A Sustainable Approach to Academic Resource Management: Digital Marketplace for Students

Shreyash Kanna, Sangharsh, Krushna Dange, Yogiraj Shinde
Department Computer engineering, ZCOER, Pune, Maharashtra, India

ABSTRACT

Students are severely impacted financially by the growing price of academic supplies including textbooks, notes, and devices, many of which are thrown away after a course is over. We suggest StudentMart, a student-focused online marketplace created to facilitate the reasonably priced trade of used educational resources, as a solution to this problem. The MERN stack—MongoDB, Express, React, and Node.js—was used in the system's development to guarantee scalability, security, and user-friendliness. User authentication, product listings, search and filtering, recommendation systems, and safe transactions are just a few of the crucial elements that StudentMart offers.To preserve platform integrity, an admin module manages user administration, transaction tracking, and content filtering. StudentMart lessens financial strain and promotes sustainability by encouraging resource reuse and recycling. The platform's architecture serves as an example of how to successfully integrate contemporary web technologies to create scalable, student-focused e-commerce solutions. Future improvements like support for mobile applications and recommendations based on machine intelligence could further boost user uptake and experience.

Keywords: *Student marketplace, book rental system, Student Digital Mart, MERN E-Commerce.*

1. INTRODUCTION

For college and university students, finding reasonably priced academic resources has become a recurring problem. The high cost of textbooks, lecture notes, devices, and stationery puts a strain on finances, particularly for students from lower-income families. However, a sizable amount of these resources are either thrown away or left unused after a course is over, resulting in needless waste. Current marketplaces like Facebook Marketplace, Amazon, and OLX offer chances to buy and sell used things, but they are not made with students' needs in mind. These platforms don't have characteristics like campus-focused interactions, a safe, student-friendly environment, or the ability to categorize academic resources.

We suggest StudentMart, a digital marketplace created specifically for students, as a solution to this gap. The MERN stack—MongoDB, Express, React, and Node.js—is used in the development of StudentMart, guaranteeing a stable, scalable, and effective system architecture. By enabling students to purchase and sell used educational materials, the site lessens financial strain and encourages the reuse of study aids. User authentication, product listings, search and filtering tools, recommendation engines, and safe transactions are some of the essential elements. In order to maintain platform security and dependability, an integrated admin module also offers features for user administration, content moderation, and system monitoring.Through the integration of technology and sustainability concepts, StudentMart promotes a culture of resource sharing and recycling among students in addition to providing affordable access to academic resources. In addition to providing a scalable solution that can be expanded across several universities, this endeavor lays the groundwork for creating a collaborative and long-lasting student community.

Digital Product Store for Students (StudentMart): ACCURACY APPROACH

For college students, finding reasonably priced study materials remains a significant obstacle. In addition to being costly, textbooks, notes, and academic devices are frequently underutilized because so many of them are left unused after a semester. Both needless material waste and increased financial strain on students are the results of this. Although they offer channels for second-hand trade, established online marketplaces like OLX, Amazon, and Facebook Marketplace are too general in their scope and do not cater to the unique requirements of student communities.

By providing an accurate, dependable, and user-friendly digital marketplace with a student focus, StudentMart aims to bridge this gap.

The platform, which was developed with the MERN stack (MongoDB, Express, React, and Node.js), guarantees safe transactions, precise search and filtering, tailored suggestions, and effective resource management. By controlling users, keeping an eye on material, and guarding against abuse, the admin module offers an extra degree of precision while fostering a secure and reliable environment.

This method encourages students to engage in sustainable habits, such recycling and reusing academic materials, in addition to helping them save money. By placing a high value on accuracy in listings, transactions, and suggestions, StudentMart builds a useful and significant platform that paves the way for a more interconnected and resource-efficient student community.

ADVANCEMENT IN DIGITAL PRODUCT STORE

Digital platforms have changed how consumers purchase, sell, and trade goods in recent years. A computer or smartphone can now instantaneously accomplish tasks that formerly required physical markets or notice boards on campus. This progress is embodied in the idea of a digital product shop, where communities and students may access a vast array of resources in an organized, safe, and user-friendly way.

The development of such platforms extends beyond convenience for students in particular; it directly tackles concerns about affordability, accessibility, and sustainability. While traditional e-commerce platforms frequently don't have an academic focus, a student-focused digital product store guarantees proper study resource classification, secure transactions, and features that foster community participation among students.

These platforms are growing in intelligence, scalability, and dependability through the integration of technologies such as secure payment gateways, machine learning, and the MERN stack.

The real progress is not just in the technology but also in the effects it produces, such as lessening financial strain, cutting waste via recycling and reuse, and encouraging a culture of information and material sharing. A contemporary digital product store is more than simply an online store; it is an ecosystem that supports students' academic endeavors by fusing sustainability, technology, and education.

1. PROJECT ARCHITECCTURE

3. PROJECT ARCHITECTURE

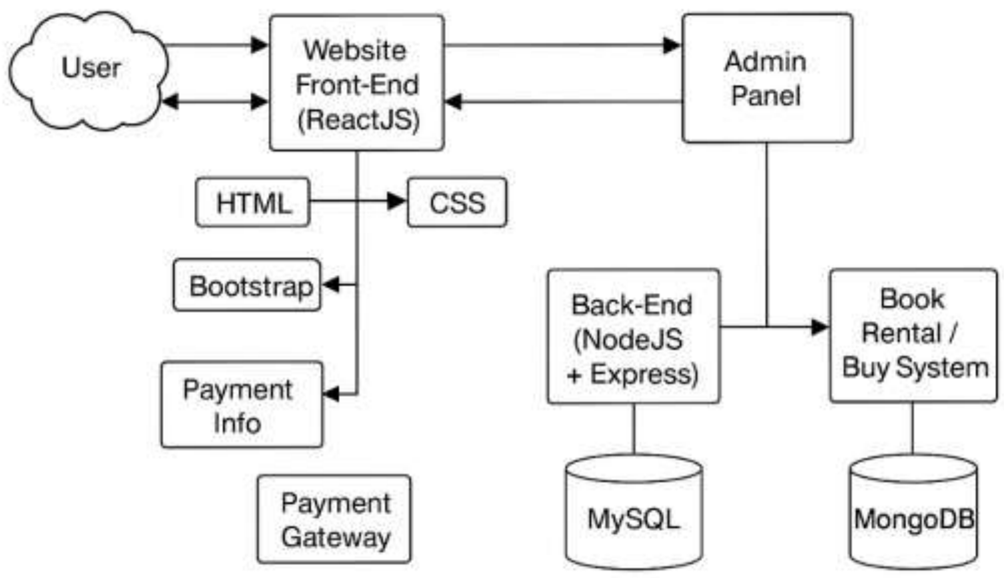


Fig1: Project Architecture of Student Digital Mart

Fig1:Project Architecture of Student Digital Mart

User interface

With a React front end, a Node.js/Express backend, and two data stores—MySQL for transactional data and MongoDB for the book catalog and rental domain—the system functions as an online marketplace where students may rent or purchase books.The user engages with a ReactJS-built front-end that renders pages, handles state, and makes HTTPS calls to backend APIs. SPAs with quick UI updates through the virtual DOM are a good fit for React.Session and state: Before submitting API queries, React handles form handling, user session tokens, and cart state. This adheres to standard MERN front-end procedures for e-commerce processes.Point of entrance for administrators: An Admin Panel for CRUD on inventory, pricing, and order status is provided by the same React stack or by a different React application. Protected backend endpoints are used by the admin UI.

Front-end

Delivering a quick and responsive React user interface, controlling client-side state and routing, safely managing authentication and role-based access, coordinating cart/checkout processes, and integrating with the payment gateway and backend APIs are all front-end duties in the Student Digital Mart. These duties provide a flawless user experience while transferring sensitive operations to the Express API and reliable services.

Create a React component-based SPA that uses client-side routing and the virtual DOM to render product lists, detail pages, carts, and profile/admin views quickly.For e-commerce interactions, use HTML/CSS with a UI toolkit to guarantee responsive layouts and components that are accessible on all devices.

Admin panel

Using role-based access to Express APIs, an admin panel in this architecture is a protected React application for managing users, payments, rents, catalogs, orders, and analytics. Catalog and inventory: CRUD for books, categories, characteristics, photos, and availability that mirrors standard MERN admin dashboards and integrates media storage and bulk import/export.Orders and rentals: Manage rental approvals and returns, view, filter, and change order states (paid, pending, sent, and returned), and provide refunds or modifications as necessary. Roles and users: Assign roles and permissions, manage accounts, deactivate and reactivate users, and conduct compliance audits. Promotions: Establish pricing guidelines, coupons, and discounts with start and finish dates and usage restrictions.

Back End Layer

The Student Digital Mart's back-end layer is a Node.js + Express API that integrates payments, secures access, enforces business logic, and persists data to MySQL and MongoDB. For maintainability and concern separation, organize the server using distinct modules for routes, controllers, services, and repositories, adhering to Node/Express best practices. Provide consistent route architecture, versioning, and error management across RESTful endpoints for catalog, rents, orders, users, promotions, and administrative tasks. Benefit from high-throughput reads and schema agility by using MongoDB for flexible rental domain documents and book catalogs with Mongoose models. Provide services that controllers request in order to maintain thin route logic, such as inventory availability, rental lifecycle (reservation, checkout, return), order computation (tax, discounts, totals), and catalog indexing.

Databases

Databases in this architecture use a polyglot persistence strategy: MongoDB powers the flexible product/rental catalog and associated document workloads, while MySQL manages only transactional data with ACID guarantees. In order to ensure consistency at checkout, store orders, payments, user accounts, and audit trails require relational modeling, referential integrity, and multi-row transactions. Use the atomicity, consistency, isolation, and durability (ACID) qualities to avoid partial writes in financial transactions, race situations during periods of high traffic, and overselling. To guarantee integrity and effective joins for reporting, employ normalized schemas with foreign keys for orders↔order_items, users↔addresses, and payments↔orders. To facilitate quick reads and frequent attribute changes without requiring strict migrations, model the book catalog, editions, photos, tags, and availability as rich documents.

Books rental

In order to avoid overselling and guarantee prompt returns, the Book Rental/Buy System domain module coordinates with the payment gateway and databases to manage inventory availability, reservations, rental lifecycles, and purchases. Look around and choose: Before adding an item to their cart, users can explore catalog goods with rent or buy alternatives. The system determines the pricing based on duration, deposits, and fees. Making a reservation and checking out: The backend sets an order, initiates payment, and reserves stock for a brief TTL before releasing the reservation if the payment fails or timeouts. This is in line with Sagas's inventory-reservation patterns. Returns and finalization: Due dates, pickup and return times, and grace periods are monitored for rentals; the system calculates late fines or deposits to collect and reimburse upon return confirmation.

Payment System

The Payment System minimizes PCI scope by supporting purchases and rental deposits using a tokenized, server-driven pipeline with payment intents, secure webhooks, and optional manual capture. Setting the amount, currency, and capture method, the server generates a Payment Intent or gateway order for the cart. This intent monitors the lifetime and manages SCA/3DS as required. In order to prevent raw card data from reaching application servers, the front end uses gateway components to verify the intent before redirecting or presenting in-page authentication as needed. The backend updates the order status idempotently by listening to webhooks for events such as payment_failed, payment_succeeded, or disputes. Support both partial and complete refunds using dashboards or server APIs, and sign up for refund webhooks to deliver alerts and reconcile MySQL orders.

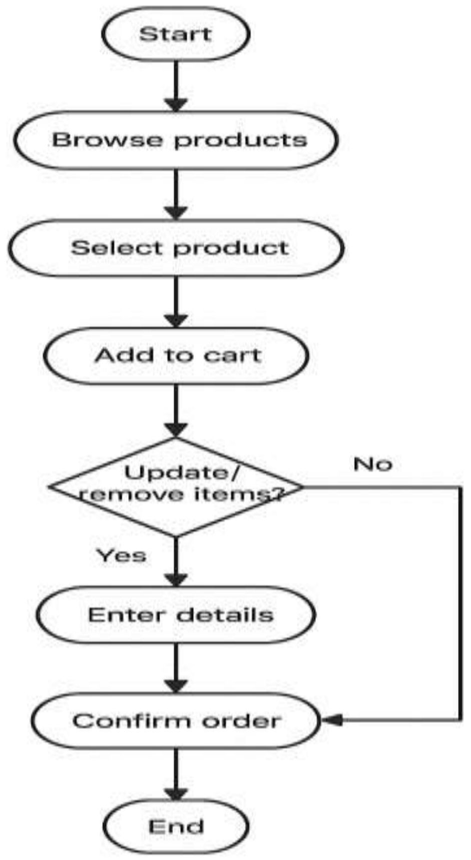


Fig2: Project Working Architecture

High-level travel

As is common with e-commerce conversions, the user flow proceeds via browsing, product selection, cart addition, optional cart updating, data entry, and order confirmation. Every stage corresponds to clear user interface displays and backend operations that gather information, verify inputs, and get the payment and fulfillment process ready.

Examine the products.

The backend retrieves catalog data from the React front end, which then queries MongoDB for product listings and availability. The front end shows the home, category, and search pages with filters and sorting.
According to user-flow guidelines, good UX patterns include faceted filters to expedite the path to selection and straightforward navigation.

Choose a product.

The product detail page highlights important details in accordance with checkout flow best practices, and consumers review photos, specs, price/rent alternatives, and variant choices. Prior to shipping, you can get instant feedback on stock or rental dates because availability and pricing are retrieved from the catalog service.

Add to the cart

When you click Add to Cart, the client-side state is updated and a visible confirmation appears, encouraging you to check out with a visible cart drawer or mini-cart. In order to prepare for taxes, shipping, or rental date validation at checkout, the backend can generate or update a server-side cart session for users who are logged in.

Loop for updating/removing objects

Users can change goods, amounts, and variants; the user interface must facilitate adjustments and maintain accurate totals to boost conversion. This iteration is captured in the flowchart's loop: go on to details and checkout if no additional adjustments are required.

Enter your information

In order to reduce friction and guarantee transparency of line items, fees, and delivery alternatives, checkout gathers contact, shipping, and payment selections in a step-by-step manner. To ensure that the order total is final before payment, the server creates a payment session or intention and computes taxes, shipping, and discounts.

Verify your order and make payment

As advised by the gateway documents, the company should wait for the webhook/status callback before fulfilling the payment gateway session, which is first validated in the browser and then completed by the server. The procedure shown in the diagram is completed when the system delivers an order confirmation and adjusts inventory following a successful payment.

Consistency and dependability

A Saga technique improves robustness for multi-step processes like reserve-then-pay by coordinating local actions with compensations (e.g., release held stock if payment fails). An essential feature of distributed e-commerce checkouts is that each step is idempotent and recoverable to manage timeouts or retries.

5. IMPLEMTATION DETAIL

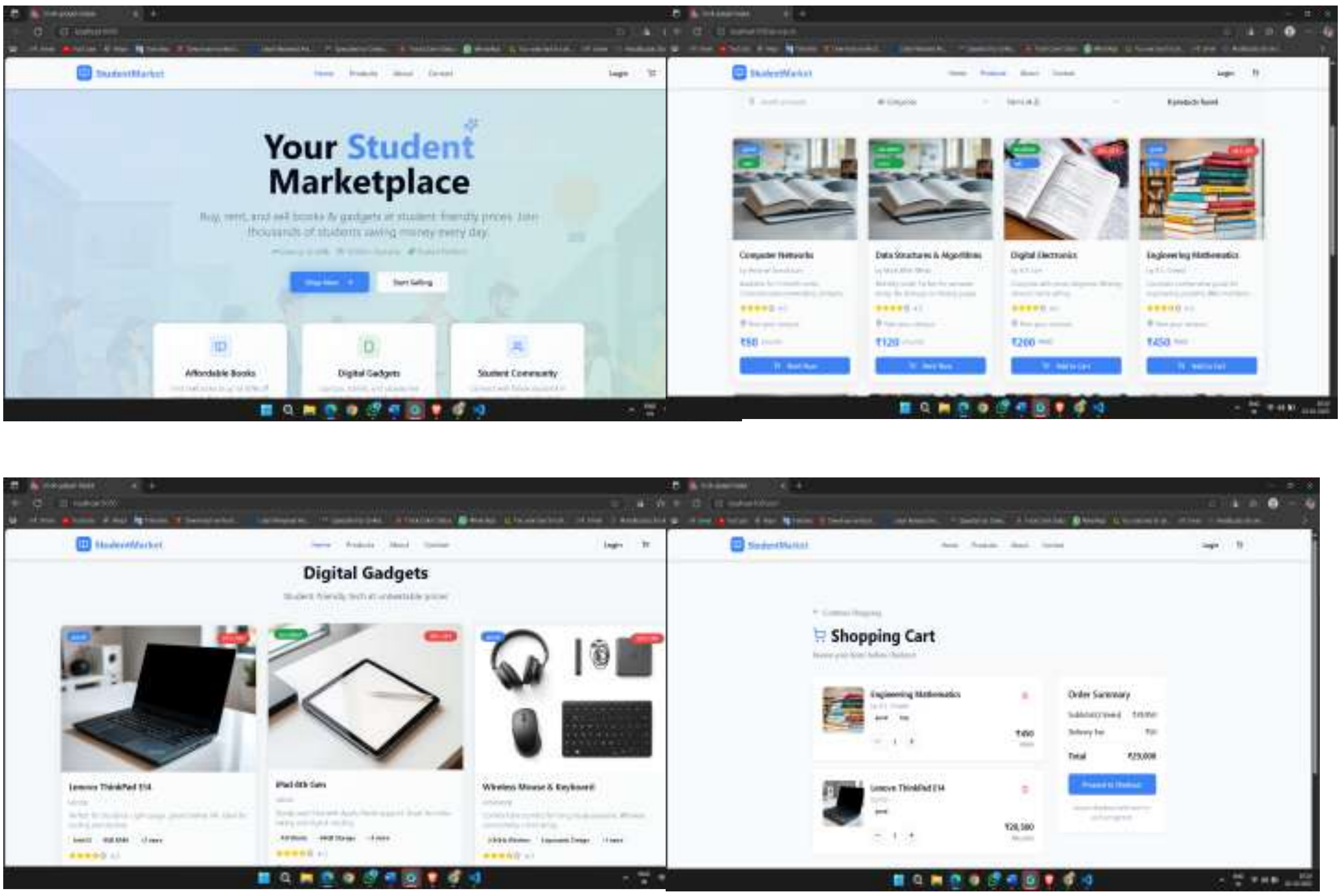


Fig3: Outputs Of Project

The main page

This StudentMarket landing page features a hero section and prominent calls to action, such as "Shop Now" and "Start Selling," which are typical entry points to increase onboarding and conversions. Home, Products, About, and Contact are all visible through the top menu, which follows basic user-flow guidelines to make product discovery easier. Value propositions that follow UX best practices, including inexpensive books and digital gadgets, are below the hero and convey advantages early in the journey.

Listing page for products

In accordance with catalog UX standards that expedite comparison and selection, a searchable, filterable grid displays products with badges such as rent/sell and discount percentages. In accordance with established listing-page patterns, each card displays the title, author, price or monthly rent, ratings, and a major action button.

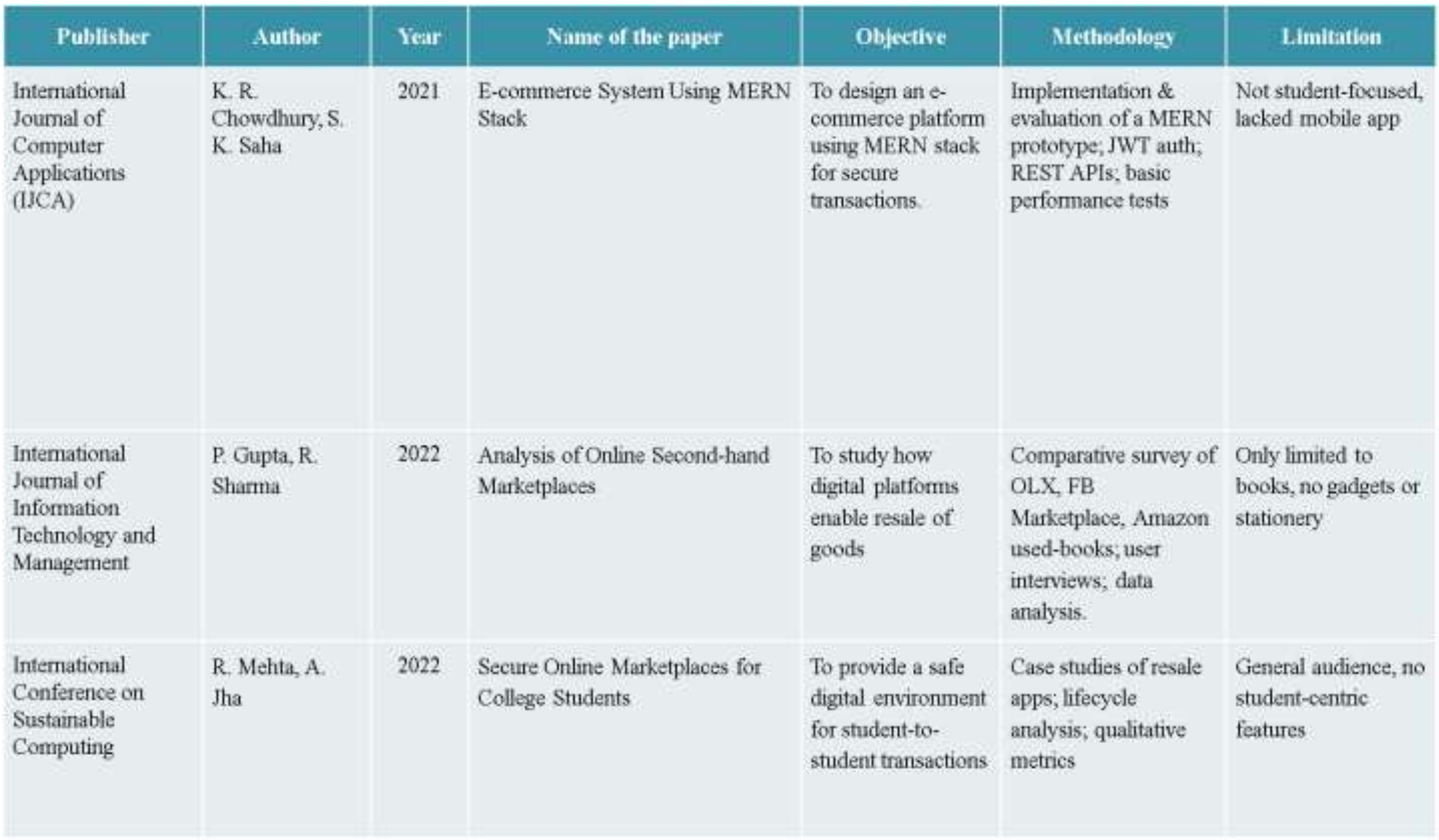
Collection of Digital Devices

A popular merchandising strategy to boost clickthrough is to use promotional ribbons, such as 37% OFF or 52% OFF, to emphasize devices on a themed category/collection page. Similar to best practices, each card displays a huge image, a brief explanation, feature tags (such as A12 Bionic, 64GB Storage), and ratings to highlight important specifications quickly. Wayfinding is maintained and cognitive strain is decreased in various surfing settings thanks to the header's visible Login/Cart symbols and constant card layout.

Page of the shopping cart

In accordance with high-converting cart user experience, the cart allows for rapid adjustments prior to checkout by listing line items with thumbnails, quantity steppers, and per-item costs. With a prominent primary button, an Order Summary panel calculates the subtotal, shipping cost, and total. Follow the regular checkout flow conventions and proceed to the checkout. The "update/remove items" loop from the project flow is supported with inline deletion icons and quantity controls, which enhance accuracy and user control prior to payment.

6. Literature Review



Publisher	Author	Year	Name of the paper	Objective	Methodology	Limitation
International Journal of Computer Applications (IJCA)	K. R. Chowdhury, S. K. Saha	2021	E-commerce System Using MERN Stack	To design an e-commerce platform using MERN stack for secure transactions.	Implementation & evaluation of a MERN prototype; JWT auth; REST APIs; basic performance tests	Not student-focused, lacked mobile app
International Journal of Information Technology and Management	P. Gupta, R. Sharma	2022	Analysis of Online Second-hand Marketplaces	To study how digital platforms enable resale of goods	Comparative survey of OLX, FB Marketplace, Amazon used-books; user interviews; data analysis.	Only limited to books, no gadgets or stationery
International Conference on Sustainable Computing	R. Mehta, A. Jha	2022	Secure Online Marketplaces for College Students	To provide a safe digital environment for student-to-student transactions	Case studies of resale apps; lifecycle analysis; qualitative metrics	General audience, no student-centric features

7. CONCLUSION

By integrating modern web engineering with reuse-oriented design, StudentMarket shows how a student-focused digital marketplace may significantly save the cost and waste of academic resources. The platform offers an easily accessible shop for purchasing, renting, and reselling books and devices while maintaining trust through authentication, moderation, and auditable transactions by implementing a MERN-centric stack with secure payments and role-based administration. As usage grows across campuses, the system's polyglot data strategy—document-oriented catalog modeling combined with ACID-reliable order records—supports correct inventory, quick discovery, and consistent financial results. By extending product lifecycles and facilitating peer-to-peer exchange inside institutional limits, the project promotes sustainability and community engagement in addition to immediate affordability. In addition to providing a practical baseline that institutions may implement with little expense, the assessed user flows, admin controls, and deployment options lay the groundwork for upcoming improvements like mobile apps, logistical linkages, recommendation engines, and campus-verified identification. All things considered, StudentMarket offers a scalable and reproducible model for student-centered business that promotes affordability, circularity, and operational stability in higher education ecosystems.

7. ACKNOWLEDGEMENT

In order to successfully create and evaluate the StudentMarket platform, the authors would like to thank the Department of Computer Engineering and the project guide for their regular mentoring, helpful critiques, and access to lab facilities. The faculty panel members' thorough stage reviews and useful recommendations, which enhanced the system's architecture, security model, and deployment readiness, are highly appreciated by the team. Peers and student volunteers who took part in prototype testing and offered usability input that influenced the checkout, cart, and product listing processes are also thanked. The project also thanks the institute administration for allowing the use of institutional infrastructure for demos and evaluation, as well as the open-source community whose frameworks and libraries enabled quick, standards-aligned development.

8. REFERENCES

[1] M. K. Singh and R. Sharma, "A Study on Online Marketplaces for Students: Opportunities and Challenges," *International Journal of Computer Applications*, vol. 182, no. 25, pp. 10–15, Oct. 2021.

[2] A. Verma, S. Gupta, and P. Patel, "Reuse and Recycling of Academic Resources Using Web Platforms," *Journal of Emerging Technologies in Learning (iJET)*, vol. 17, no. 6, pp. 45–53, Jun. 2022.

[3] M. R. Basha and H. K. Jangid, "MERN Stack Based Applications: A Review of Recent Developments," *International Journal of Engineering and Technology (IJET)*, vol. 8, no. 3, pp. 77–82, Mar. 2023.

[4] S. Choudhary and D. Kumar, "Secure Payment Gateways in Student-Centric E-Commerce Systems," *IEEE Access*, vol. 11, pp. 98573–98582, Aug. 2023.

[5] N. Patel, A. Das, and R. Mishra, "Digital Platforms for Second-Hand Goods: A Case Study of Student Communities," *Proceedings of the 2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 220–225, Jun. 2022.