

**MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING**

(A Unit of Rajalaxmi Education Trust®, Mangalore)

Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

# QUICKBITE Food Delivery Application

### MINI-PROJECT

**(23MCSE527)**

**REPORT**

***Submitted by***

**SHWETHA**

**4MT23MC071**

***In partial fulfillment for the award of the degree of***

### MASTER OF COMPUTER APPLICATIONS

**2023-24**



**MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING**

(A Unit of Rajalaxmi Education Trust®, Mangalore)

Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS**

CERTIFICATE

This is to certify that **SHWETHA,** bearing **USN (4MT23MC071)** has successfully completed the second semester Mini-Project entitled **QUICKBITE Food Delivery Application** as a partial fulfillment of the requirements for the award of **MASTER OF COMPUTER APPLICATIONS** degree, during the Academic Year **2023-24**.

**Signature of the Guide Head of the Department**

Internal Examiner External Examiner

Name & Signature Name & Signature

# DECLARATION

I, **Shwetha**, student of II Semester MCA, bearing **USN 4MT23MC071** hereby declare that the project work entitled **“QUICKBITE Food Delivery Application”** has been carried out under the supervision of **Name of the Supervisor, Designation** and submitted as the partial fulfillment of the requirements for the award of the Degree of **Master of Computer of Applications**, Mangalore Institute of Technology & Engineering, an Autonomous Institution, Affiliated to Visvesvaraya Technological University during the academic year **2023-24**. This report has not been submitted to any other Organization/University for any award of degree.

Name:

Signature:

Date:

# ACKNOWLEDGEMENT

I would like to thank **Mr. Rajesh Chouta**, Chairman, Mangalore Institute of Technology & Engineering for providing a good infrastructure for timely completion of my mini-project.

I sincerely express my gratitude to **Dr. Prashanth C M,** Principal, MITE**,** for supporting me in all my technical activities and giving proper guidance when it was required. I would like to thank **Dr. Madhwaraj K G,** Professor & HoD, Department of MCA, for granting permission to undertake this project. I would like to express my gratitude to my project guide (Name of the guide with designation) for giving all the instructions and guidelines at every stage of the Project work.

I thank all the staff members of the Department of Master of Computer Applications, for extending their constant support to complete this project. I express my heartfelt thanks to my parents and friends who were also a constant source of support and inspiration throughout this project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter**  **No.** | **Title** | **Page No** |
|  | **ABSTRACT** | **(i)** |
| **1** | **INTRODUCTION** |  |
| 1.1 | Introduction |  |
| 1.2 | Problem Statement |  |
| 1.3 | Objectives of the project | **1-6** |
| 1.4 | Existing System |  |
| 1.5 | Proposed System |  |
| 1.6 | Feasibility Study |  |
| **2** | **REVIEW OF LITERATURE** | **7-8** |
| **3** | **SYSTEM CONFIGURATION** |  |
| 3.1 | Hardware requirements | **9** |
| 3.2 | Software requirements |  |
| **4** | **MODULE DESCRIPTION** |  |
| 4.1 | Modules | **10-12** |
| **5** | **SYSTEM DESIGN** |  |
| 5.1 | Data Flow Diagrams | **11-12** |
| 5.2 | Use Case Diagram |  |
| **6** | **SYSTEM IMPLEMENTATION** |  |
| 6.1 | Implementation | **13-17** |
| 6.2 | Screenshots |  |
| **7**  7.1 | **SYSTEM TESTING**  Test cases | **18-20** |
| **8**  8.1  8.2 | **RESULTS AND DISCUSSION**  Conclusions  Future Enhancements | **21** |
| **9**  9.1  9.2 | **REFERENCES**  Text references  Web references | **22** |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Figure No** | **Title** | **Page No.** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |

# ABSTRACT

The project "Online Food Delivery System Using MERN Stack" aims to build an efficient platform where users can browse through restaurant menus, place orders, and track delivery in real-time. The primary objective is to offer users a smooth and engaging experience through a responsive frontend developed using React.js. The backend, powered by Node.js and Express.js, ensures fast order management, while MongoDB stores user data and order history securely.

This platform provides four user interfaces: for customers, restaurant managers, delivery personnel, and the admin. Key features include user authentication, restaurant management, order tracking, and secure payment integration. Admins can monitor activities and perform system operations efficiently. Our goal is to deliver a reliable, scalable, and user-friendly platform with reduced delays in order processing and improved restaurant delivery management.

The system has been rigorously tested, with different test cases covering user registration, order placement, payment transactions, and order tracking to ensure quality performance. In conclusion, the online food delivery system built using the MERN stack offers a seamless digital solution for users to explore, order, and enjoy their meals at the click of a button, enhancing both restaurant efficiency and customer

**CHAPTER 1**

**INTRODUCTION**

In today’s digital world, **online food delivery platforms** are transforming the way people access food services. With the rise of smartphones and internet penetration, customers increasingly prefer the convenience of ordering food online instead of dining out. Platforms like **Swiggy, Zomato, UberEats**, and **Domino’s** have created a new marketplace, allowing restaurants to extend their services beyond dine-in facilities.

This project is an effort to **develop a scalable online food delivery system** using the **MERN stack**, combining modern JavaScript technologies for both frontend and backend. This system aims to overcome bottlenecks in food delivery, such as delays, poor order tracking, and unresponsive interfaces, by providing a platform where **customers, restaurants, delivery agents, and administrators** interact seamlessly.

The MERN stack—comprising **MongoDB, Express.js, React.js, and Node.js**—is chosen for its **scalability, performance, and ease of development**. The platform not only facilitates food ordering but also supports restaurant management, delivery tracking, and payment processing. Through modular design and clean interfaces, the system ensures smooth user interactions and fast operations, addressing common challenges in online food delivery services.

### Problem Statement

The current landscape of food delivery platforms faces several operational and user experience challenges, as described below:

1. Inconsistent Delivery Experiences

* Delays in deliveries due to poor coordination between delivery personnel and restaurants.
* Sudden surge in orders during peak hours causes disruptions, affecting service quality.
* Delivery time estimates are often inaccurate, frustrating customers.

1. User Experience Issues

* Some platforms have complicated navigation and crowded interfaces, making it difficult for users to browse menus or place orders quickly.
* Limited filtering options (e.g., dietary preferences, cuisine types) reduce the ease of selection for users.

1. Lack of Transparency

* Users find it hard to track real-time delivery statuses, leading to uncertainty and complaints.
* Restaurants are not always updated on canceled or modified orders, causing mismanagement and loss of revenue.

1. Restaurant Management Issues

* Restaurants require streamlined ways to update menus, manage orders, and analyze customer feedback.
* Existing platforms do not provide detailed analytics tools to help restaurants optimize their operations (e.g., peak-hour patterns, most ordered items).

### Objectives of the Project

This project aims to solve the issues identified above by providing a user-friendly, feature-rich, and technically scalable platform. The following objectives guide the system's design:

1. User-friendly Interface:

* The platform will use React.js to create an intuitive, responsive user interface that works on all devices (desktop, mobile, tablet).
* Provide easy-to-navigate menus, search, and filtering options to improve usability.

1. Efficient Order Management:

* Implement smooth backend workflows that streamline order creation, update statuses, and cancellations.

1. Secure Data Handling:

* Store user and order data in MongoDB with encryption to ensure data integrity and security.
* User passwords will be secured with bcrypt hashing to prevent unauthorized access.

1. Multiple User Interfaces:

* User Dashboard: For browsing menus, placing orders, and tracking deliveries.
* Restaurant Dashboard: For managing menus, processing orders, and monitoring feedback.
* Admin Dashboard: For monitoring system operations and handling restaurants, delivery staff, and customer queries.
* Delivery Staff Interface: For receiving tasks and updating order statuses.

1. Payment Integration:

* Support multiple payment modes (UPI, credit card, wallet payments) and notify users of payment confirmations instantly.

1. Scalable Architecture:

* Design the platform to handle scaling efficiently by adding or removing resources based on traffic.
* The system will incorporate load balancing and caching to ensure quick responses under high traffic conditions.

### Existing System

### Current food delivery platforms (like Swiggy, Zomato, and UberEats) offer various services but face several issues:

### High Delivery Charges During Peak Hours

### Delivery fees rise during peak times, making the service costly and deterring frequent orders from budget-conscious users.

### Unreliable Tracking Systems

### Users often experience delays or inaccurate delivery timelines, leading to frustration and reduced trust.

### Limited Order Customization

### Platforms restrict customization options, impacting users with specific dietary needs or preferences.

### Inconsistent Customer Support

### Delays and unhelpful responses during issue resolution (like refunds or cancellations) cause dissatisfaction.

### Monopoly Effects

### Some platforms give preference to specific restaurants, reducing exposure for smaller ones and limiting user choices.

### Proposed System

### The new food delivery system aims to overcome these limitations by focusing on efficient order processing, seamless user interaction, and flexible.

### Key Features of the Proposed System:

### Fast and Reliable Delivery Tracking

### Real-time order status updates (via WebSockets) ensure users receive precise delivery progress.

### Modular Architecture

### Independent modules (User, Restaurant, Admin, Delivery) enhance system performance, simplify maintenance, and enable easy upgrades.

### Restaurant and Menu Management

### Restaurants can update menus dynamically and receive instant order notifications, with built-in analytics to track best-sellers and peak order times.

### User Authentication and Profiles

### Users can create accounts to track order history and get personalized recommendations. Secure JWT-based authentication will protect sessions.

### Payment Flexibility

### Razorpay integration ensures secure transactions with instant payment confirmations through email and app notifications. The system also supports smooth refund management for order cancellations.

### Feasibility Study

### – Technical Feasibility

* The MERN stack (MongoDB, Express.js, React.js, and Node.js) offers high scalability and performance.
* The use of REST APIs ensures that different modules communicate efficiently.
* Integration with payment gateways, WebSockets, and geolocation APIs enables fast deployment and reliable operations.
* The system can easily be hosted on cloud platforms like AWS or Azure for better scalability

### - Social Feasibility

### The platform makes it easy for users to access food services from their homes without the hassle of going out.

### Restaurants benefit from increased visibility and online revenue streams, which is especially useful during events or emergencies.

### The system improves user engagement through offers, discounts, and personalized recommendations.

### - Operational Feasibility

### Intuitive dashboards for restaurants and delivery personnel ensure minimal training is required for staff.

### Admins can monitor operations effectively through the backend, managing issues in real time (e.g., refund requests, delivery delays).

### Error notifications and logs allow for proactive troubleshooting and smooth orderfulfillment.

## CHAPTER 2

**REVIEW OF LITERATURE**

The rise of online food delivery platforms, including Swiggy, Zomato, and UberEats, has transformed how users access restaurant services. This section explores prior research on technological frameworks, user experience, and operational challenges in food delivery systems, with a focus on how the QuickBite platform, developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), addresses these issues effectively.

* Technological Frameworks in Food Delivery Systems

The use of the MERN stack enhances system scalability and performance, enabling smooth interaction among customers, restaurants, and delivery staff. Real-time tracking powered by WebSockets improves transparency and customer satisfaction (Gupta et al., 2021). Integration with payment gateways like Razorpay ensures secure and instant transactions, reducing delays and boosting user trust.

* User Experience and Interface Design

A responsive and intuitive user interface is crucial for food delivery systems. React.js enables dynamic frontend development, improving accessibility across devices. Patel (2021) highlighted the importance of features like search filters and clear navigation for better user engagement. Modular dashboards for customers, restaurants, and delivery agents streamline operations, ensuring task-specific functionality for each role.

* Challenges in Existing Systems

Despite their popularity, many food delivery platforms face challenges such as inaccurate delivery timelines, peak-hour delays, and high service charges. Sharma and Kumar (2020) emphasized that ineffective order tracking and sudden order surges result in poor user experiences. QuickBite addresses these issues through efficient backend workflows using Express.js and Node.js to ensure smooth coordination between stakeholders.

* Restaurant Management and Analytics

Modern platforms also support restaurant operations by offering real-time insights into order trends and customer preferences. QuickBite provides restaurants with tools to update menus, monitor sales, and track performance, aligning with best practices suggested by Kaushik et al. (2021). This integration helps restaurants improve operational efficiency and customer satisfaction.

* Conclusion

The QuickBite platform leverages the MERN stack to deliver an enhanced user experience with real-time tracking, modular design, and secure payment integration. By addressing the limitations of existing systems, QuickBite aims to create a scalable and efficient solution for online food delivery services, benefiting both customers and restaurants.

**System Configuration**

**Hardware Requirements**

* Client: Laptop or smartphone with an internet connection
* Server: Minimum 8 GB RAM, quad-core processor

**Software Requirements**

* Frontend: React.js
* Backend: Express.js, Node.js
* Database: MongoDB
* Development Tools: Visual Studio Code, GitHub

This configuration ensures the system runs efficiently, with the React-based frontend providing a responsive user interface. The backend, powered by Express.js and Node.js, handles server-side logic, while MongoDB stores user data and order histories. Development tools such as Visual Studio Code and GitHub facilitate seamless coding and version control.

**Module Description**

The QuickBite food delivery application consists of several essential modules, each serving a distinct role to facilitate smooth operations. These modules work in synergy to provide seamless ordering, delivery tracking, payment processing, and system administration. Below is a detailed description of the core modules:

1. User Module

The User Module is designed to offer an intuitive interface for customers. It focuses on providing a smooth browsing and ordering experience. Key functionalities include:

* Browse Restaurants and Menus: Users can explore restaurants, view detailed menus, and select their preferred dishes.
* Place Orders: Customers can add items to the cart and place orders with a simple checkout process.
* Order Tracking: Real-time tracking allows users to monitor the status of their orders, from preparation to delivery.
* User Accounts: Users can register and log in to save delivery addresses, view order history, and access personalized recommendations.
* Notifications and Alerts: Users receive notifications about order status updates, promotions, and offers.

1. Admin Module

The Admin Module serves as the control center for managing the platform’s operations. It provides tools for monitoring activities, resolving issues, and ensuring smooth platform functioning. Core functionalities include:

* System Monitoring: Admins can oversee all ongoing orders, platform metrics, and user activities in real time.
* User Management: The admin can handle user accounts, including deactivating inactive users or managing complaints.
* Performance Analytics: This feature provides insights into peak usage periods, high-demand restaurants, and order trends, helping optimize platform operations.
* Issue Resolution and Refunds: Admins manage customer complaints, issue refunds, and handle disputes efficiently.
* Security Management: The module ensures data privacy and system security by managing encryption protocols and detecting suspicious activities.
* This module plays a critical role in ensuring the platform operates efficiently and securely.

1. Delivery Module

The Delivery Module is designed for delivery staff, helping them efficiently complete their tasks. It ensures smooth coordination between restaurants, customers, and delivery agents. Key features include:

* Order Assignment: Orders are automatically assigned to delivery personnel based on proximity and availability.
* Route Optimization: The system suggests the shortest delivery routes using GPS to reduce delays.
* Order Updates: Delivery staff can update the order status (e.g., “Picked Up” or “Delivered”) in real time. Customers receive notifications accordingly.
* Performance Tracking: Drivers can view their completed deliveries and performance metrics, such as customer ratings and reviews.

1. Payment Module

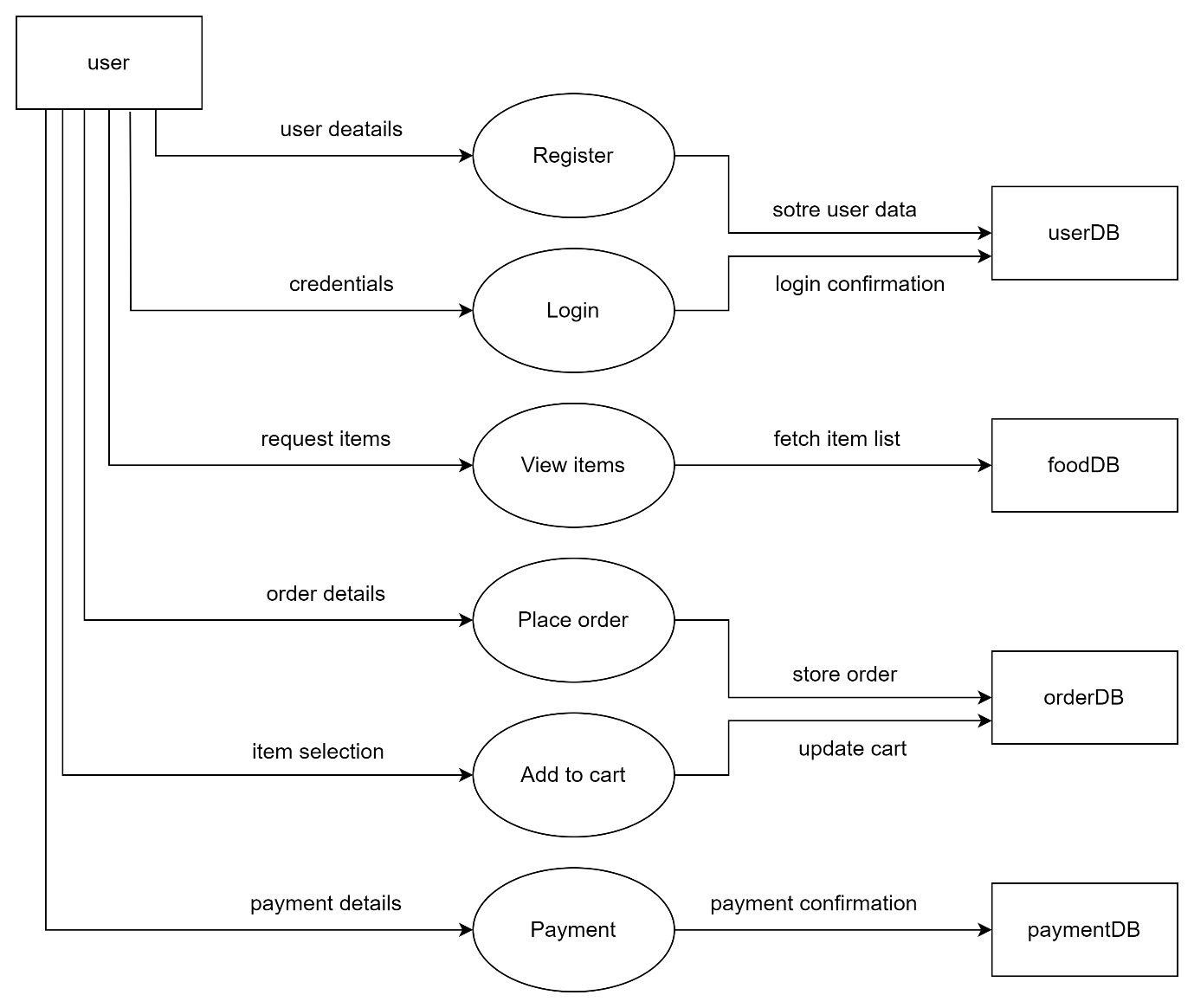
The Payment Module integrates multiple payment gateways to facilitate smooth transactions. It ensures secure payment processing for users and administrators. Key features include:

* Multiple Payment Options: Users can pay through credit/debit cards, UPI, wallets, and net banking.
* Payment Confirmation: After successful payments, users and restaurants receive confirmation notifications.
* Refund Management: The module handles refunds for canceled or failed orders, ensuring users receive their money back quickly.
* Transaction Security: All transactions are encrypted, and the system complies with security standards to prevent fraud.
* Admin Monitoring: Admins can track all transactions and resolve disputes in case of payment failures or unauthorized activities.

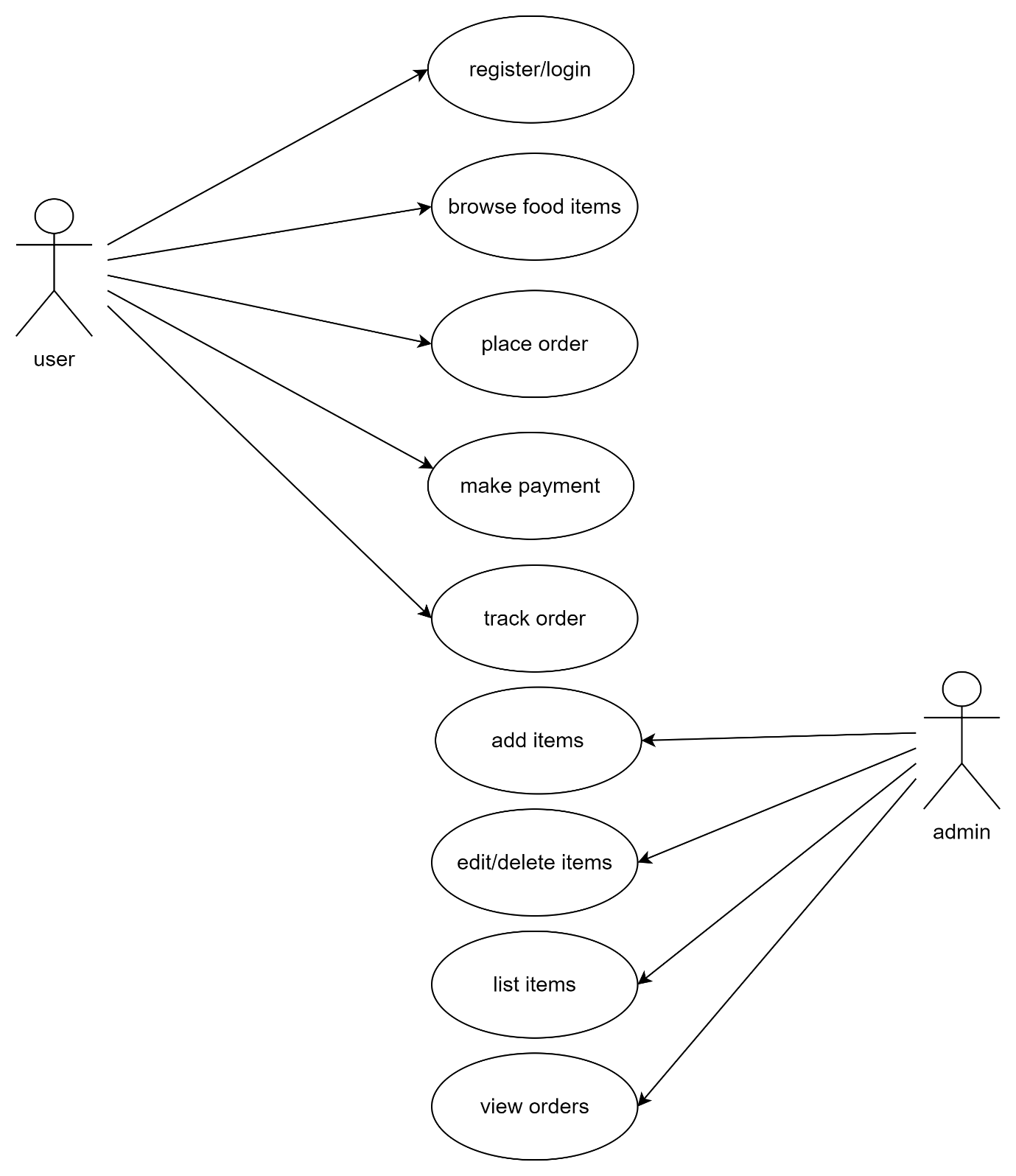
These modules collectively ensure that the QuickBite platform offers a seamless food delivery experience with efficient management, real-time tracking, secure payments, and reliable support. By focusing on user-centric features and operational efficiency, the platform ensures smooth coordination among customers, delivery staff, and administrators.

**System Design**

Data Flow Diagrams (DFD)



Use Case Diagram

****

**System Implementation**

Implementation Details

* Frontend Development: Using React components and Bootstrap for the user interface.
* Backend Development: REST API development with Express and Node.js.
* Database Setup: MongoDB schemas for storing user and order details.
* Payment Integration: Razorpay API for secure transactions.

**Screenshots**

**System Testing**

System testing ensures the QuickBite application functions as intended, offering a seamless user experience while meeting security, performance, and usability standards. The following sections summarize the objectives, testing methodologies, and key results.

1. Objectives of System Testing

* Verify Functionality: Ensure all features like registration, order placement, and payment work correctly.
* Performance Testing: Assess the system’s ability to handle multiple users simultaneously.
* Security Testing: Detect vulnerabilities and protect user data.
* Usability Testing: Confirm that the interface is easy to navigate.

1. Testing Methodologies

* Black-Box Testing: Tests outputs based on inputs without checking internal logic.
* Integration Testing: Ensures smooth interaction between modules like payments and orders.
* System Testing: Validates the complete platform functionality.
* User Acceptance Testing (UAT): Involves users to confirm the platform meets their expectations.

1. Types of Testing Conducted

3.1 Functional Testing

* Tested user registration, order placement, and tracking to ensure correct operation.

3.2 Performance Testing

* Load Testing: Simulated high traffic to check responsiveness.
* Stress Testing: Evaluated system behavior under extreme load.

3.3 Security Testing

* Conducted penetration testing to detect issues like SQL injection and XSS.
* Verified that sensitive data is encrypted and stored securely.

3.4 Usability Testing

* Ensured the interface was responsive and easy to use on all devices.

3.5 Compatibility Testing

* Confirmed the app works smoothly on different browsers and devices.

1. Key Test Cases and Results

| **Test Case ID** | **Scenario** | **Expected Result** | **Status** |
| --- | --- | --- | --- |
| TC\_01 | User Registration | User registers successfully | Pass |
| TC\_02 | Place Order | Order placed successfully | Pass |
| TC\_03 | Payment Processing | Payment completed | Pass |
| TC\_04 | Order Tracking | Real-time updates received | Pass |
| TC\_05 | Security Check (SQL) | No unauthorized access | Pass |

1. Conclusion and Recommendations

The QuickBite system passed all tests, demonstrating functionality, performance, and security. Key recommendations include:

* Regular Testing: Schedule routine tests to catch bugs early.
* User Feedback: Integrate a feedback system for reporting issues.
* Ongoing Security Audits: Monitor for emerging threats to ensure data safety.

In conclusion, the QuickBite application is ready for deployment, providing users with a reliable, secure, and user-friendly platform.

**Results and Discussion**

Results

* Highlight key achievements, such as reduced delivery time or improved user interface.

Future Enhancements

* Adding AI for personalized recommendations.
* Implementing chatbots for customer service.

**References**

* Text References: Books or papers used for research.
* Web References: Websites or APIs referred to during development.

**Level 1 dfd(user)**

**Use Case Diagram**

Actors:

* User
* Admin
* Restaurant
* Delivery Staff

Use Cases:

* Login, Browse Menu, Place Order, Manage Orders, Update Order Status.