



International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 3, May - June 2026

Impact Factor: 9.971



E-Commerce Website using Django

K. Swapna¹, P.Mounika², M.Sandhya³, K.Shalini⁴

Assistant Professor, Department of CSE, Guru Nanak Institute of Technology, Hyderabad, India¹

Student, Department of CSE, Guru Nanak Institute of Technology, Hyderabad, India^{2,3,4}

ABSTRACT: This project aims to develop a Django-based e-commerce website that provides a seamless online shopping experience. The system will include user authentication, product listings, a shopping cart, and an order management system. Users can browse products, add them to the cart, and complete purchases, while administrators can manage products, categories, and orders efficiently. The website will be built using Django (Python) for the backend, along with HTML, CSS, and JavaScript for the frontend. Django's built-in ORM will handle database interactions, ensuring efficient product and user data management. Payment integration can be added using third-party gateways for secure transactions. This system will offer a user-friendly, scalable, and secure e-commerce platform. Future enhancements may include product recommendations, customer reviews, real-time order tracking, and AI-driven insights to improve user engagement and sales.

I. INTRODUCTION

In the digital era, e-commerce platforms have transformed shopping by providing convenience, accessibility, and a wide range of products online. This project focuses on developing a Django-based e-commerce website that offers a secure, efficient, and user-friendly shopping experience for customers and administrators. Using Django's robust backend features such as ORM, authentication, and role-based access control, the system ensures secure data management and smooth handling of products, orders, and users. The frontend, built with HTML, CSS, and JavaScript, provides a responsive and interactive interface. The platform supports key functionalities including product browsing, category filtering, shopping cart management, order placement, and an admin dashboard for inventory and sales management. The system is also designed to be scalable, allowing future enhancements such as payment gateway integration, real-time order tracking, wishlists, customer reviews, and personalized product recommendations.

II. LITERATURE REVIEW

L. Das and A. Jain (2022) discussed shopping cart system design and workflows in e-commerce applications. The study aims to analyze how shopping carts manage user interactions such as adding, updating, and removing products. The paper explains the role of sessions, cookies, and relational databases in maintaining cart states and emphasizes Django's efficient session management capabilities for handling shopping cart data in e-commerce systems.

P. Nagaraj, V. Muneeswaran, AVSR Pavan Naidu, N. Shanmukh, P. Vinod Kumar, and G. Sri Satyanarayana (2023) discussed the development of an automated e-commerce price comparison website using PHP, XAMPP, MongoDB, Django, and web scraping techniques. The study aims to help users compare product prices, ratings, and availability across multiple e-commerce platforms efficiently. The proposed system uses web scraping for automatic data collection and MongoDB for efficient data storage and retrieval. The results showed that the system improves online shopping experiences by reducing manual effort and supporting better purchasing decisions.

Marian Ileana, Maria Ioana Oproiu, and Constantin Viorel Marian (2024) discussed e-commerce solutions using distributed web systems with microservices-based architecture for high-performance online stores. The study aims to improve scalability, reliability, and flexibility in e-commerce systems by dividing applications into independent microservices for functions such as user management, payment processing, and order handling. The research highlights that microservices architecture reduces system complexity, improves fault tolerance, and supports high traffic with faster response

III. METHODOLOGY

The Django-based e-commerce website is developed using multiple integrated modules to provide a secure, efficient, and user-friendly online shopping experience. The system includes a User Authentication Module that manages user registration, login, logout, password reset, and role-based access control using Django's built-in authentication and

session management features. A Product Management Module allows administrators to add, update, delete, and categorize products with details such as price, description, images, and stock quantity for effective inventory management. The Shopping Cart Module enables users to add products, update quantities, and remove items while maintaining cart information through sessions or user accounts. The Order Management Module handles order processing, shipping details, cost calculation, order confirmation, and order status tracking to ensure smooth purchase management. The Admin Dashboard Module provides a centralized interface for managing products, users, categories, orders, and sales records efficiently. Additionally, the User Profile Module allows customers to manage personal details, addresses, passwords, and order history, thereby enhancing personalization and improving the overall shopping experience.

Disadvantages of existing system:

- Flask provides only minimal core functionalities, so additional libraries and extensions are required for advanced features.
- Developers need to manually configure many components, which increases coding effort and development time.
- Flask does not include a built-in Object Relational Mapper (ORM), requiring external tools such as SQL Alchemy for database management.
- Requires more manual coding, which increases development time.
- Not suitable for large-scale applications like e-commerce websites.
- Security features are limited, so developers must implement them manually.
- Managing database and integrations becomes complex without built-in support

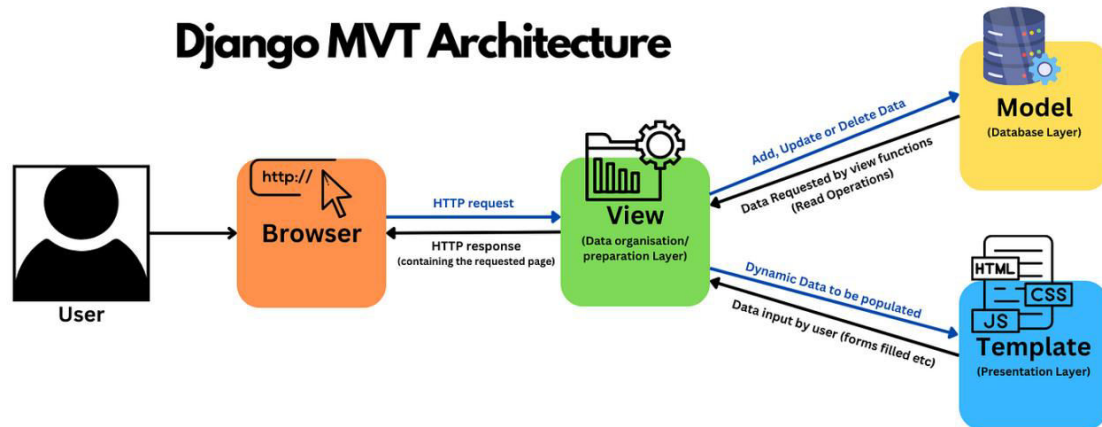
IV. PROPOSED SYSTEM

The proposed system is a Django-based portfolio website that includes features such as project showcasing, skill representation, and a contact form. Although Django provides a well-structured framework with built-in database support, it can introduce unnecessary complexity for a simple portfolio website. To overcome these limitations, the proposed system migrates the portfolio to Flask while retaining core functionalities like the Projects Section, Skills Showcase, and Contact Form without depending on a database. Developed using Flask, HTML, CSS, and JavaScript, the new system offers a lightweight, responsive, and efficient user experience. This migration improves customization, simplifies maintenance, and enhances overall performance. Additionally, future enhancements such as blog integration, real-time analytics, and an admin panel can further improve the platform, making it more professional and effective for self-branding and networking.

Proposed System Advantages

- **Scalability:** The system can efficiently handle increasing users, data, and functionalities without affecting performance.
- **Built-in Features:** It offers built-in functionalities such as authentication, database management, and security features, reducing additional development effort.
- **Rapid Development:** The framework provides reusable components and tools that help in faster application development and deployment.
- Secure authentication and data management.
- User-friendly interface for better experience.
- Efficient database handling using **ORM**.
- Easy to integrate payment gateways and APIs.

V. SYSTEM ARCHITECTURE



The Django MVT (Model–View–Template) Architecture is a framework design pattern used to organize web applications. The **User** sends a request through the browser, which is handled by the **View**. The View processes the request, interacts with the **Model** to retrieve or update data in the database, and sends the required data to the **Template**. The Template uses HTML, CSS, and JavaScript to display the webpage to the user. Finally, the processed response is sent back to the browser. This architecture separates database operations, business logic, and presentation, making applications more organized, scalable, and easy to maintain.

MODULES:**1. User Authentication Module**

This module manages secure user registration, login, password reset, and role-based access using Django's authentication and session management features.

2. Product Management Module

This module allows administrators to add, update, delete, and categorize products with details such as price, image, description, and stock quantity.

3. Shopping Cart Module

This module enables users to add, update, and remove products from the cart while maintaining cart data through sessions or user accounts.

4. Order Management Module

This module processes orders, manages shipping details, calculates total costs, confirms purchases, and tracks order status.

5. Admin Dashboard Module

This module provides centralized control for managing products, users, orders, categories, sales records, and stock details efficiently.

6. User Profile Module

This module allows users to manage personal information, addresses, passwords, and order history, improving personalization and user experience.

VI. IMPLEMENTATION

The proposed e-commerce website was developed using the Django framework with Python as the backend technology and HTML, CSS, and JavaScript for the frontend interface. The system follows the Model–View–Template (MVT) architecture to ensure modularity, scalability, and efficient data management. Key functionalities such as user authentication, product management, shopping cart operations, order processing, and admin control were successfully integrated to provide a secure and seamless online shopping experience. Django's built-in ORM and authentication mechanisms were utilized to simplify database operations and enhance application security. The implemented system delivers reliable performance, easy maintainability, and flexibility for future enhancements such as payment gateway integration, product recommendations, and real-time order tracking.

VII. EXPERIMENTAL RESULTS

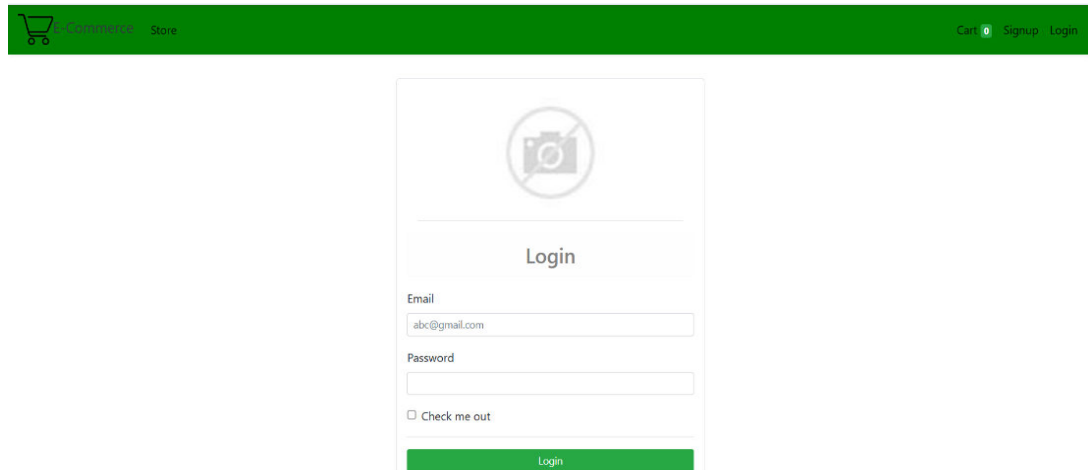


Figure. Login Page

This image shows the login page of the website.

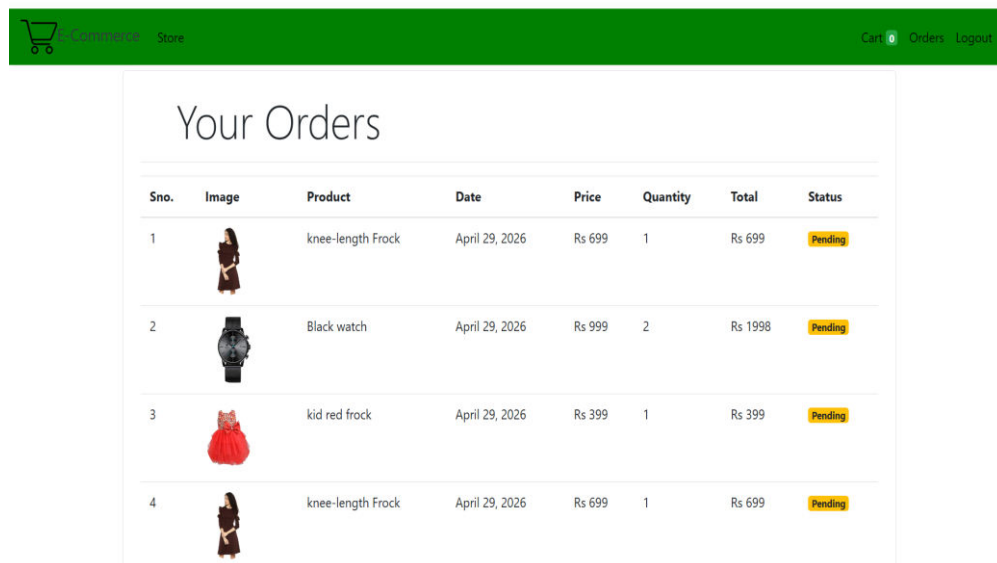


Figure. Orders Page

This is the orders page of the e-commerce website.

VI. CONCLUSION

In conclusion, this project successfully demonstrates the development of a robust and user-friendly Django-based e-commerce website. The system offers essential functionalities such as user authentication, product browsing, shopping cart management, order processing, and admin control, all built with a scalable and secure architecture. Leveraging Django's built-in features and the flexibility of frontend technologies like HTML, CSS, and JavaScript, the platform ensures smooth performance and a seamless shopping experience. The inclusion of a payment integration system further adds real-world value to the application. With future enhancements, the platform holds the potential to evolve into a highly intelligent and dynamic e-commerce solution tailored to modern consumer expectations and business needs.

VII. FUTURE ENHANCEMENT

To further enrich the user experience and boost the efficiency of the e-commerce platform, several feature enhancements can be implemented in future iterations. One of the key additions could be a product recommendation system using machine learning algorithms that suggest items based on user preferences and browsing history. Customer reviews and ratings can also be integrated to build trust and transparency among buyers. Implementing real-time order tracking would improve user engagement and provide better visibility of shipping status. Additionally, integrating multiple secure payment options, including digital wallets and UPI, can expand payment flexibility. A mobile-responsive design or dedicated mobile application can further improve accessibility. From an admin perspective, sales analytics dashboards and inventory forecasting tools powered by AI could help optimize business decisions and streamline operations.

REFERENCES

- [1] GeeksforGeeks, "E-commerce Website using Django," GeeksforGeeks, Mar. 24, 2025. [Online]. Available: <https://www.geeksforgeeks.org/e-commerce-website-using-django/>
- [2] Shyam999, "Django-ecommerce," GitHub, Jun. 20, 2024. [Online]. Available: <https://github.com/shyam999/Django-ecommerce>
- [3] Hackstarsj, "Django-ecommerce-project-amazon-clone," GitHub, Jun. 26, 2024. [Online]. Available: <https://github.com/hackstarsj/django-ecommerce-project-amazon-clone>
- [4] Fl4viooliveira, "django_ecommerce," GitHub, Sep. 8, 2023. [Online]. Available: https://github.com/fl4viooliveira/django_ecommerce
- [5] YashMarmat, "FullStack_Ecommerce_App," GitHub, Dec. 13, 2022. [Online]. Available: https://github.com/YashMarmat/FullStack_Ecommerce_App
- [6] JustDjango, "django-ecommerce," GitHub, 2025. [Online]. Available: <https://github.com/justdjango/django-ecommerce>
- [7] Atulguptag, "Django-eCommerce-Website," GitHub, 2025. [Online]. Available: <https://github.com/atulguptag/Django-eCommerce-Website>
- [8] Zinmyoswe, "Django-Ecommerce," GitHub, 2025. [Online]. Available: <https://github.com/zinmyoswe/Django-Ecommerce>



International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 9.971