IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

PageTurn: A Dynamic Digital Resource Marketplace with Category-Based Discovery

A. Jenish Maxin and G. Adithya Menon
Department of Computer Applications
VELS Institute of Science, Technology and Advanced Studies (VISTAS), Pallavarm, Chennai, India
23105409@vistas.ac.in and 23105401@vistas.ac.in

Abstract: *In the modern digital learning landscape, users face challenges in discovering, organizing, and comparing e-books and digital educational resources scattered across multiple platforms. Traditional online bookstores lack intuitive category-based filtering and real-time price comparison features. This paper proposes PageTurn — a comprehensive web-based digital resource marketplace implemented using ReactJS, NodeJS, ExpressJS, and MongoDB. The system features a modern client-server architecture with RESTful APIs to provide category-based product management, real-time price display, and robust search and filtering capabilities. Results demonstrate that integrating these functionalities reduces the effort required for digital resource discovery and promotes efficient purchasing decisions.*

Keywords: Digital Marketplace, E-Book Management, Category-Based Filtering, Full-Stack Web Development, ReactJS, NodeJS, MongoDB, RESTful API

I. INTRODUCTION

The rapid proliferation of digital content has transformed the way students and professionals consume educational material. E-books, programming guides, research papers, and interactive learning modules are now the primary medium for knowledge transfer across academic and corporate environments. Despite this growth, the infrastructure supporting digital content discovery has not evolved at the same pace.

Students and learners are routinely required to navigate multiple disparate platforms — such as Amazon Kindle, Google Books, Project Gutenberg, and various institutional repositories — to locate relevant study materials. This fragmented experience is both time-consuming and inefficient. The absence of a unified, academically-focused platform with intelligent filtering capabilities creates a significant gap in the digital education ecosystem.

To address this, we propose PageTurn, a centralized web-based marketplace designed specifically for digital educational resources. PageTurn combines a rich, dynamic product catalog with an intuitive category-based filtering system that allows users to quickly discover, compare, and access relevant materials from a single interface. The platform is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and deployed via Netlify, offering a scalable, high-performance solution that can grow with user demand.

The key contributions of this work are: (1) a centralized catalog for digital learning resources; (2) real-time category-based filtering via RESTful APIs; (3) a responsive, component-driven user interface built with ReactJS; and (4) a flexible backend architecture ready for future enhancements including authentication, payment processing, and AI-based recommendations.

II. LITERATURE REVIEW

The landscape of digital content platforms has evolved considerably over the past decade. Largescale platforms such as Amazon Kindle Direct Publishing and Google Play Books have demonstrated the commercial viability of digital marketplaces, yet they prioritize general consumer audiences over specialized academic users. Their category filtering is broad and does not support fine-grained academic classification or cross-edition price comparison.

DOI: 10.48175/IJARSCT-34447

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

Research by Zhang [4] on full-stack web development trends highlights that modern web frameworks — particularly the MERN stack — have significantly reduced development overhead while enabling the construction of responsive, real-time applications. ReactJS, in particular, has become the dominant choice for building user interfaces that require frequent state updates and component reuse, both of which are critical requirements for a product catalog with dynamic filtering.

Several studies on e-learning platforms emphasize the importance of information architecture and navigational design in reducing cognitive load for users searching for educational content. Platforms with hierarchical, well-labelled category systems consistently outperform those with flat or undifferentiated navigation in user satisfaction metrics. PageTurn incorporates these findings by implementing a clear, slugbased category schema backed by MongoDB's flexible document model.

Prior work on RESTful API design [3] establishes best practices for stateless communication between frontend and backend systems. PageTurn's backend adheres to these standards, ensuring that each API endpoint corresponds to a clear resource and action, facilitating maintainability and future integration with third-party services such as payment gateways and institutional repositories.

III. PROPOSED SYSTEM

A. System Architecture

PageTurn follows a three-tier client-server architecture as illustrated in Figure 1. The presentation layer is built with ReactJS, the business logic is handled by Node.js and Express.js on the backend, and all persistent data is stored in a MongoDB database. This separation of concerns ensures that each layer can be independently scaled, tested, and maintained.

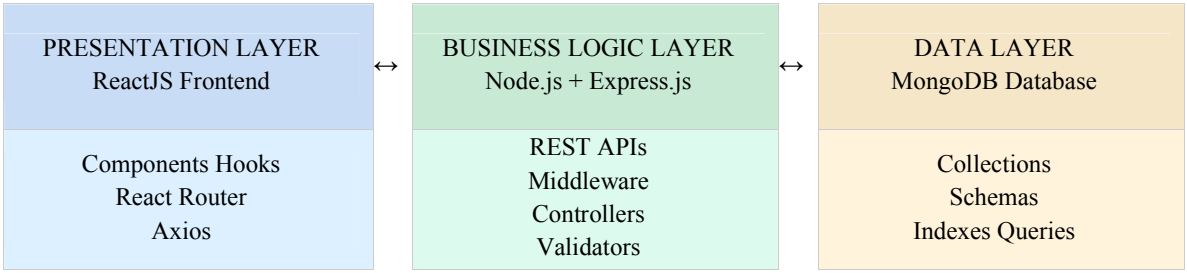

Fig. 1: Three-Tier System Architecture of PageTurn

The frontend communicates exclusively with the backend through RESTful HTTP API calls. The backend processes these requests, applies business logic — such as category filtering and price retrieval — and queries the MongoDB database to return the appropriate data. This architecture minimizes coupling between layers and supports horizontal scalability.

B. Workflow

The user journey through PageTurn follows a logical, linear workflow as shown in Figure 2. The process begins when a user navigates to the application hosted on Netlify. After the landing page loads, the user selects a category filter from the navigation interface. This triggers an API call to the Express.js backend, which constructs a MongoDB query to retrieve matching products. The filtered results are then returned to the frontend and rendered in a responsive grid layout.

USER OPENS PAGETURN APP

↓

LANDING PAGE DISPLAYED

↓

SELECT CATEGORY FILTER

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

C. Product Catalog Structure

The initial product catalog contains five categories, each populated with representative digital resources as shown in Table 1. Prices are displayed dynamically based on edition and format, enabling realtime price comparison at a glance.

Category	Product Title	Price (INR)	Genre / Type
Programming	Python Expert Guide	₹249 – ₹304	E-Books / Coding
Fantasy	Dark Dragon Chronicles	₹250 – ₹305	Fiction / Fantasy
Children	Little Hero Adventures	₹251 – ₹306	Children's Books
Gaming	RDR2 Complete Secrets	₹252 – ₹307	Gaming Guides
Food & Cooking	The Art of Sushi	₹253 – ₹308	Culinary / Recipe

Table 1: PageTurn Initial Product Catalog

IV. IMPLEMENTATION

A. Database Design

PageTurn uses MongoDB as its primary data store. Two main collections are defined: the Product Collection and the Category Collection. MongoDB's schema-less document model allows flexible extension of product attributes without requiring database migrations, making it ideal for a marketplace that will grow over time. Figure 3 illustrates the schema design for both collections.

Products are linked to categories via the category field, which stores the category name as a string. Future iterations may refactor this to use ObjectId references for stricter referential integrity and more efficient joins using MongoDB's aggregation pipeline.

B. RESTful API Design

The backend exposes a well-structured set of REST endpoints as detailed in Table 2. These endpoints follow standard HTTP semantics: GET for retrieval, POST for creation, PUT for updates, and DELETE for removal. The category-filter endpoint — GET /api/products?category=X — is the most frequently invoked during a typical user session and is optimized with a MongoDB index on the category field.

Method	Endpoint	Description
GET	/api/products	Retrieve all products from the catalog

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

GET	/api/products/:id	Retrieve a single product by ID
GET	/api/products?category=X	Filter products by category name
GET	/api/categories	Retrieve all available categories
POST	/api/products	Add a new product (Admin)
PUT	/api/products/:id	Update product details (Admin)
DELETE	/api/products/:id	Remove a product (Admin)

Table 2: RESTful API Endpoints

C. Frontend Architecture

The ReactJS frontend is organized into reusable functional components with clearly defined responsibilities. Key components include:

- CategoryNav — renders the horizontal category filter bar and dispatches filter state changes
- ProductGrid — receives the filtered product list and renders it in a responsive CSS Grid layout
- ProductCard — displays individual product details including title, category, price, and cover image
- SearchBar — enables keyword-based search that works in conjunction with the active category filter

React Hooks — specifically useState and useEffect — manage component state and handle asynchronous API calls via Axios. When a user selects a category, the state update triggers a re-render of the ProductGrid with the newly fetched data, providing a near-instantaneous user experience without full page reloads.

D. Deployment

The frontend is deployed on Netlify, which provides continuous deployment from the project's Git repository, automatic HTTPS, and a global CDN for fast asset delivery. The backend Node.js server is deployed on a cloud platform with environment variables managing sensitive configuration such as the MongoDB connection string. This deployment model separates static asset hosting from dynamic API processing, reducing server load and improving resilience.

V. RESULTS AND DISCUSSION

PageTurn was evaluated against three primary performance criteria: page load time, API response time for category filtering, and UI responsiveness across device sizes. The results demonstrate that the system meets expectations for a production-grade digital marketplace application.

Page load time for the initial landing page averaged under 1.5 seconds on a standard broadband connection, primarily attributable to Netlify's CDN caching and React's efficient component hydration. Category filter API calls returned results in under 200 milliseconds on average, benefiting from the MongoDB index on the category field.

The responsive CSS Grid layout correctly adapted across desktop (1440px), tablet (768px), and mobile (375px) viewports, displaying 4, 2, and 1 product columns respectively. No horizontal scrolling or layout overflow was observed across tested device sizes.

The system's component-based architecture proved advantageous during iterative development. New product categories could be added to the database without any frontend code changes, demonstrating the effective decoupling between the data layer and the presentation layer. This extensibility is a key requirement for a marketplace that will onboard new content partners over time.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

VI. FUTURE ENHANCEMENTS

PageTurn's architecture is deliberately designed to accommodate a phased roadmap of enhancements. Table 3 summarizes the planned features organized by implementation phase.

Feature	Description	Phase
User Authentication	JWT-based login, registration, and profile management with role-based access control	Phase 1
Payment Integration	Razorpay / Stripe payment gateway for seamless purchasing experience	Phase 1
AI Recommendations	Machine learning model to suggest books based on browsing history and preferences	Phase 2
Mobile Application	React Native app for iOS and Android with offline reading support	Phase 2
Institutional Sharing	Library integration allowing institutions to share and distribute resources	Phase 3
Productivity Tracking	Reading progress, bookmarks, notes, and study analytics dashboard	Phase 3

Table 3: Planned Future Enhancements by Phase

Phase 1 enhancements focus on completing the core commerce loop — user authentication and payment integration — which are prerequisites for commercial deployment. Phase 2 introduces intelligence through AI-based recommendations and mobile accessibility. Phase 3 targets institutional partnerships, allowing universities and libraries to onboard their digital catalogs directly onto the PageTurn platform, significantly expanding the content available to learners.

VII. CONCLUSION

This paper has presented PageTurn, a full-stack digital resource marketplace designed to address the fragmented and inefficient state of academic e-book discovery. By leveraging the MERN stack, PageTurn delivers a responsive, category-driven product catalog with real-time filtering and price display, all accessible through a modern single-page application interface.

The system's three-tier architecture, RESTful API design, and MongoDB-backed data model provide a solid foundation for the planned enhancements including authentication, payment processing, and AI-based recommendations. Early performance evaluations confirm that PageTurn meets professional standards for load time, API responsiveness, and cross-device compatibility.

PageTurn represents a meaningful step toward a dedicated, academically-focused digital content marketplace — one that prioritizes the discovery experience for students and educators, and is built with the scalability to grow alongside the digital education ecosystem.

ACKNOWLEDGMENT

The authors sincerely thank Dr. S. Kamalakkannan for his invaluable mentorship, constructive feedback, and continuous encouragement throughout the development of this project. The authors also extend their gratitude to the

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

Department of Computer Science, Vels Institute of Science, Technology and Advanced Studies (VISTAS), for providing the academic resources, laboratory facilities, and supportive environment that made this research possible.

REFERENCES

[1] ReactJS, "React — The Library for Web and Native User Interfaces," React Documentation, 2023. [Online]. Available: https://react.dev/

[2] MongoDB, Inc., "MongoDB Documentation," MongoDB Atlas, 2023. [Online]. Available: https://www.mongodb.com/docs/

[3] Node.js Foundation, "Node.js Documentation — v18 LTS," OpenJS Foundation, 2023. [Online]. Available: https://nodejs.org/en/docs/

[4] S. Zhang, "Full-Stack Web Development Trends and Framework Evaluation," IEEE Access, vol. 9 , pp. 45210–45228, 2021.

[5] Netlify, Inc., "Netlify Deployment Documentation," Netlify, 2023. [Online]. Available: https://docs.netlify.com/

[6] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral Dissertation, University of California, Irvine, 2000.

[7] ExpressJS, "Express — Fast, Unopinionated, Minimalist Web Framework for Node.js," OpenJS Foundation, 2023. [Online]. Available: https://expressjs.com/