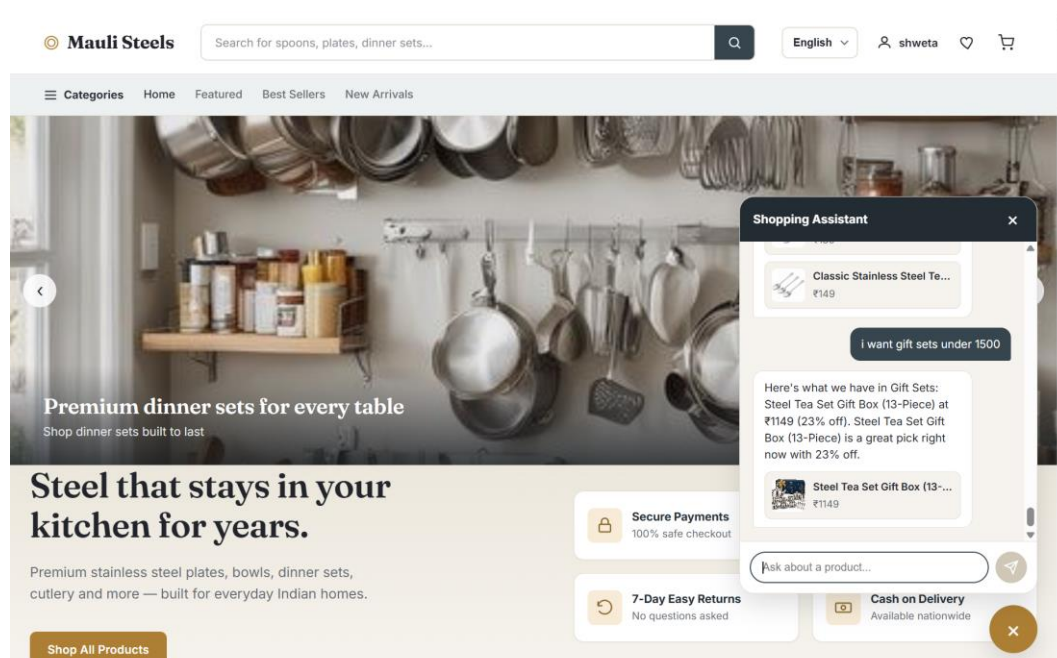


Fig. 4 AI Shopping Assistant Answering a Natural-Language Query with Database-Grounded Product Cards



C. Checkout and Payment

Checkout is a three-step flow (customer information, delivery address, review and pay). Fig. 5 shows the delivery-address step, in which the customer selects a saved address or adds a new one, next to the persistent payment summary listing the subtotal, shipping charge, and total. The customer chooses between Razorpay online payment (card, UPI, net banking, or wallet) and cash on delivery, and a notice confirms that online payments are verified on the server before the order is confirmed. Personal details in the saved address have been masked in the figure.

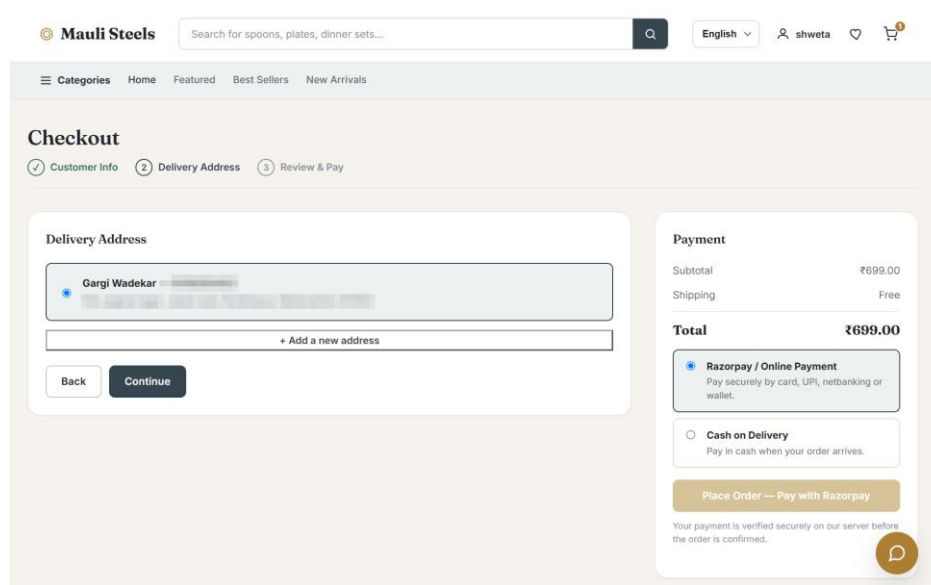


Fig. 5 Checkout: Delivery Address Selection and Payment Options

D. Administrator Dashboard

Fig. 6 shows the administrator's landing page. A sidebar gives access to the dashboard, sales analytics, inventory, products, categories, orders, coupons, customers, administrator accounts, and the profile page. Summary cards report total revenue, total orders, total customers, active products, and low-stock items; for the demonstration data they read ₹7,073, 7 orders, 5 customers, 29 active products, and no low-stock items. Two shortcuts lead to the Sales and Customer Analytics page, and to the Inventory and Restock Intelligence page. A recent-orders table lists each order number, customer, status, total, and date, and a low-stock alert panel confirms when all products are adequately stocked.

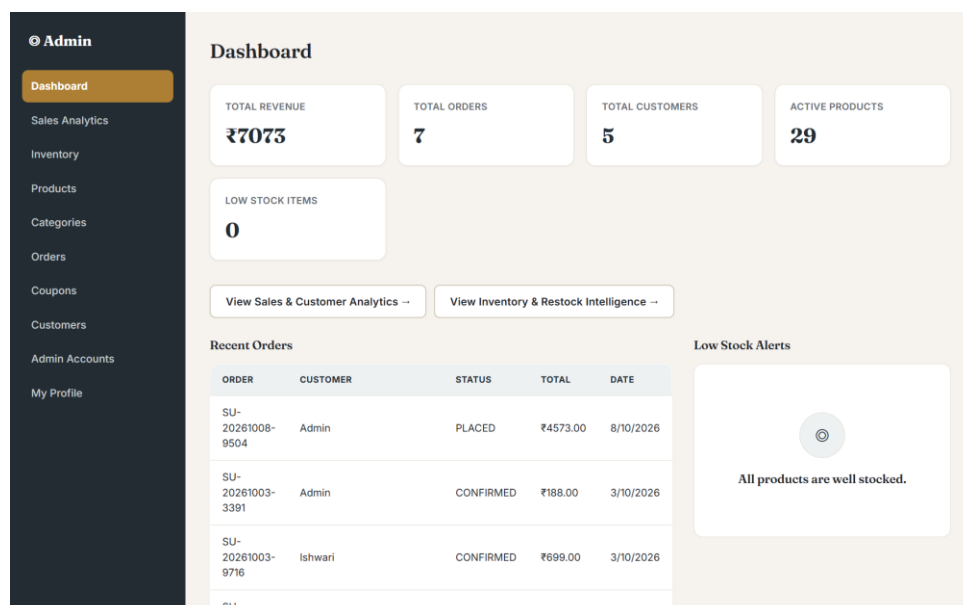


Fig. 6 Administrator Dashboard with Key Performance Indicators and Recent Orders



E. Inventory and Stock Intelligence

Fig. 7 shows the inventory console, whose subtitle states that sales velocity is computed from the last 30 days of paid orders. Summary cards count total products, products in stock, low on stock, out of stock, needing restock, and fast- or slow-moving. The product stock table lists the product, SKU, category, stock, minimum threshold, status, units sold, last sale date, average weekly sales, and movement class. Products without enough recent order history are labeled “Not enough data” instead of receiving an invented forecast, which illustrates the cautious design described in Section IV-D.

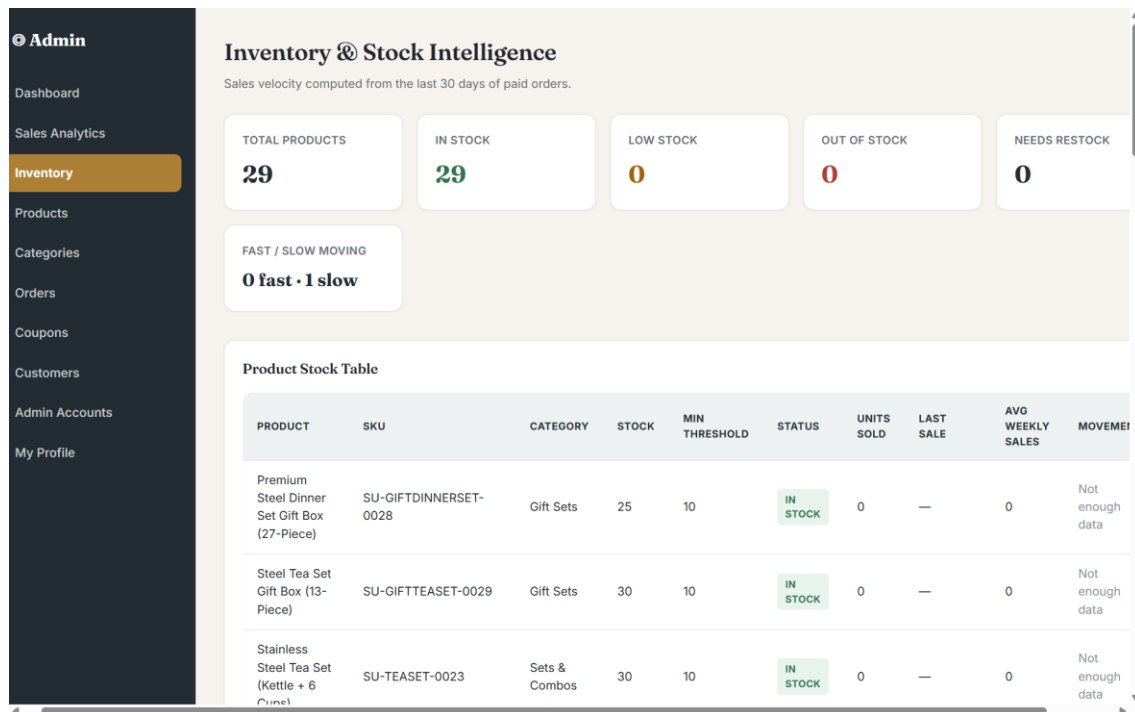


Fig. 7 Inventory and Stock Intelligence Console

VIII. FEASIBILITY ANALYSIS AND DISCUSSION

A. Technical Feasibility

The platform uses widely adopted open-source technologies: React 18 and Vite for the frontend, Node.js and Express.js for the API, MongoDB with Mongoose for persistence, i18next for localization, Recharts for visualization, Nodemailer for e-mail, and the Razorpay SDK for payments. The optional Gemini integration is isolated behind a service interface with a rule-based fallback, so the system remains fully functional without any external AI service. These technologies run on commodity hardware and on standard cloud hosting.

B. Economic Feasibility

Because every core component is open source and the personalization and analytics functions run on the store's own database, the platform avoids licensing costs and does not require a data-science team. Managed database tiers and the optional language-model service can be adopted incrementally, so a small retailer can begin with minimal infrastructure expenditure and let costs grow with traffic.

C. Operational Feasibility

Customers interact through familiar e-commerce patterns, with an assistant and a trilingual interface that reduce the effort of finding products. Administrators use dashboards and plain-language restock recommendations instead of raw tables, which is intended to reduce the need for spreadsheet-based analysis.

D. Expected Outcomes

The platform is intended to improve product discovery, shorten the path from search to purchase, reduce stock-outs through earlier restocking signals, and give owners evidence for pricing and catalog decisions. These benefits are expected rather than measured; they depend on adoption and on the accumulation of enough interaction data. Wider acceptance also depends on transparent handling of personal data in line with applicable regulation [27] and on keeping conversational answers factual [18].



E. Limitations

The present evaluation is a functional demonstration. No controlled user study, offline recommendation-accuracy benchmark, or load test has yet been conducted, and the small seeded catalog limits conclusions about ranking quality. The recommender is heuristic and does not learn from data. Analytics and velocity estimates count paid orders, so cash-on-delivery sales enter them only after delivery marks the order as paid. Payment confirmation currently relies on the browser callback verified by the server; gateway webhooks for reconciliation and automated refunds are planned. Usability will be measured with the System Usability Scale [28].

IX. CONCLUSION AND FUTURE SCOPE

This manuscript presented the design and prototype implementation of a Smart AI-Powered E-Commerce Platform that integrates personalized recommendation, a database-grounded multilingual shopping assistant, and sales and inventory analytics. By combining an explainable scoring engine, co-purchase suggestions, server-verified commerce, and role-based administration within one application, the platform addresses the main shortcomings of static small-business storefronts: generic discovery, language barriers, unreliable chatbot answers, and weak visibility of sales and stock.

The principal achievements are an explainable hybrid recommendation engine with a cold-start fallback; a three-stage assistant that never invents products or prices; transactionally consistent order and stock handling with online and cash-on-delivery payment; a layered security model with one-time-password two-factor login; and an analytics and restock-intelligence console for the merchant.

Subsequent development phases will focus on:

- Training collaborative-filtering and latent-factor models on accumulated interaction logs, with offline evaluation by ranking metrics [9][10].
- Adding payment-gateway webhooks, automated refunds, and reconciliation of paid orders.
- Extending demand estimation from recent averages to seasonal time-series models that capture festival-season demand.
- Introducing voice input and additional regional languages for the assistant.
- Conducting a System Usability Scale study and load testing with real shoppers [28].
- Adding consent management and data-retention controls aligned with the Digital Personal Data Protection Act [27].
- Providing a mobile application with push notifications.

In summation, this work shows that explainable, database-grounded AI components can be integrated into a conventional web store at modest cost, helping small retailers deliver a more personalized customer experience while giving owners the evidence needed to manage stock and growth.

REFERENCES

- [1]. N. Arora, D. Ensslen, L. Fiedler, W. W. Liu, K. Robinson, E. Stein, and G. Schüler, "The value of getting personalization right—or wrong—is multiplying," *McKinsey & Company*, Nov. 2021. [Online]. Available: <https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying>. (Accessed: Oct. 10, 2026).
- [2]. Bain & Company and Flipkart, "How India shops online 2020," Jun. 2020.
- [3]. P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.
- [4]. J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Min. Knowl. Discov.*, vol. 5, nos. 1–2, pp. 115–153, 2001.
- [5]. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.
- [6]. F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*, 2nd ed. New York, NY, USA: Springer, 2015.
- [7]. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, Hong Kong, 2001, pp. 285–295.
- [8]. G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.
- [9]. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
- [10]. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, Perth, Australia, 2017, pp. 173–182.
- [11]. S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, Art. no. 5, 2019.



- [12]. R. Burke, "Hybrid recommender systems: Survey and experiments," *User Model. User-Adapt. Interact.*, vol. 12, no. 4, pp. 331–370, 2002.
- [13]. N. Tintarev and J. Masthoff, "A survey of explanations in recommender systems," in *Proc. IEEE 23rd Int. Conf. Data Eng. Workshops (ICDEW)*, Istanbul, Turkey, 2007, pp. 801–810.
- [14]. J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proc. ACM Conf. Comput. Supported Cooperative Work (CSCW)*, Philadelphia, PA, USA, 2000, pp. 241–250.
- [15]. E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Mach. Learn. Appl.*, vol. 2, Art. no. 100006, 2020.
- [16]. D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2009.
- [17]. Gemini Team, Google, "Gemini: A family of highly capable multimodal models," *arXiv:2312.11805*, 2023.
- [18]. Z. Ji et al., "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, Art. no. 248, 2023.
- [19]. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.
- [20]. H. Chen, R. H. L. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS Q.*, vol. 36, no. 4, pp. 1165–1188, Dec. 2012.
- [21]. E. A. Silver, D. F. Pyke, and D. J. Thomas, *Inventory and Production Management in Supply Chains*, 4th ed. Boca Raton, FL, USA: CRC Press, 2016.
- [22]. OWASP Foundation, "OWASP Top 10:2021," 2021. [Online]. Available: <https://owasp.org/Top10/>. (Accessed: Oct. 10, 2026).
- [23]. M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," Internet Eng. Task Force (IETF), RFC 7519, May 2015.
- [24]. N. Provos and D. Mazières, "A future-adaptable password scheme," in *Proc. USENIX Annu. Tech. Conf. (FREEIX Track)*, Monterey, CA, USA, 1999, pp. 81–91.
- [25]. Razorpay, "Standard Checkout integration guide: Verify payment signature," Razorpay Docs. [Online]. Available: <https://razorpay.com/docs/developer-tools/integrations/standard-checkout/>. (Accessed: Oct. 10, 2026).
- [26]. MongoDB, Inc., "Transactions," MongoDB Node.js Driver Documentation. [Online]. Available: <https://www.mongodb.com/docs/drivers/node/current/fundamentals/transactions/>. (Accessed: Oct. 10, 2026).
- [27]. Government of India, Ministry of Law and Justice, "The Digital Personal Data Protection Act, 2023 (Act No. 22 of 2023)," Aug. 2023.
- [28]. J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189–194.