

**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***

**SHRUTHAKEERTHIRAJ**

**4MT23MC070**

***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 **SHRUTHAKEERTHIRAJ,** bearing **USN (4MT23MC070)** 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, **Shruthakeerthiraj**, student of II Semester MCA, bearing **USN 4MT23MC070** 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 | **1-5** |
| 1.4 | Existing System |  |
| 1.5 | Proposed System |  |
| 1.6 | Feasibility Study |  |
| **2** | **REVIEW OF LITERATURE** | **6-7** |
| **3** | **SYSTEM CONFIGURATION** |  |
| 3.1 | Hardware requirements | **8** |
| 3.2 | Software requirements |  |
| **4** | **MODULE DESCRIPTION** |  |
| 4.1 | Modules | **9-10** |
| **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**

**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

**Module Description**

**Modules**

1. User Module: Users can browse restaurants and place orders.
2. Restaurant Module: Restaurants manage menus and view orders.
3. Admin Module: Admin oversees operations and adds new restaurants.
4. Delivery Module: Delivery staff can update order statuses.
5. Payment Module: Integrates with payment gateways.

**System Design**

Data Flow Diagrams (DFD)

* Level 0 DFD: Represents the entire system overview.
* Level 1 DFD: Explains detailed interactions (e.g., between users, restaurant, and delivery system).

Use Case Diagram

* Show the interactions between actors (User, Admin, Restaurant, and Delivery Staff) and the system.

**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**

Test Cases

| **Test Case ID** | **Test Scenario** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_01 | User Login | User logs in successfully | As expected | Pass |
| TC\_02 | Place Order | Order is placed | As expected | Pass |
| TC\_03 | Payment Failure | Error message on payment failure | As expected | Pass |

**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.

**Diagrams**

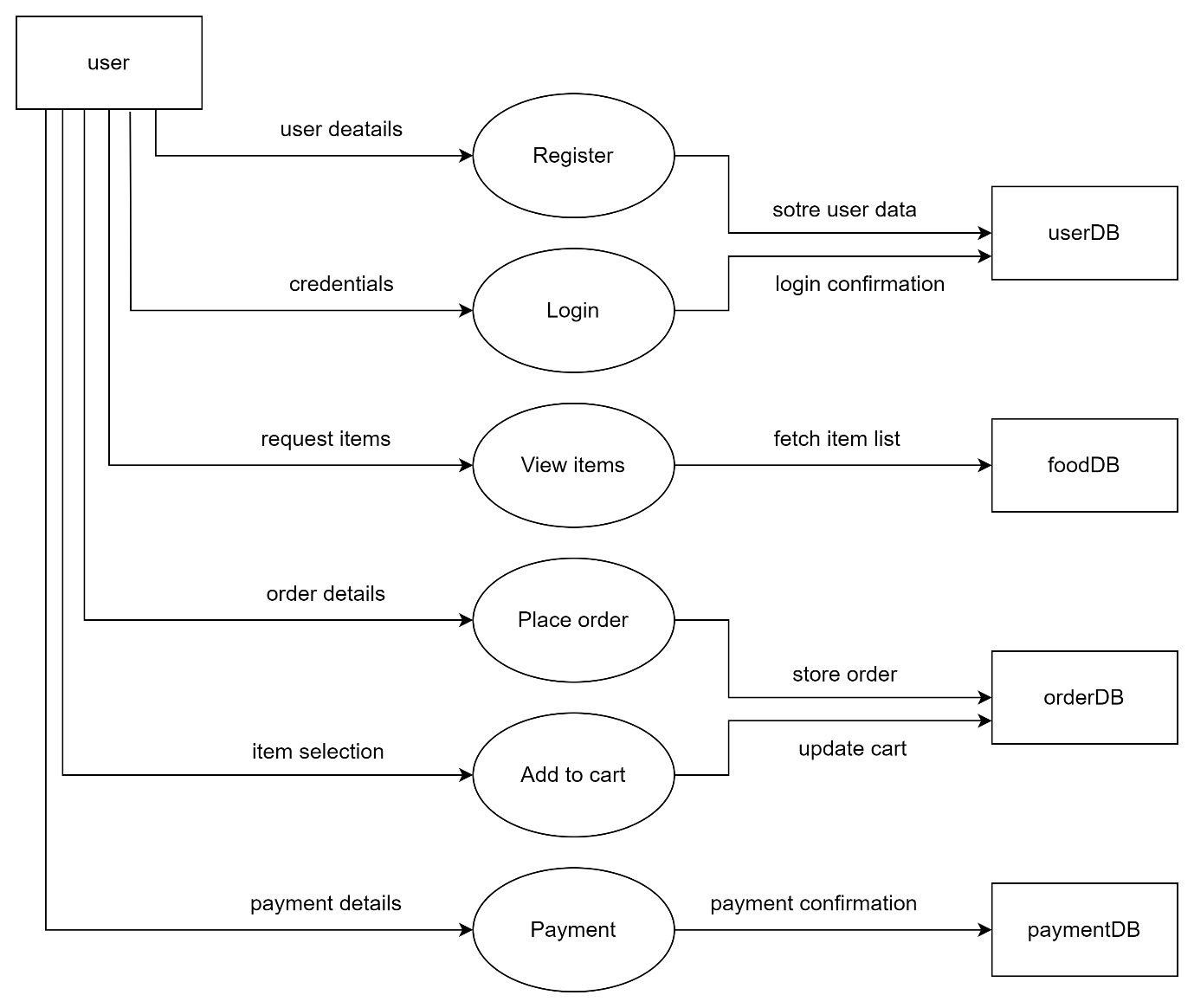
Level 0 DFD (System Overview)

[User] ---> [Food Delivery System] ---> [Restaurant/Admin/Delivery]

Level 1 DFD (Order Process)

1. User selects restaurant and places order.
2. System forwards order to the restaurant.
3. Delivery agent picks up and delivers the order.
4. Payment gateway handles transactions.

**Level 1 dfd(user)**

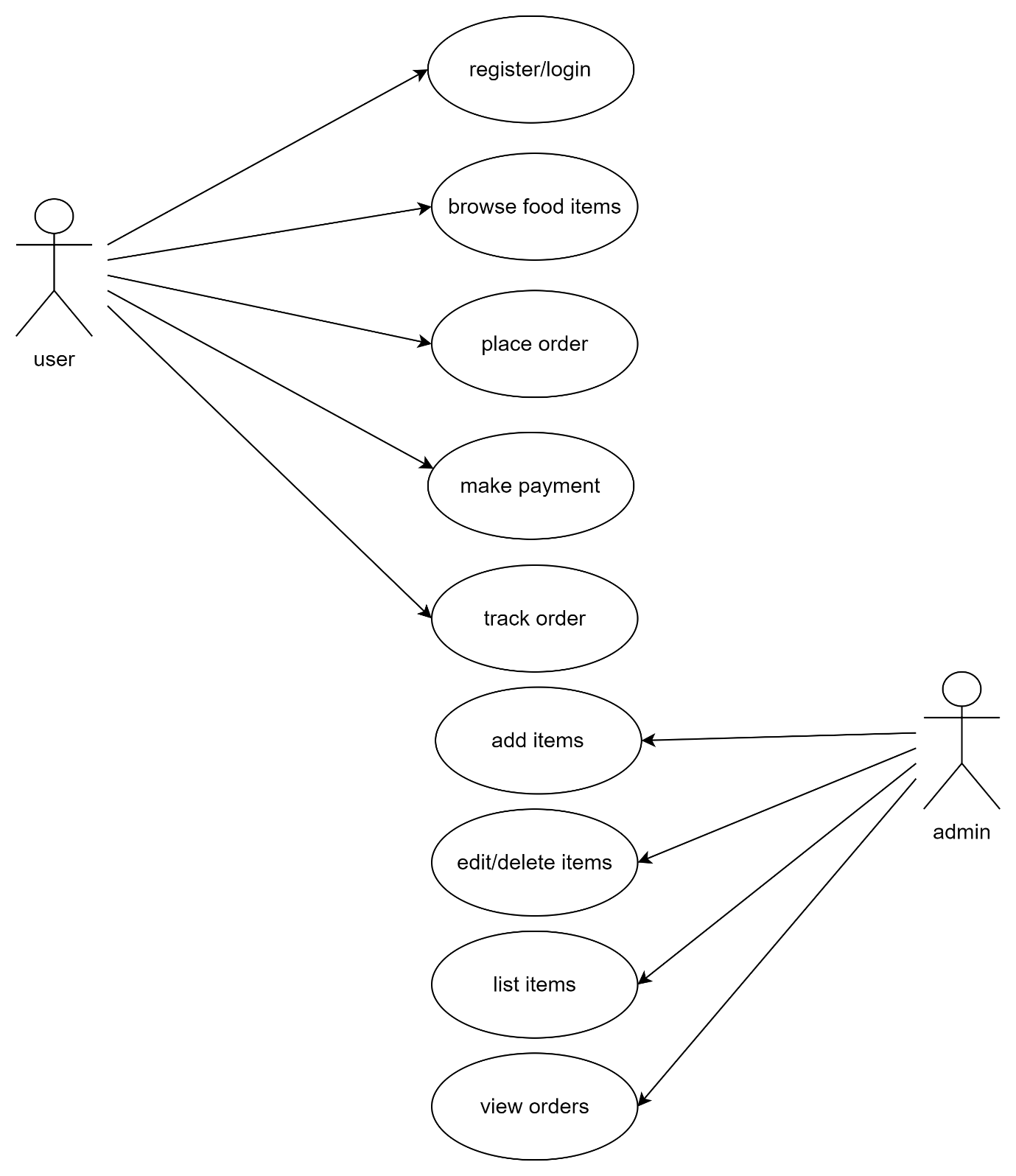


**Use Case Diagram**

Actors:

* User
* Admin
* Restaurant
* Delivery Staff

**Use case diagram**

****

Use Cases:

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