STM JOURNALS

Recent Trends in
Programming Languages

ISSN: 2455-1821
Volume 13, Issue 1, 2026
January–April
DOI (Journal): 10.37591/RTPL

https://journals.stmjournals.com/rtpl

Review RTPL

Role-Based Online Food Ordering and Delivery System Using Django and RESTful Architecture

Oleti Durga Prasad[1], Pedakapu Surya Prasad[1], Ravipati Raja[1], Madasu Hemanth[1], Y. Satya Vinod[2,*]

Abstract

Traditional food ordering processes in restaurants rely heavily on manual interactions, including in-person ordering, phone-based bookings, and unstructured coordination between customers, restaurants, and delivery personnel. These approaches lead to inefficiencies such as delayed order processing, incorrect order handling, lack of real-time tracking, and poor coordination among stakeholders. Although modern applications exist, many academic implementations lack modular architecture, role-based access control, and scalable backend design. This paper presents the design and implementation of a role-based online food ordering and delivery system developed using Django and RESTful architectural principles. The system follows a modular architecture separating user roles, including admin, customer, restaurant, and delivery personnel. Each module is designed with clear responsibility boundaries, improving maintainability and scalability. Core functionalities include food browsing, cart management, order placement, restaurant-side order handling, and delivery tracking. The system ensures data consistency through centralized database management and structured workflows. Role-based access control ensures secure and restricted operations across modules. The implementation demonstrates how modern web frameworks like Django can be used to build scalable, role-driven systems with clear separation of concerns. The system serves as a foundation for further enhancements toward real-world deployment, including payment integration and real-time tracking.

Keywords: Business rule validation, REST API, role-based access control, Spring Boot, transactional integrity

INTRODUCTION

Food service management systems play a crucial role in modern urban lifestyles by enabling efficient coordination among customers, restaurants, and delivery personnel. Within this domain, online food ordering represents a fundamental yet operationally complex process that requires seamless interaction across multiple stakeholders, including customers, restaurant staff, administrators, and delivery agents [1]. Traditional food ordering approaches, primarily involving physical visits or telephone-based requests, introduce procedural inefficiencies, such as delayed order processing, miscommunication, manual billing errors, and a lack of real-time tracking [2].

Conventional workflows require customers to visit restaurants or place orders via phone calls, after which the restaurant staff manually records the orders and coordinates the preparation and delivery. This process often leads to extended waiting times, incorrect order handling, and the absence of centralized order tracking mechanisms [3].

***Author for Correspondence**
Y. Satya Vinod
E-mail: satyavinod55@gmail.com

[1]Student, Department of Electronics and Communication Engineering, Bonam Venkata Chalamayya Engineering College (affiliated to JNTU Kakinada), Andhra Pradesh, India
[2]Assistant Professor, Department of Electronics and Communication Engineering, Bonam Venkata Chalamayya Engineering College (affiliated to JNTU Kakinada), Andhra Pradesh, India

Received Date: March 31, 2026
Accepted Date: April 02, 2026
Published Date: April 30, 2026

Citation: Oleti Durga Prasad, Pedakapu Surya Prasad, Ravipati Raja, Madasu Hemanth, Y. Satya Vinod. Role-Based Online Food Ordering and Delivery System Using Django and RESTful Architecture. Recent Trends in Programming Languages. 2026; 13(1): 1–11p.

Additionally, communication gaps between restaurants and delivery personnel increase operational delays and reduce the efficiency of the service. The lack of integrated systems also limits transparency for customers regarding order status and estimated delivery times.

Academic research has explored digital food ordering solutions, ranging from basic web-based platforms to mobile applications and enterprise-level delivery systems [4]. However, existing implementations have several architectural and functional limitations. Many systems lack a proper modular design, resulting in tightly coupled components that reduce scalability and maintainability. Role-based access control is often weak or absent, allowing unauthorized access to critical operations to occur. Furthermore, database design and workflow coordination are frequently insufficient, leading to inconsistent data handling and inefficient order lifecycle management [5].

This paper presents an online food ordering and delivery system developed as an academic prototype to demonstrate modern web application design principles and role-based workflow management. The system was implemented using the Django framework, following a structured client–server architecture that separates the presentation, application logic, and data persistence layers. The primary focus is on order lifecycle management, from food browsing and cart operations to order placement, restaurant processing, and delivery coordination, with supporting modules for user management, food item administration, and system control.

The implementation contributes to the academic understanding of full-stack web application development in several ways. First, it demonstrates a modular system design with a clear separation of responsibilities across multiple user roles, including admin, customer, restaurant, and delivery personnel. Second, the system enforces structured workflows for order processing, ensuring logical transitions between the different stages of the order lifecycle. Third, centralized database management ensures the consistency and integrity of the data across all modules. Fourth, the system illustrates the practical implementation of web technologies, including Django, HTML, CSS, and database integration, for building scalable applications.

The remainder of this paper is organized as follows. Section II reviews related work and identifies existing gaps in food ordering systems. Section III presents the system architecture and technology stack. Section IV details the design and implementation of the core modules, with an emphasis on workflow management and system integration. Section V presents functional validation and performance analysis. Section VI discusses the limitations and potential improvements. Finally, Section VII concludes the paper.

LITERATURE REVIEW

Online food ordering systems have been widely studied as representative applications of workflow automation and e-commerce. This section reviews existing academic implementations and identifies recurring architectural and functional limitations relevant to food ordering and delivery systems.

Web-Based Implementations

Early academic efforts focused on replacing traditional restaurant ordering processes with web-based interfaces and database-backed data storage. Adamu [1] developed a PHP-based food ordering system incorporating menu browsing and order placement functionalities, demonstrating improved efficiency compared with manual ordering methods. However, the procedural design lacked proper modularization and service layer abstraction, limiting system scalability and maintainability. Alade et al. [3] proposed a PHP–MySQL system for small-scale restaurant operations, successfully digitizing order workflows but exhibiting weak validation mechanisms and limited coordination between order processing and delivery modules.

These studies highlight a common trend in early implementations: a strong emphasis on user interface design and basic create, read, update, delete (CRUD) operations, with minimal attention to architectural layering, role-based access control, or workflow management.

Mobile and Platform-Specific Systems

Some studies have explored mobile-based solutions for food ordering. Development of an Android-based application that enables users to browse menus and place orders directly from mobile devices, thereby improving accessibility and user convenience. Although effective within its scope, the platform-specific design restricted accessibility across different devices and limited integration with web-based administrative systems.

Similar limitations have been reported by Sapona et al. [6]. Lack of cross-platform compatibility constrained system scalability and interoperability with other services, such as delivery tracking and payment systems.

Framework-Based Implementations

More recent studies have adopted modern web development frameworks to build scalable food ordering platforms. Birje et al. [7] implemented a food ordering system using the MERN stack, demonstrating responsive user interfaces and REST API-based communications. However, the study identified limitations in handling complex workflows, particularly in coordinating order processing and delivery tracking across multiple modules. In addition, data consistency issues arise owing to insufficient transaction handling in concurrent operations [8].

Other studies have explored intelligent automation in food services. An AI-driven food recommendation and ordering system incorporating user preferences and predictive analytics. Although innovative, such systems require large datasets and computational resources, making them less suitable for small- and medium-scale implementations. Furthermore, the integration of recommendation systems and core order processing workflows has not been fully addressed [9].

Identified Gaps

An analysis of the existing literature reveals several recurring gaps. First, many systems lack a clear modular architecture with tightly coupled components that reduce flexibility and maintainability. Second, role-based access control is often insufficient, leading to security vulnerabilities and improper access to system functions. Third, workflow coordination among customers, restaurants, and delivery personnel is frequently incomplete, resulting in inefficient order lifecycle management [10].

Additionally, real-time order tracking and synchronization between modules are often absent or poorly implemented. Database design in many systems does not adequately support the complex relationships between users, orders, and delivery entities. The inconsistent handling of order states, such as placement, processing, and delivery, creates ambiguity and operational inefficiencies [9].

The system presented in this paper addresses these gaps through a role-based modular architecture, clear separation of system components, structured order lifecycle management, and a centralized database design that ensures data consistency across all modules.

SYSTEM ARCHITECTURE

The proposed online food ordering and delivery system adopts a three-tier client–server architecture that separates presentation, application logic, and data persistence concerns.

Three-Tier Architecture

The Presentation Tier consists of a web-based user interface developed using HTML, CSS, and Django templates, providing interfaces for food browsing, cart management, order placement, and other administrative operations. Different dashboards are provided for customers, restaurants, delivery personnel, and administrators. Communication with the backend occurs through HTTP requests

handled by Django views, ensuring a structured interaction between the frontend and backend components.

The application tier is implemented using the Django framework and encapsulates all business logic, request handling, and workflow coordination. Internally, it follows the Model-View-Template (MVT) architectural pattern. Views handle HTTP requests and responses, models define database schemas and relationships, and templates manage presentation logic. The application layer coordinates core functionalities, including order processing, user authentication, food management, and delivery assignment.

The Data Tier employs an SQLite relational database for persistent storage and manages entities such as users, customers, restaurants, food items, orders, and delivery records. The database ensures data consistency and integrity through structured relationships and constraints on the data. The overall architecture is shown in Figure 1.

System Modules and API Design

The application is organized into multiple role-based modules: admin (system control), customer (food browsing and ordering), restaurant (food and order management), and delivery (order delivery tracking). Each module interacts with the backend through structured URL routing, which is handled by Django views [11–13].

The endpoints followed resource-oriented patterns such as order creation, user registration, and food management. HTTP methods are used appropriately: POST for creating resources (e.g., placing orders), GET for retrieving data (e.g., viewing food items), and PUT/UPDATE operations for modifying order status. JavaScript object notation (JSON) and form-based data were used for communication. The organization of the system module is presented in Table 1.

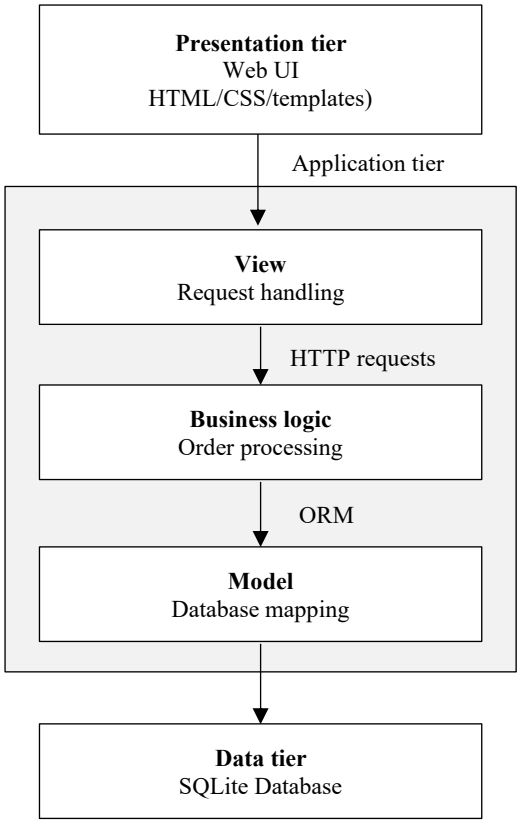

Figure 1. Three-tier architecture illustrates the separation of concerns between presentation, application logic, and persistent storage.

Table 1. System module and endpoint organization across backend components.

Module	Endpoint	Method	Purpose
Auth	/register	POST	User registration
Auth	/login	POST	User login
Customer	/foods	GET	View food items
Customer	/cart	GET	View cart
Customer	/place-order	POST	Place order
Restaurant	/add-food	POST	Add food item
Restaurant	/manage-orders	GET	View orders
Restaurant	/update-status	POST	Update order status
Delivery	/assigned-orders	GET	View assigned orders
Delivery	/deliver-order	POST	Mark as delivered
Admin	/manage-users	GET	Manage users
Admin	/manage-restaurants	GET	Manage restaurants

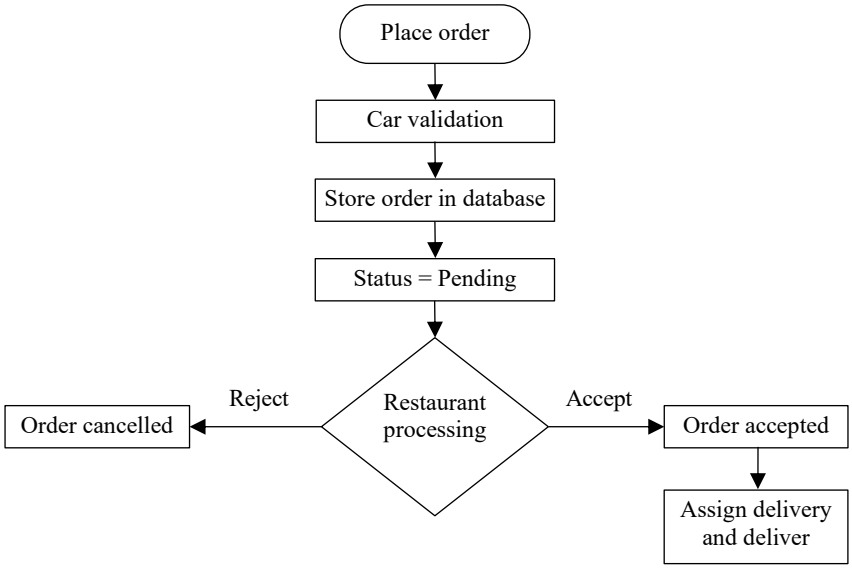

Figure 2. An entity-relationship diagram representing the database schema and inter-module dependencies.

Data Model and Entity Relationships

The data model comprises multiple entities with well-defined relationships. The user entity stores authentication credentials and role types (admin, customer, restaurant, and delivery). The customer entity extends user details with ordering capabilities. The restaurant entity manages food items and incoming orders. Food item stores menu details, including name, price, and description. Order captures order details with references to the customer and restaurant, including the order status. Delivery records track the assigned delivery personnel and delivery status.

Entity relationships are implemented using Django object-relational mapping (ORM) with foreign key associations to ensure referential integrity. One-to-many relationships exist between restaurant and food item, customer and order, and order and delivery. These relationships enable a structured data flow across the modules. The entity-relationship diagram is shown in Figure 2.

Technology Stack Selection

Django provides rapid development capabilities with built-in features such as ORM, authentication, and URL routing. SQLite is used for lightweight relational data storage that is suitable for academic

implementations. HTML, CSS, and template rendering provide the user interface functionality. The system demonstrates full-stack development principles, with an emphasis on backend architecture, modular design, and role-based workflow implementation rather than frontend complexity.

SYSTEM DESIGN AND IMPLEMENTATION

This section describes the design and implementation of the core functional modules, with an emphasis on workflow management, validation logic, and integration across system components.

Authentication and Role Management

The authentication module implements user registration and login functionalities, supporting multiple roles, including admin, customer, restaurant, and delivery personnel. User registration ensures the uniqueness of the username and email before creating user records in the database. Login authentication validates the credentials and redirects users to role-specific dashboards. Django's built-in authentication system securely manages sessions, enabling role-based access control across different modules. This approach simplifies user management while ensuring restricted access to system functionality.

Order Lifecycle and Processing

Order management represents the core functionality of the system, demonstrating the structured workflow coordination between customers, restaurants, and delivery personnel. Order placement begins when a customer selects food items and submits a request using the system. The complete life cycle is illustrated in Figure 3.

Validation Framework

The system implements validation at several levels. (1) Cart validation ensures that the selected food items exist and that the quantities are valid. (2) Price validation dynamically calculates the total cost to prevent manipulation. (3) User validation ensures that only authenticated users can place orders. (4) Restaurant availability validation ensures that orders are placed only for active restaurants.

Order Management Logic

Order processing is handled using Django views and models. When a user places an order, the system creates an order entity linked to the customer and restaurant entities. Order status transitions follow a predefined workflow: pending, accepted, preparing, out for delivery, and delivered. Each transition is triggered by specific roles, ensuring controlled workflow progression.

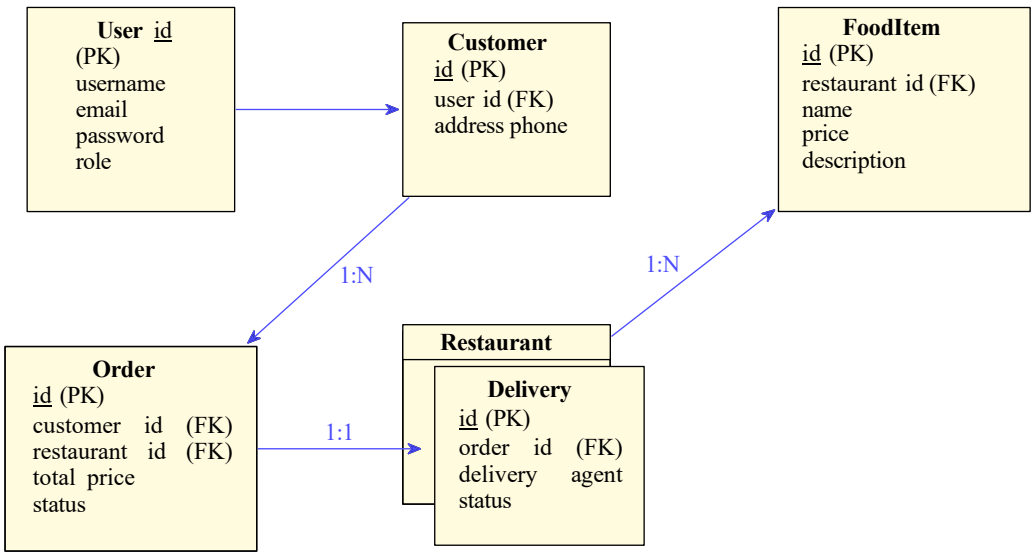

Figure 3. Lifecycle of food order processing from placement to delivery.

Data Consistency Design

A key design consideration is to ensure the consistency of order data across modules. Orders are stored centrally and updated as they progress through the different stages. This prevents duplication and ensures synchronized data between the customer, restaurant, and delivery modules. Database constraints enforce referential integrity among related entities.

Workflow Coordination

The system coordinates multiple modules during order processing. Restaurant users can view incoming orders and update their status, whereas delivery personnel receive assigned orders for delivery. Each role interacts with the same order data but performs restricted operations based on permissions, thereby ensuring controlled workflow execution.

Food and Restaurant Management

The restaurant module manages food items and order handling. Restaurants can add, update, and delete food items using dedicated interfaces. Each food item is associated with a specific restaurant, demonstrating a one-to-many relationship. The order management functionality allows restaurants to view incoming orders and update their status during preparation.

Customer and Cart Management

The customer module provides functionalities for browsing food items, adding items to the cart, and placing orders. The cart system temporarily stores selected items and calculates the total cost before checkout. This module ensures smooth interaction between the user interface and backend processing.

Delivery Management

Delivery Module

The Delivery module manages order delivery operations. Delivery personnel can view assigned orders and update the delivery status. This module acts as the final stage in the order lifecycle, ensuring the successful completion of the workflow.

Order Tracking

The system provides basic order tracking functionality, allowing customers to view the current status of their orders. Although real-time tracking is not implemented, status updates provide visibility into the order progress.

Admin Management

The admin module provides system-level control, enabling the management of users, restaurants, and overall system data. Admin users can monitor system activity and ensure the proper functioning of all modules.

Data Transfer and Exception Handling

The system uses Django models to handle data transfer between the database and application layers. Form validation ensures the correctness of the input data before processing. Exception handling mechanisms manage errors such as invalid inputs, missing data, and unauthorized access to ensure stable system behavior.

Implementation Scope and Simplifications

Several simplifications distinguish this implementation from the production systems. Payment integration has not been implemented, and order processing does not include real-time tracking or notifications. The system uses an SQLite database suitable for academic purposes but is not optimized for large-scale deployment. The security mechanisms are basic and rely on Django authentication without advanced encryption or token-based systems. These simplifications focus on demonstrating core concepts such as modular design, role-based workflows, and database integration.

RESULTS AND PERFORMANCE EVALUATION

This section presents the functional validation outcomes and analytically derived performance characteristics of the implemented online food ordering and delivery system based on architectural design and operational behavior.

System Performance Characteristics

System performance characteristics are derived from architectural analysis rather than from empirical stress benchmarking. Simple read-oriented operations, such as food item retrieval, menu browsing, and order listing, exhibit low latency because of efficient database queries and minimal processing overhead. In contrast, transactional workflows, including order placement, status updates, and delivery assignments, incur higher overheads because they involve multi-stage validation, entity state transitions, and coordinated database updates across multiple modules.

The system follows Django's built-in development server configuration and ORM-based database interaction, making it suitable for small-to medium-scale applications under typical usage conditions. The performance and scalability depend on the deployment environment, database optimization, and server configuration. Because SQLite is used as a database, it provides sufficient performance for academic use but may not scale efficiently for high-concurrency scenarios. Performance benchmarking under concurrent user loads will be considered in future work.

Functional Validation Results

Functional validation was conducted using scenario-based testing, covering typical workflows such as user registration, login, food browsing, cart operations, order placement, and order tracking, along with error conditions including invalid inputs and unauthorized access. Testing focused on the correctness of workflow execution and data consistency rather than performance benchmarking.

Multiple validation rules were verified: the correctness of user authentication, validation of cart items, accurate calculation of order totals, proper association between customers and orders, and controlled status transitions in the order lifecycle. These validations ensured reliable system behavior and data integrity across all modules.

Workflow coordination was tested by simulating interactions between the customer, restaurant, and delivery roles. The results confirmed the correct propagation of order status updates from placement to delivery completion. The system successfully maintained the consistency of order data across modules, preventing duplication and ensuring synchronized updates.

Integration testing validated the interactions between modules, ensuring that order placement triggered restaurant processing and delivery assignment correctly. Although certain advanced features, such as real-time tracking and notifications, have not been implemented, the system architecture supports future integration.

Compared with traditional manual ordering processes, the system significantly reduces the order processing time from several minutes to a few seconds. Automated order handling minimizes human error, whereas centralized data storage improves visibility and coordination among stakeholders. These improvements highlight the effectiveness of digital transformation in food service operations.

Testing was conducted manually using predefined scenarios and sample data under the single-user condition. Automated testing, multi-user concurrency evaluation, and performance benchmarking remain areas for future research. The system demonstrated functional correctness but was not optimized for production-scale deployment.

Comparative Analysis of Manual Systems

Compared with conventional manual food ordering processes, the implemented system significantly reduces operational overhead by automating menu browsing, order placement, and delivery coordination. Manual processes requiring physical presence or phone-based communication have been replaced by a centralized digital platform, enabling instant interaction between users and service providers.

Automated order management eliminates errors caused by miscommunication and manual record-keeping, whereas centralized database storage ensures consistent tracking of orders and user data. The system provides improved transparency, allowing customers to monitor order status and enabling restaurants to manage their operations efficiently. These capabilities directly address the inefficiencies and limitations observed in traditional food ordering systems.

DISCUSSION

Design Validation and Key Observations

The implementation follows a three-tier architecture with Django's MVT pattern, demonstrating a clear separation of concerns and improved maintainability. The modular structure ensures that different components, such as user management, food handling, order processing, and delivery coordination, operate independently while maintaining system integration. Django's built-in ORM simplifies database operations, enabling efficient data handling without the need for complex query management.

The key complexities involved in designing an effective order lifecycle workflow and ensuring proper coordination between multiple roles, including customers, restaurants, and delivery personnel. Managing consistent state transitions across order stages and maintaining data integrity across modules requires careful design. The use of structured models and relationships helps decouple application logic from the database representation, improving flexibility and scalability.

The current implementation has several limitations. The system uses basic authentication mechanisms without advanced security features such as token-based authentication or encryption enhancements. Payment processing is not implemented, and real-time tracking of deliveries is absent. Additionally, the system relies on manual testing rather than automated testing frameworks, which limits the validation coverage.

Despite these limitations, the system provides practical exposure to full-stack web development, role-based system design, database modeling, and workflow management. It serves as an effective academic prototype that demonstrates the key principles of modern web application development.

Limitations and Constraints

Although the system demonstrates functional correctness, it was not designed for production-level deployment. Security mechanisms are basic, relying on Django's default authentication without advanced authorization techniques such as JSON Web Tokens (JWT) or OAuth. Sensitive data handling and encryption mechanisms are minimal, which may introduce security vulnerabilities in real-world applications.

The system supports only single-application deployment and does not provide multi-tenant capability. Order processing does not include advanced features such as real-time tracking, push notifications, or dynamic delivery assignment algorithms. Additionally, scalability is limited because of the use of the SQLite database, which is not optimized for handling large-scale concurrent operations.

Performance optimization techniques such as caching, query optimization, and pagination were not implemented. These limitations highlight the need for further enhancements to transform the system into a production-ready application.

Alignment with Literature Findings

The implementation addresses several shortcomings identified in the existing literature, particularly the lack of modular architecture and inefficient workflow management in food ordering systems. The use of a structured, role-based design ensures a clear separation of responsibilities and improves system maintainability. Centralized database management and controlled order lifecycle transitions enhance data consistency and operational efficiency.

The system also demonstrated improved coordination between different modules compared to earlier implementations. Although advanced features such as intelligent recommendations and real-time tracking are not included, the foundational architecture supports the future integration of such functionalities. Overall, the system aligns with modern web development practices and addresses the key limitations identified in previous studies.

CONCLUSION AND FUTURE WORK

Conclusion

This study presents an online food ordering and delivery system prototype demonstrating modular architecture, role-based workflow management, and efficient database integration. The system validates key software engineering principles through practical implementation using the Django framework. The layered architectural approach ensures maintainability and clear separation of concerns, whereas structured order lifecycle management enables consistent coordination between customers, restaurants, and delivery personnel. The system demonstrates the effectiveness of modern web technologies in developing scalable and reliable food service applications.

Future Work

Future work will involve integrating secure payment gateways, implementing advanced authentication mechanisms such as token-based security, and enhancing automated testing coverage. Additional improvements include real-time order tracking, notification systems, and dynamic delivery assignments to improve operational efficiency. Further enhancements may include migration to scalable databases, such as MySQL or PostgreSQL, implementation of caching and pagination techniques, and the development of mobile applications to extend accessibility. The integration of AI-based recommendation systems and analytics features can further improve the user experience and system intelligence.

REFERENCES

1. Adamu A. Employee leave management system. FUDMA J Sci. 2020;4(2):86–91. doi:10.33003/fjs-2020-0402-162.
2. Shah RK. Role-based leave management system using Spring Boot and RESTful architecture. Int J Res Appl Sci Eng Technol. 2026;14:787–797. doi:10.22214/ijraset.2026.77447.
3. Samuel Mayowa A, Samuel A, Temitope John A. Design and implementation of a web based leave management system. Int J Comput Appl Technol Res. 2022;11:123–144. doi:10.7753/IJCATR1104.1006.
4. Srinithi R, Sakthi Murugan P. Employee leave management system. Int J Innov Res Electr Electron Instrum Control Eng. 2025;13(4):198–202. doi:10.17148/IJIREEICE.2025.13432.
5. Khan NA, Khan AR, Rajeyyagari S, Akhtar M, Siddiqi AMU, Ahmad M. Serverless absence management: a Google-native playbook for rapid, auditable university leave workflows. Middle East J Appl Sci Technol. 2026;9(1):23–36. doi:10.46431/mejast.2026.9103.
6. Sapona R, Thohari AH, Nelmiawati. Web-based leave management system for Politeknik Negeri Batam. In: Proceedings of the 3rd International Conference on Applied Engineering; 2020. Setúbal, Portugal: SCITEPRESS – Science and Technology Publications; 2020. p. 70–74. doi:10.5220/0010351800700074.
7. Birje RS, Benne R, Unki A. Design and development of e-leave management system. Int J Res Publ Rev. 2025;6(10):6464–6472.

8. Choudhary N, Khalfe A, Khan Y, Ansari M. Leave management system for AIKTC. Int Res J Eng Technol. 2020;7(3):1715–1717.
9. Mantri Y, Kumar DAR, Kumar SU. Emergency leave management system with company data analysis. Int J Res Eng Sci. 2023;11(21):163–174.
10. Singh M, Singh P, Singh R, Singh S, Gupta S. Leave and payroll management system. In: Proceedings of the International Conference on Computing and Virtualization (ICCCV-17); 2017. Mumbai, India: Thakur College of Engineering and Technology; 2017. p. 62–66.
11. Pathan AI, Nayak B, Nayak B, Dhatrak VB, Daivat AK. Design of AI-based leave scheduling and managing application. Int J Comput Sci Eng. 2020;8(4):97–99. doi:10.26438/ijcse/v8i4.115.
12. Jadhav SD, Ranaware AA, More PD. Design steps of online leave management application system for academic institution. In: Proceedings of the National Conference on Emerging Trends in Science and Advances in Engineering; 2023; Phaltan, India. Int J Innov Eng Res Technol. 2023.
13. Kaushik VK, Gupta AK, Kumar A, Prasad A. Student leave management system. Int J Adv Res Innov Ideas Educ. 2017;3(5):124–131.

