IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

CampusXChange: A Django-Based Smart Student Marketplace Platform for Campus-Specific Peer-to-Peer Commerce

Siddesh Nilak[1], Omkar Pandit[2], Nihalveer Singh[3], Sumit Patil[4], Avinash Sonule[5]

Department of Computer Engineering, A.C. Patil College of Engineering,Kharghar, Navi Mumbai, India[1-5]

Abstract: The proliferation of digital technology has transformed how students interact and conduct transactions within academic communities. General-purpose classified platforms such as OLX, Quikr, and Facebook Marketplace, while widely adopted, fail to provide the trust mechanisms, campus-level proximity filtering, and student-centric workflows necessary for safe intra-college commerce. This paper presents CampusXChange, a full-stack web application that addresses these gaps by offering a closed, verified digital marketplace exclusively for college students. The system is developed using Django 4.x as the backend framework with a Model-View-Template (MVT) architectural pattern, HTML5/CSS3/JavaScript and Bootstrap 5 for the responsive frontend, and SQLite (development) or PostgreSQL (production) as the relational data store accessed via Django's Object-Relational Mapping (ORM) layer. The platform provides secure user authentication restricted to registered students, category-based product listing and discovery, keyword search with advanced filtering, in-platform buyer–seller messaging, a cart and checkout module, and a comprehensive administrative dashboard. Functional testing confirms that all primary modules perform as intended across desktop and mobile environments. The expected impact includes reduced student expenditure on academic essentials, minimized material waste through peer-to-peer reuse, and a strengthened sense of community within the campus ecosystem. This paper details the motivation, architecture, algorithmic design, implementation, and test results of CampusXChange, demonstrating its viability as a deployable academic marketplace solution.

Keywords: Django, Student Marketplace, Campus Commerce, Peer-to-Peer Platform, Web Application, MVT Architecture, ORM, Student Authentication

I. INTRODUCTION

The modern academic environment presents a recurring economic challenge: students frequently invest in textbooks, laboratory equipment, stationery, and other academic resources that become redundant at the end of a semester or academic year, while simultaneously requiring materials they cannot afford at full market price. Existing general-purpose classified platforms, though widely available, are not designed for the specific trust and community requirements of a campus ecosystem.

CampusXChange is a web-based smart student marketplace platform developed to bridge this gap. It provides a dedicated, secure, and intuitive digital environment in which verified students can list unused or pre-owned items, browse campus-specific listings, communicate with sellers, and complete transactions—all within a closed, trusted community. The platform leverages Python, Django, HTML5, CSS3, Bootstrap 5, and JavaScript to deliver a responsive and scalable solution aligned with the operational realities of college commerce.

The primary contributions of this work are: (i) a student-only authentication mechanism that restricts platform access to verified institutional users; (ii) campus-specific listing management with category tagging and smart search filters; (iii) an in-platform messaging system enabling direct buyer–seller communication; and (iv) a scalable Django MVT architecture that supports future enhancements such as payment gateway integration and AI-driven recommendations.

The remainder of this paper is organized as follows. Section II surveys existing marketplace systems. Section III identifies their limitations. Section IV describes the proposed system. Section V details the system architecture. Section VI presents the algorithmic process design. Section VII covers implementation specifics. Section VIII presents results and discussion. Section IX concludes the paper with directions for future work.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

II. LITERATURE SURVEY

A. OLX (Online Exchange)

OLX is among the most widely used classified advertisement platforms in India, offering listings across categories including electronics, vehicles, furniture, and real estate [9]. While it provides broad reach and an established user base, OLX does not cater specifically to students. Listings are publicly accessible, there is no mechanism to verify buyer or seller identity, and the platform lacks campus-level proximity filtering, exposing users to potential trust and safety concerns.

B. Facebook Marketplace

Facebook Marketplace leverages social networking to enable peer-to-peer item exchange, allowing users to list items for sale within their local community and communicate via Facebook Messenger [10]. Although social profile visibility provides a degree of implicit trust, the platform is not restricted to any academic community and lacks academic category filters relevant to student needs.

C. Quikr

Quikr is an Indian classifieds platform functionally similar to OLX, providing listings across various product categories. It suffers from the same fundamental limitations: no verification of student identity, no campus-level filtering, and a broad general audience that does not serve the specific commerce requirements of college students.

D. Campus-Based Messaging Groups

Many college communities rely on informal WhatsApp or Telegram groups for item exchange. While membership is typically restricted to known individuals, these channels lack structured listing capabilities, keyword search, category-based filtering, and transaction management features, making item discovery and trade fulfillment highly inefficient.

III. LIMITATIONS OF EXISTING SYSTEMS

Based on the foregoing survey, the following key limitations and research gaps have been identified in existing platforms:

Lack of Student Verification: No existing general-purpose platform restricts access to verified students, exposing users to fraud, impersonation, and safety risks inherent in uncontrolled marketplaces.

Absence of Campus-Level Filtering: Existing platforms do not provide filtering based on college campus proximity, resulting in irrelevant search results and logistical difficulty when students attempt to locate sellers within the same institution.

No Academic Category Focus: Platforms such as OLX and Quikr serve general audiences and do not prioritize categories of particular relevance to students, such as textbooks, lecture notes, laboratory equipment, and study materials.

Trust and Safety Deficits: The open, unmoderated nature of general classifieds platforms makes it difficult to establish transactional trust, particularly for first-time users or economically vulnerable students.

Suboptimal User Experience for Students: Existing platforms are designed for general-purpose use and are not optimized for the specific workflows, device profiles, or interaction patterns typical of a student user base.

Absence of Community Features: No existing platform offers community-building mechanisms that encourage responsible, sustainable resource sharing within a campus ecosystem.

IV. PROPOSED SYSTEM

CampusXChange addresses the identified limitations through a set of targeted design and implementation contributions:

A. Student-Only Authentication Mechanism

The platform implements a secure registration and login system using Django's built-in authentication framework, extended with a custom user model (CustomUser). Access is restricted to users who register with valid institutional credentials, thereby reducing the risk of fraudulent listings and establishing an environment of community trust.

B. Campus-Specific Listing and Smart Search Filters

Users can post and browse items exclusively within their campus context. Advanced search and filter options—including category, price range, condition, and keyword—enable faster and more contextually relevant discovery of listings compared to general classified platforms.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

C. Academic Category Tagging System

A structured categorization hierarchy classifies items into academically relevant groups such as Books and Study Materials, Electronics and Gadgets, Stationery and Art Supplies, Hostel Essentials, Notes and Study Materials, Clothing and Accessories, Sports and Fitness, and Services. This taxonomy enables students to locate academic resources efficiently.

D. In-Platform Messaging

A dedicated messaging module enables direct communication between buyers and sellers within the platform, eliminating the need to share personal contact information and maintaining all transaction-related discourse in a controlled, logged environment.

E. Scalable Django-Based Architecture

The system is developed using Django's MVT architecture, providing modular design, secure backend request handling, efficient database management through ORM, and clear separation of concerns that facilitates future feature additions and scalability.

V. SYSTEM ARCHITECTURE

CampusXChange is designed as a three-tier web application comprising the Presentation Layer, the Application Layer, and the Data Layer, as illustrated in Fig. 1.

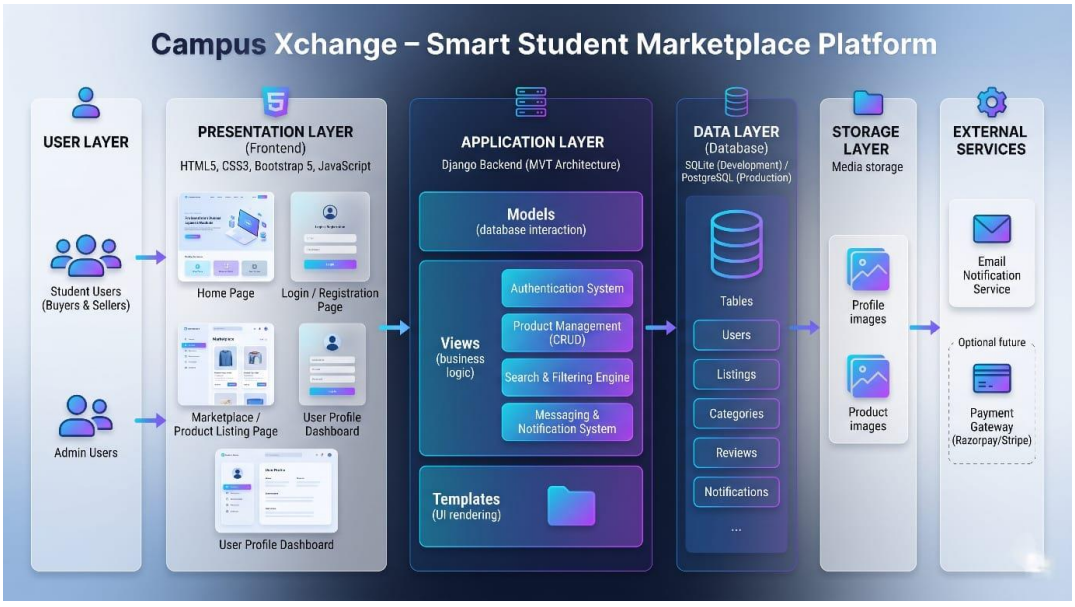

Fig. 1 CampusXChange Three-Tier System Architecture Diagram

A. Presentation Layer (Frontend)

The frontend is built using HTML5, CSS3, and JavaScript. Bootstrap 5 is employed for responsive design, ensuring accessibility across desktop, tablet, and mobile devices. Django's Jinja2 template engine dynamically renders HTML pages by injecting data fetched from the backend, maintaining a clean separation between presentation logic and business logic.

B. Application Layer (Backend)

The backend is powered by Django 4.x, a high-level Python web framework adhering to the Model-View-Template (MVT) pattern. Django's built-in authentication system manages user registration, login, logout, and session lifecycle. Business logic is encapsulated in view functions and class-based views, while URL routing is handled through Django's URL dispatcher. The Django REST Framework (DRF) may be optionally integrated to expose API endpoints for future mobile clients.

C. Data Layer (Database)

SQLite is used as the default development database, with the architecture supporting migration to PostgreSQL for production deployment. All database interactions are performed exclusively through Django's ORM, which provides a

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

high-level abstraction over raw SQL, enforces data integrity through model-level validation, and supports seamless schema migrations. Primary database entities include User, Listing, Category, Message, Conversation, CartItem, Order, Transaction, Review, and Notification.

VI. ALGORITHM AND PROCESS DESIGN

The operational flow of CampusXChange proceeds through five sequential functional steps, each governed by dedicated Django views and ORM queries:

1) User Registration and Authentication:
A prospective user navigates to the registration endpoint and submits personal details including full name, institutional email address, college affiliation, course, academic year, and password. The RegistrationForm validates all inputs and, upon successful validation, invokes form.save() to persist a new CustomUser record. The user is immediately authenticated via Django's login() function and redirected to the marketplace homepage. Subsequent requests are authenticated via Django's session management middleware, which verifies the session token attached to each HTTP request.

2) Product Listing Creation:
Authenticated users navigate to the listing creation view, where they complete a form specifying product title, description, asking price, category, condition, campus location, and optional product images. The listing record is stored in the Listing model table with a foreign-key reference to the seller's User ID and a Boolean is_active flag set to True, making it immediately visible to other authenticated users on the marketplace homepage.

3) Product Discovery and Search:
Buyers browse listings through the marketplace view, which retrieves all active listings via a Django ORM query (Listing.objects.filter(is_active=True)). Keyword search is implemented through Django ORM Q-objects that perform case-insensitive substring matching across the title and description fields. Category-based filtering chains additional .filter() clauses to narrow results by product type. Price-range filtering applies __gte and __lte field lookups. Results are ordered by recency (created_at descending) and rendered through the marketplace template.

4) Buyer–Seller Interaction:
A prospective buyer clicking on a listing accesses the product detail view, where full listing information is displayed. To initiate communication, the buyer submits a message through the in-platform messaging interface. The messaging module retrieves or creates a Conversation object linking the buyer and seller to the specific listing, then persists a Message object containing the message body, sender reference, and timestamp. Both parties receive in-platform Notification records, surfaced through their respective dashboards.

5) Transaction Completion:
Once buyer and seller reach agreement, the buyer proceeds to the cart and checkout module. CartItem records accumulate selected listings; an Order object is created summarizing total amount and delivery details; and a Transaction record logs the payment method (cash on delivery, UPI, or Razorpay gateway) and payment status. Upon order confirmation, the seller may mark the listing as sold by setting is_active=False and is_sold=True, removing it from active marketplace listings.

VII. IMPLEMENTATION

A. Technology Stack
Table I presents the complete software stack employed in the implementation of CampusXChange.

TABLE I SOFTWARE TECHNOLOGY STACK

Component	Specification / Technology
Language	Python 3.10+
Backend	Django 4.x (MVT Architecture)
Frontend	HTML5, CSS3, JavaScript, Bootstrap 5

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

Database	SQLite (Development) / PostgreSQL (Production)
ORM	Django ORM (Object-Relational Mapping)
Template Engine	Jinja2 (Django Templates)
IDE	VS Code / PyCharm

B. Custom User Model

The default Django user model is extended through the CustomUser class, which inherits from AbstractUser and adds campus-specific fields: role (student or faculty), college (drawn from a curated list of Mumbai University-affiliated institutions), course, academic year, bio, phone number, profile picture (uploaded to profile_pics/), and an is_verified Boolean flag that may be toggled by administrators. The average_rating property dynamically computes the mean of all Review records associated with the user, providing a real-time seller reputation indicator.

C. Authentication Flow

The register_view function validates the submitted RegistrationForm and, on success, persists the new user and calls login(request, user) to establish an authenticated session. The login_view authenticates credentials using Django's authenticate() and login() functions, supporting redirection to the originally requested URL via the next query parameter. The @login_required decorator is applied to all views that require authentication, automatically redirecting unauthenticated requests to the login page.

D. ORM and Database Design

All data access is performed through Django's ORM layer. The Listing model stores product data and maintains a ForeignKey to CustomUser (seller). The Conversation model captures a ManyToManyField of participants and a ForeignKey to the associated Listing, while the Message model records individual messages with sender and timestamp metadata. The Order and Transaction models capture checkout state and payment status respectively. Django migrations manage schema evolution without direct SQL intervention, ensuring database portability between SQLite and PostgreSQL environments.

E. Administrative Module

The Django admin interface is customized with ModelAdmin subclasses for Order, Transaction, Cart, and Listing. The OrderAdmin class registers custom actions (mark_confirmed, mark_delivered, mark_cancelled) that perform bulk status updates via queryset.update(), enabling platform administrators to manage orders efficiently. TransactionAdmin exposes payment status and method filters, while the ListingAdmin provides category and condition filters alongside an is_active toggle.

VIII. RESULTS AND DISCUSSION

The CampusXChange platform was subjected to a series of functional and usability tests conducted across multiple user accounts on desktop (Google Chrome, Mozilla Firefox) and mobile browser environments. Table II summarizes the test cases and their outcomes.

TABLE II FUNCTIONAL TEST RESULTS

Test Case	Expected Result	Status
User Registration & Login	Credentials validated, session created	PASS
Product Listing Creation	Item stored in DB, visible in marketplace	PASS
Keyword Search & Category Filter	Accurate, relevant results returned	PASS
In-platform Messaging	Messages delivered between buyer and seller	PASS
Cart & Checkout Flow	Order created, status updated in DB	PASS
Responsive Design (Desktop & Mobile)	Bootstrap layout adapts correctly	PASS
Admin Dashboard – Listing Management	Admin can view, filter, update listings	PASS

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

The user registration and authentication module correctly validated institutional credentials, managed session state, and rejected duplicate usernames. Product listing creation, editing, and deletion functions performed as expected across all test accounts, with uploaded images stored and retrieved correctly from the configured media storage path.

Keyword search returned accurate results through ORM Q-object queries across title and description fields, while category and price-range filters correctly narrowed the result set. The Bootstrap 5 responsive layout rendered correctly across viewport widths from 375px (mobile) to 1920px (desktop), with no observed layout breakage.

The in-platform messaging module reliably delivered messages between buyer and seller accounts in test scenarios, with Notification records generated for each incoming message. The cart and checkout flow correctly accumulated CartItem records, computed order totals, and updated transaction status upon completion.

The administrative dashboard provided full visibility into listings, orders, transactions, and user accounts, with bulk status-update actions functioning correctly across selected querysets.

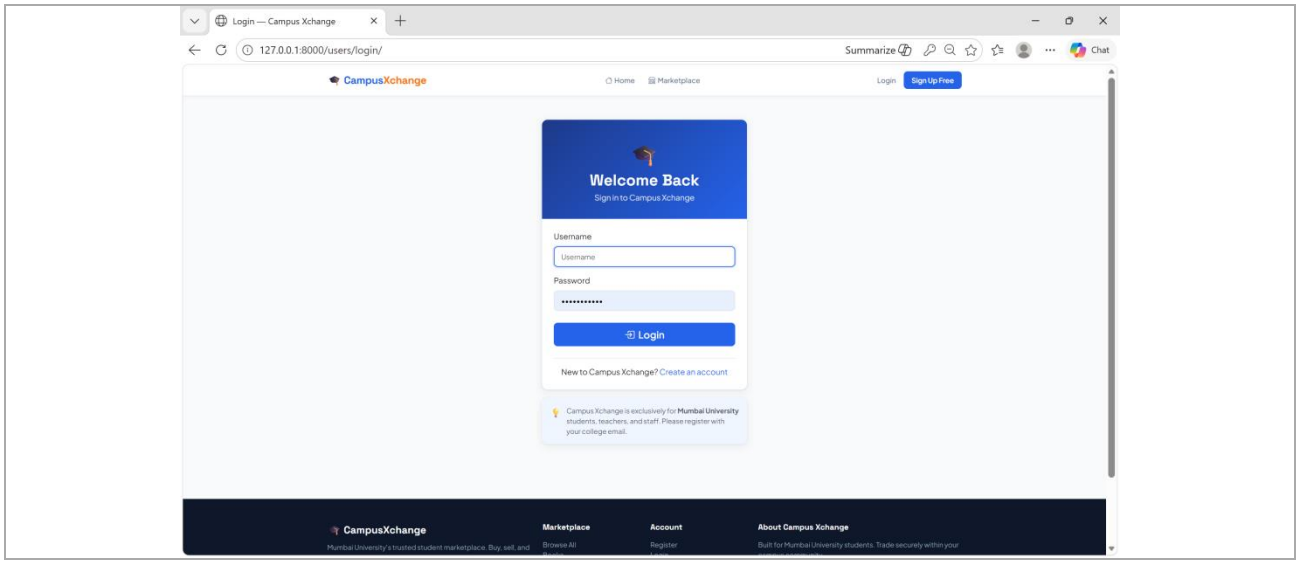

Fig. 2 User Login Interface of the CampusXChange Platform

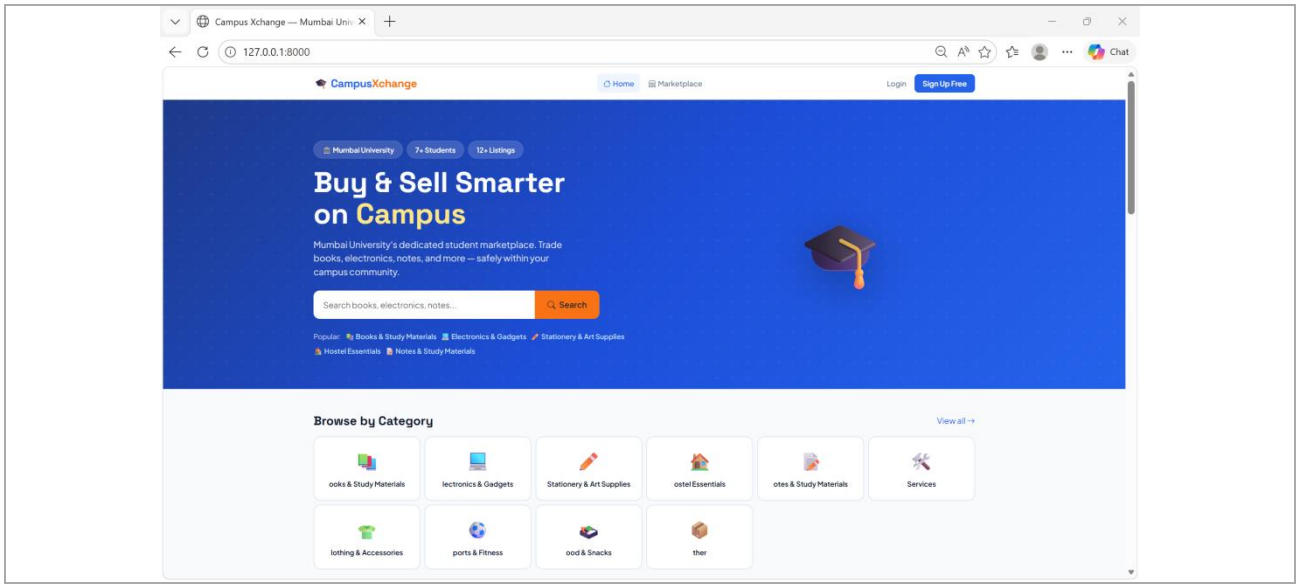

Fig. 3 Homepage Showing Category Browse and Latest Listings

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

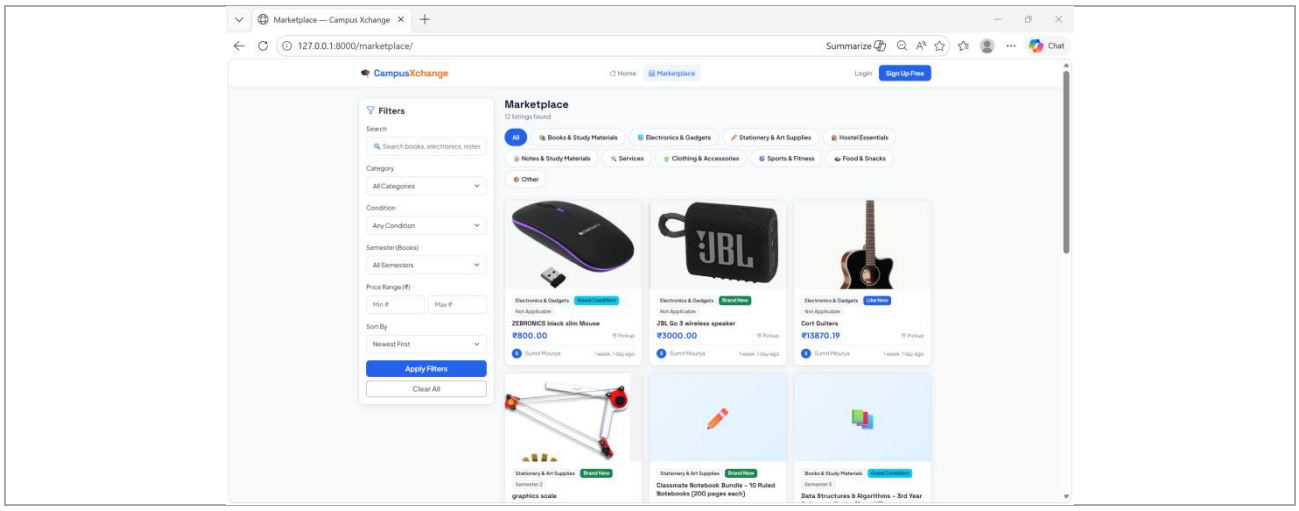

Fig. 4 Marketplace View with Search Filters and Product Cards

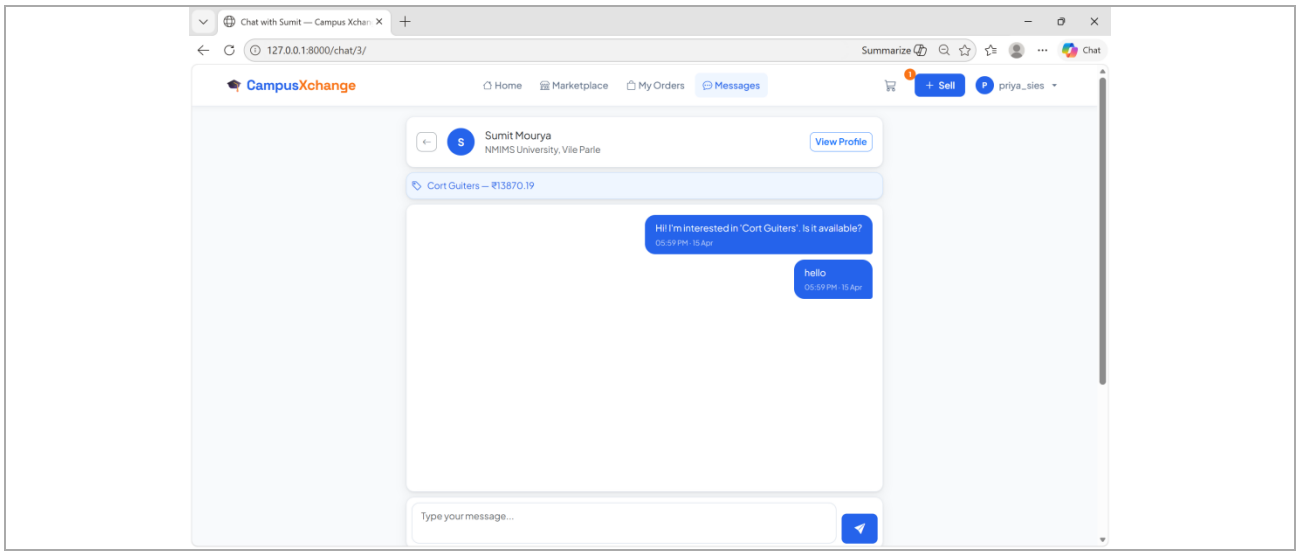

Fig. 5 In-Platform Messaging Interface between Buyer and Seller

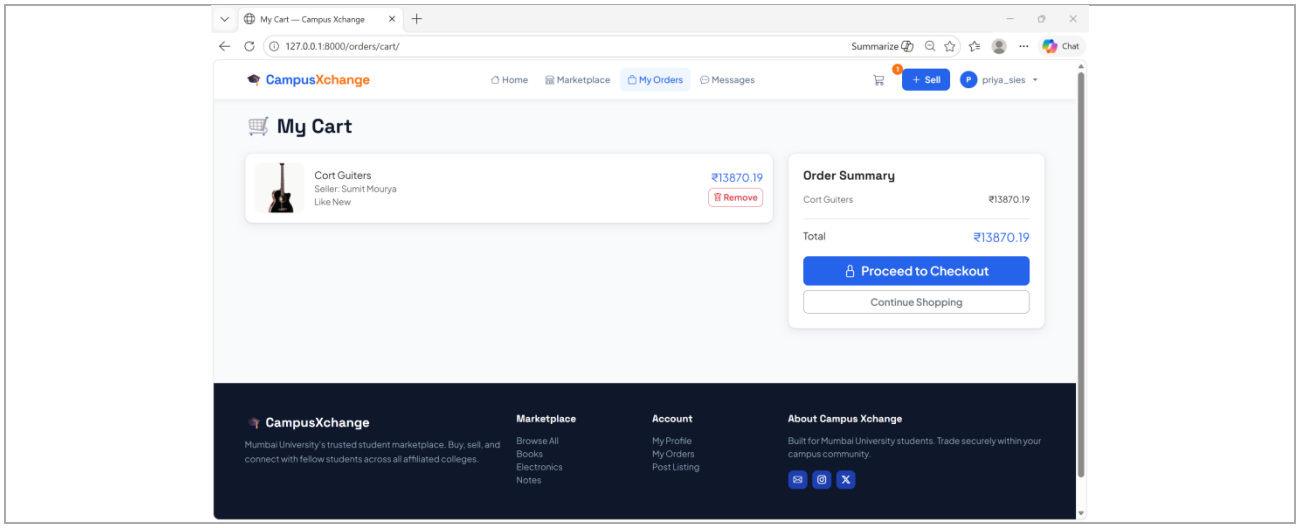

Fig. 6 Cart and Checkout Page with Order Summary and Payment Options

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154245

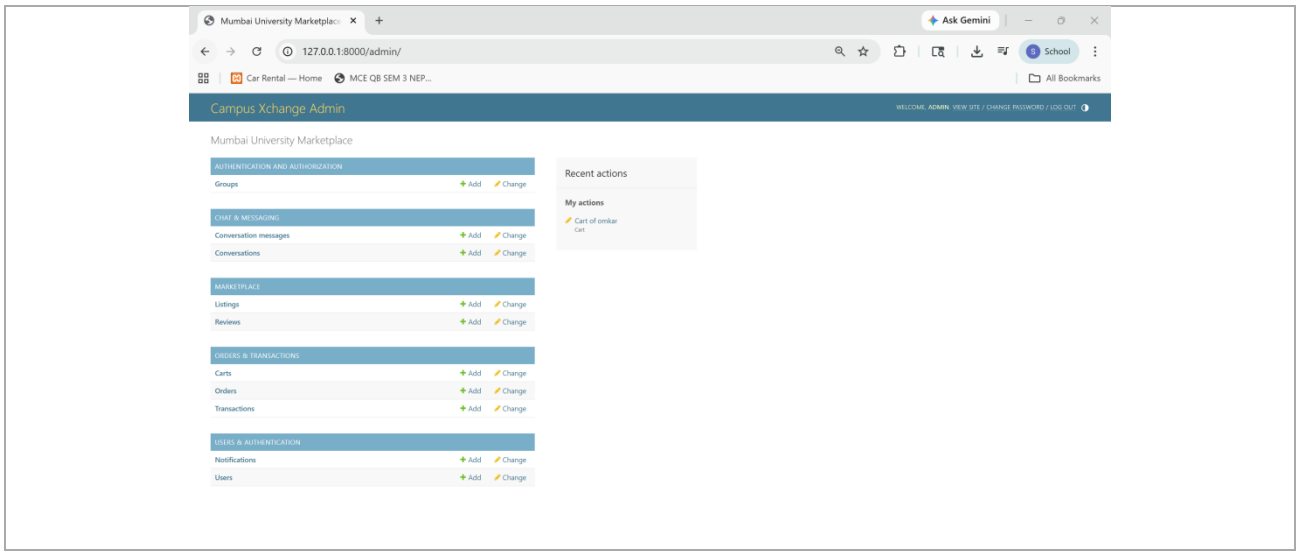

Fig. 7 Admin Dashboard Showing Listings and Transaction Management

IX. CONCLUSION

This paper has presented CampusXChange, a Django-based smart student marketplace platform that addresses the absence of a trusted, campus-specific digital commerce environment for college students. The system successfully implements student-only authentication, category-tagged product listing, keyword and filter-based discovery, in-platform buyer–seller messaging, a cart and checkout workflow, and a comprehensive administrative module—all within a responsive, mobile-compatible interface.

Functional testing confirms that all primary modules operate correctly across target environments. The platform promotes affordability by enabling peer-to-peer resale of academic materials, contributes to environmental sustainability through reuse of goods, and fosters community engagement within the campus ecosystem.

Future work will extend the platform through: (i) integration of a secure payment gateway such as Razorpay or Paytm for cashless transactions; (ii) development of dedicated Android and iOS mobile applications using React Native or Flutter; (iii) automated email and SMS notification services for listing updates and message alerts; (iv) an AI-driven recommendation engine leveraging collaborative filtering on user browsing and transaction history; and (v) expansion of the verified user base to encompass multiple Mumbai University-affiliated institutions, establishing CampusXChange as a university-wide marketplace network.

REFERENCES

[1]. D. Chaffey and F. Ellis-Chadwick, Digital Marketing: Strategy, Implementation and Practice, 7th ed. Pearson Education, 2019.
[2]. A. Holovaty and J. Kaplan-Moss, The Definitive Guide to Django: Web Development Done Right, 2nd ed. Apress, 2009.
[3]. W. S. Vincent, Django for Beginners: Build Websites with Python and Django, 4th ed. LearnDjango.com, 2022.
[4]. R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, Univ. of California, Irvine, 2000.
[5]. P. Deitel and H. Deitel, Internet & World Wide Web: How to Program, 5th ed. Prentice Hall, 2011.
[6]. Django Software Foundation, "Django Documentation," [Online]. Available: https://docs.djangoproject.com/. [Accessed: March 2025].
[7]. Bootstrap Team, "Bootstrap 5 Documentation," [Online]. Available: https://getbootstrap.com/docs/5.0/. [Accessed: March 2025].
[8]. Mozilla Developer Network, "HTML, CSS, JavaScript Documentation," [Online]. Available: https://developer.mozilla.org/. [Accessed: March 2025].
[9]. OLX India, "OLX – Buy & Sell Used Goods," [Online]. Available: https://www.olx.in/. [Accessed: February 2025].

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154245

[10]. Facebook Inc., "Facebook Marketplace," [Online]. Available: https://www.facebook.com/marketplace/. [Accessed: February 2025].
[11]. Python Software Foundation, "Python 3.10 Documentation," [Online]. Available: https://docs.python.org/3/. [Accessed: March 2025].
[12]. SQLite Consortium, "SQLite Documentation," [Online]. Available: https://www.sqlite.org/docs.html. [Accessed: March 2025].