

**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 **Ms. Yogitha M, Assistant Professor** 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 | **10-11** |
| **5** | **SYSTEM DESIGN** |  |
| 5.1 | Data Flow Diagrams | **12-14** |
| 5.2 | Use Case Diagram |  |
| **6** | **SYSTEM IMPLEMENTATION** |  |
| 6.1 | Implementation | **15-22** |
| 6.2 | Screenshots |  |
| **7**  7.1 | **SYSTEM TESTING**  Test cases | **23-25** |
| **8**  8.1  8.2 | **RESULTS AND DISCUSSION**  Conclusions  Future Enhancements | **26** |
| **9**  9.1  9.2 | **REFERENCES**  Text references  Web references | **27** |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Figure No** | **Title** | **Page No.** |
| 1. | Figure 5.1.1 | Data Flow Diagram - Level 0 | 12 |
| 2. | Figure 5.1.2 | Data Flow Diagram - Level 1[Admin] | 12 |
| 3. | Figure 5.1.3 | Data Flow Diagram - Level 1[User] | 13 |
| 4. | Figure 5.1.4 | Use Case Diagram | 14 |
| 5. | Figure 6.2.1 | User Home Page | 17 |
| 6. | Figure 6.2.2 | Explore Menu Page | 17 |
| 7. | Figure 6.2.3 | Menu Page | 18 |
| 8. | Figure 6.2.4 | User Sign Up Page | 18 |
| 9. | Figure 6.2.5 | User Login Page | 19 |
| 10. | Figure 6.2.6 | Cart Page | 19 |
| 11. | Figure 6.2.7 | User Delivery Information Page | 20 |
| 12. | Figure 6.2.8 | Payment Page | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| 13. | Figure 6.2.9 | My Orders Page | 20 |
| 14. | Figure 6.2.10 | Add Items Page | 21 |
| 15. | Figure 6.2.11 | List Items Page | 21 |
| 16. | Figure 6.2.12 | Order Page | 22 |

# ABSTRACT

The project "**OuickBite Food Delivery Application** 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, 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.

The QuickBite Food Delivery Application Using MERN Stack offers a streamlined platform for users to browse, order, and track food delivery in real-time. With a React.js frontend and Node.js backend, the system ensures fast, secure, and efficient order management. MongoDB stores user and order data securely, and features such as user authentication, order tracking, and payment integration enhance usability and security. The QuickBite platform delivers a reliable and user-friendly experience, optimizing restaurant operations while ensuring customer satisfaction.

**CHAPTER 1**

**1. INTRODUCTION**

**1.1 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.

### 1.2 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.

**2**. **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.

**3**. **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.

**4**. **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).

### 1.3 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 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 and filtering options to improve usability.

**2. Efficient Order Management:**

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

**3. 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.

**4. Multiple User Interfaces:**

* User Dashboard: For browsing menus, placing orders, and tracking deliveries.
* Admin Dashboard: For managing menus, processing orders and updating order statuses.

5. Payment Integration:

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

6. 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.

### 1.4 Existing System

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

### 1. High Delivery Charges During Peak Hours

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

### 2. Unreliable Tracking Systems

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

### 3. Limited Order Customization

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

### 4. Inconsistent Customer Support

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

### 5. Monopoly Effects

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

### 1.5 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:

### 1. Fast and Reliable Delivery Tracking

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

### 2. Modular Architecture

### Independent modules (User, Restaurant, Admin, Delivery) enhance system

### performance, simplify maintenance, and enable easy upgrades.

### 3. 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.

### 4. User Authentication and Profiles

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

### 5. Payment Flexibility

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

### 1.6 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

## 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 Web Sockets improves transparency and customer satisfaction (Gupta et al., 2021). Integration with payment gateways like Stripe 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.

## CHAPTER 3

**3. SYSTEM CONFIGURATION**

**3.1 Hardware Requirements**

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

**3.2 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.

## CHAPTER 4

**4. MODULE DESCRIPTION**

**4.1 Modules**

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 Menu:** Users can explore the menu and select their preferred dishes.
* **Place Orders:** Customers can add items to the cart and place orders through a simple checkout process.
* **Order Tracking:** Users can monitor the status of their orders, from preparation to delivery, in real time.
* **User Accounts:** Users can register and log in to save delivery addresses, view their order history, and access personalized recommendations.
* **Notifications and Alerts:** Users receive updates about their order status, along with alerts for promotions and offers.

**2. 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.
* **Menu and Order Management:** Admins can add, update, or remove menu items and monitor customer orders.
* **Order Status Updates:** Admins update order statuses (e.g., "Preparing" or "Delivered") to keep users informed.
* **Performance Analytics:** This feature provides insights into peak usage periods and order trends to help optimize platform operations.
* **Issue Resolution and Refunds:** Admins handle customer complaints, manage refunds, and resolve disputes efficiently.
* **Security Management:** The module ensures data privacy and system security by detecting suspicious activities and managing encryption protocols.

This module plays a critical role in ensuring the platform operates efficiently and securely.

**3. Payment Module**

The platform integrates the Stripe payment gateway to enable secure and seamless transactions. Key aspects include:

* **Multiple Payment Options:** Users can pay through credit/debit cards, UPI, and wallets supported by Stripe.
* **Payment Confirmation:** Upon successful payment, users receive notifications confirming their orders.
* **Refunds:** Admins can initiate refunds through Stripe for canceled or failed orders to ensure timely reimbursement.
* **Transaction Security:** Payments are encrypted, ensuring compliance with industry standards to prevent fraud.
* **Admin Monitoring:** Admins can track all transactions and resolve payment-related issues through the Stripe dashboard. This version aligns with your setup, focusing on **Stripe’s capabilities** without introducing unnecessary modules.

## CHAPTER 5

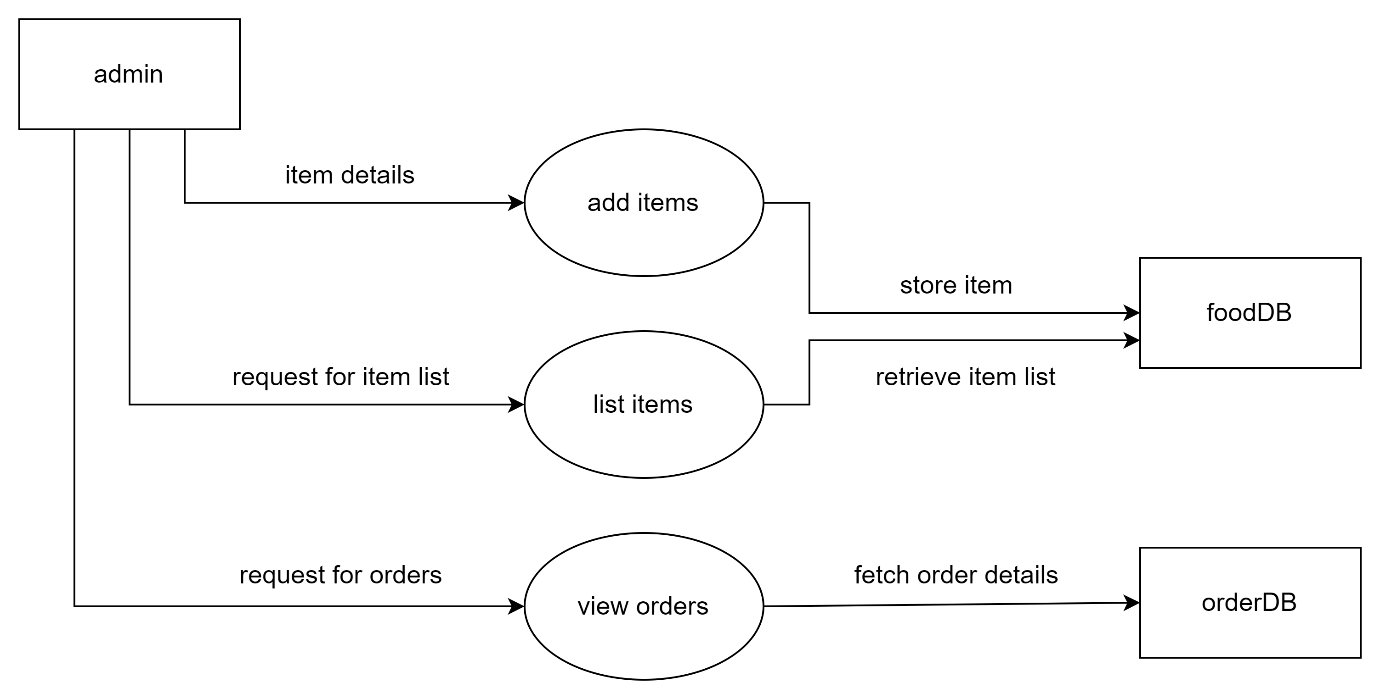
**5. SYSTEM DESIGN**

**5.1 Data Flow Diagrams (DFD)**

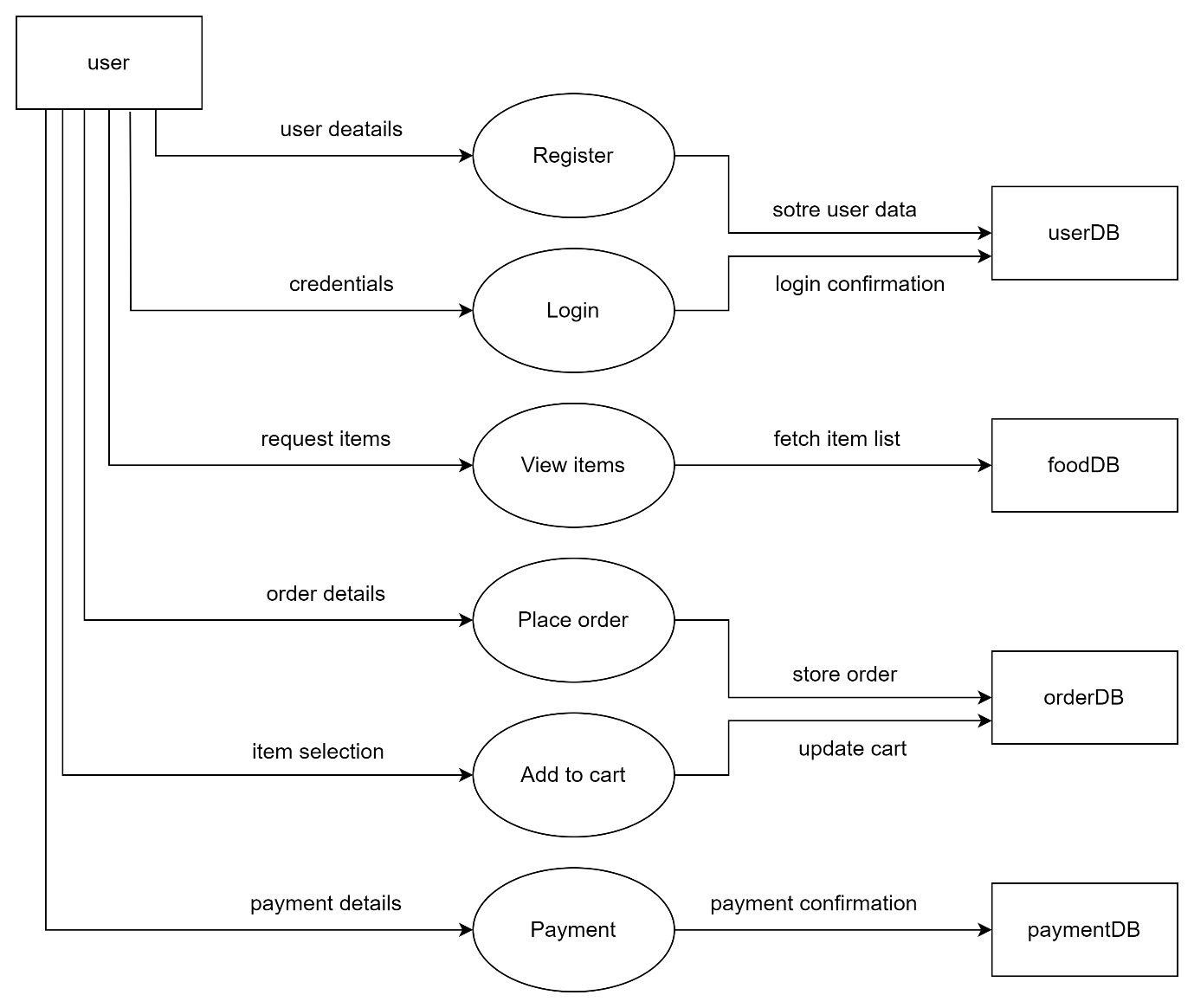
**5.1.1 DFD Level 0:**

****

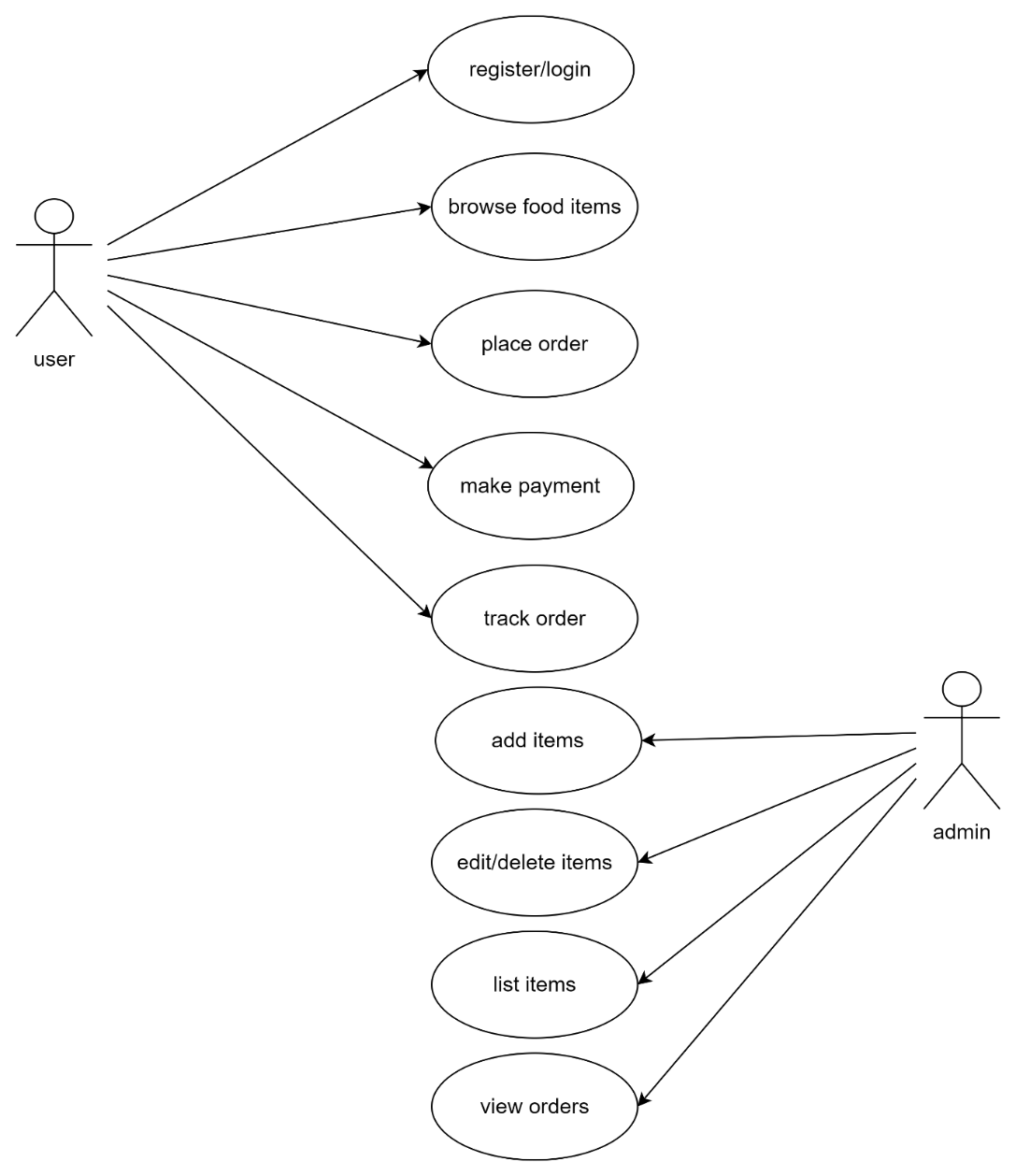
**5.1.2 DFD Level 1 for Admin:**



**5.1.3 DFD Level 1 for User:**



**5.2 Use Case Diagram**

****

## CHAPTER 6

**6. SYSTEM IMPLEMENTATION**

**6.1 Implementation**

The QuickBite Food Delivery Application is developed using the **MERN stack** (MongoDB, Express.js, React.js, Node.js) with two modules: **User Module** and **Admin Module**. Below are the key aspects of the implementation.

**1. Technologies Used**

* **MongoDB**: NoSQL database for user, food, and order data.
* **Express.js & Node.js**: Backend logic and APIs.
* **React.js**: Frontend framework for building interactive UI.
* **JWT & Bcrypt**: For secure authentication and password management.

**2. Database Design**

* **User Collection**: Stores user details (user\_id, name, email, password, cartData).
* **Food Collection**: Stores food items (food\_id, name, description, price, image, category).
* **Order Collection**: Manages orders (order\_id, user\_id, items, amount, address, date, status, payment).

**3. User Authentication**

* **Registration and Login**: Users register and log in securely using **JWT**.
* **Role-based Access Control**: Admin and user roles determine access to the respective modules.
* **Session Management**: JWT tokens stored in local storage maintain user sessions.

**4. User Module (Frontend: React.js)**

* **Features**:
  + - Users browse food items, add them to the cart, and place orders.
    - Cart management with quantity updates and real-time totals.
    - Order status tracking from the user dashboard.
* **UI Design**:
  + - Responsive pages built using React.js to ensure a smooth user experience.

**5. Admin Module**

* **Features**:
  + - Admins manage food items (add, update, list items).
    - View and track order statuses.
    - No direct integration with the user interface for security and modularity.
* **Admin Access**:
  + - Admins access the system through a separate **admin module**, typically under /admin.

**6. System Logic**

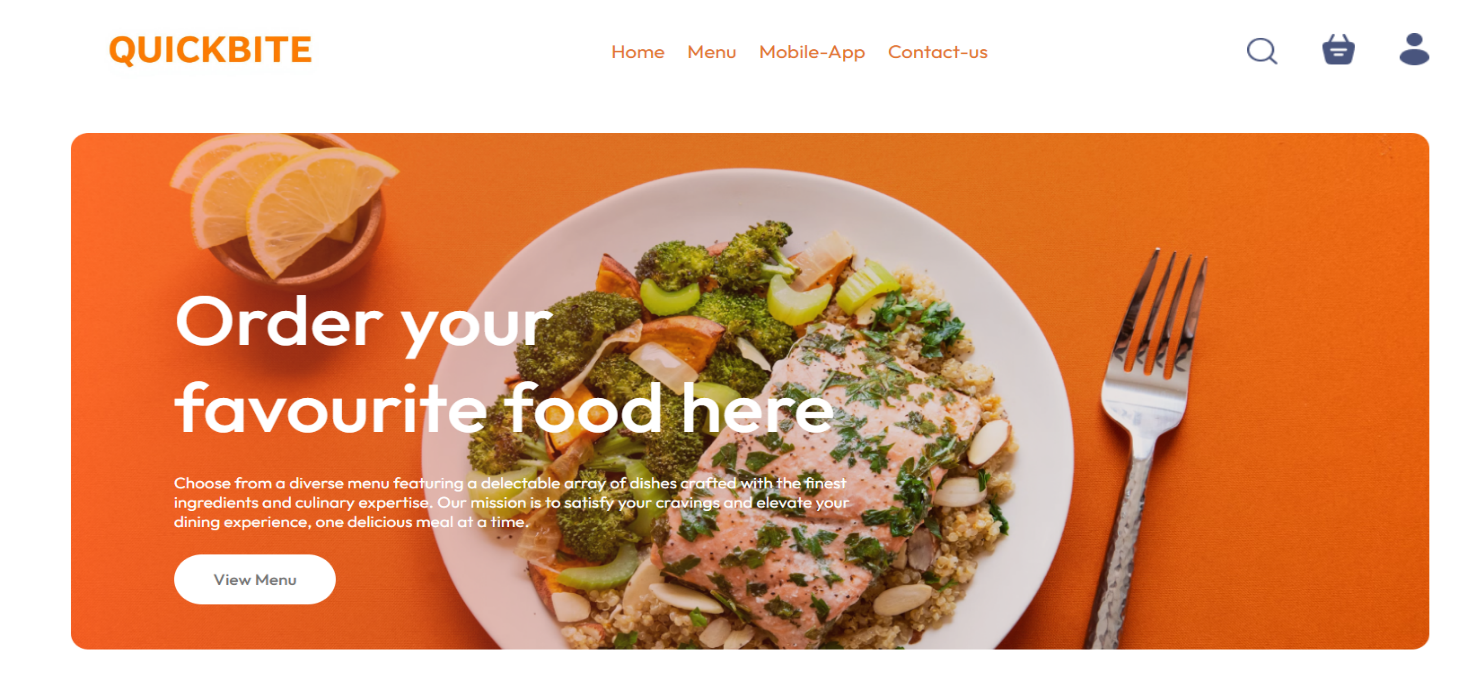
* **Order Handling**: Orders are stored in the **Order Collection** and can be updated (e.g., "Pending", "Delivered").
* **Cart Management**: Users add or remove items from the cart, with live price updates.
* **Admin Management**: Admins control item availability and pricing via the admin module.

**7. Security and Testing**

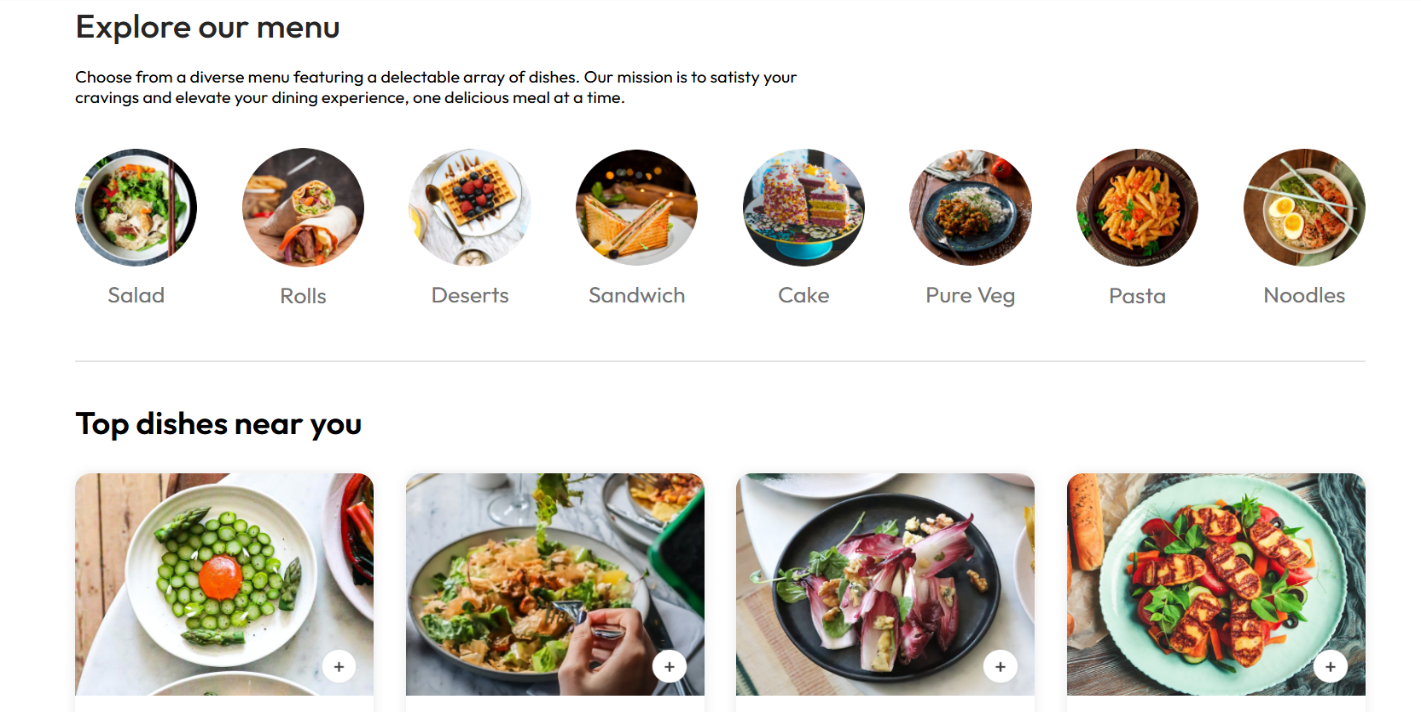
* **Bcrypt for Passwords**: Ensures secure password storage.
* **JWT Authentication**: Validates user sessions for secure access.
* **Testing**:
* **Unit Testing**: Verifying individual components like login and cart.
* **Integration Testing**: Ensuring smooth interaction between backend APIs and frontend.
* **System Testing**: Validating that both modules function seamlessly.

**6.2 Screenshots**

**User Module**



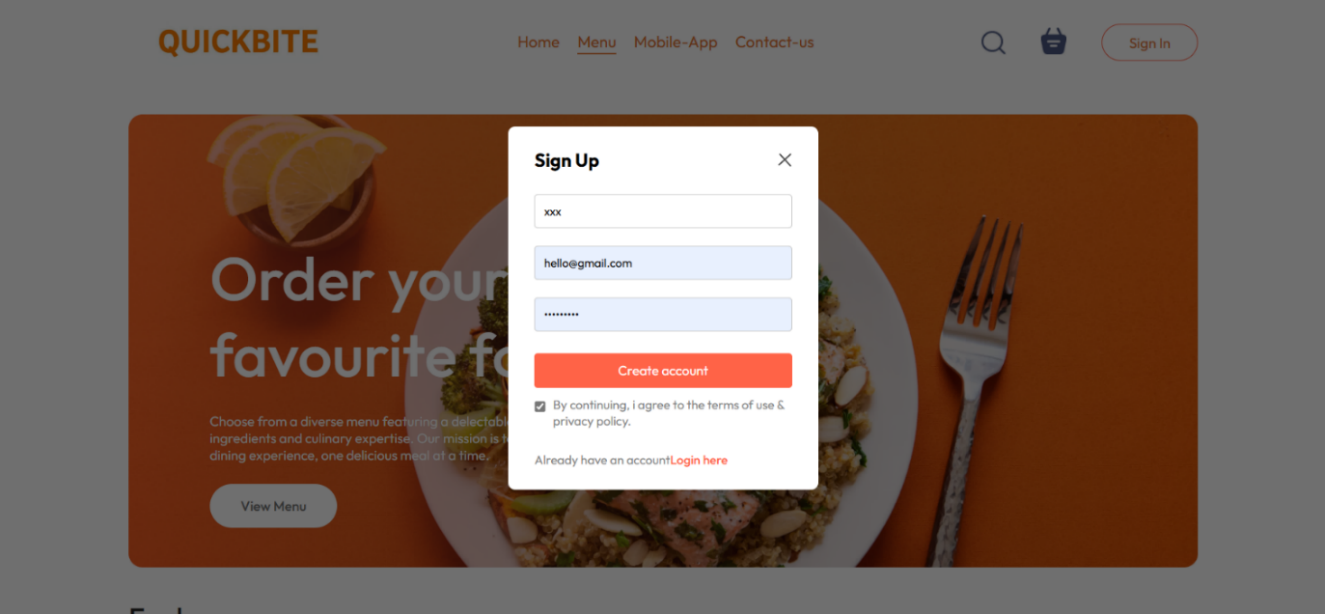
**Figure 6.2.1 User Home Page**



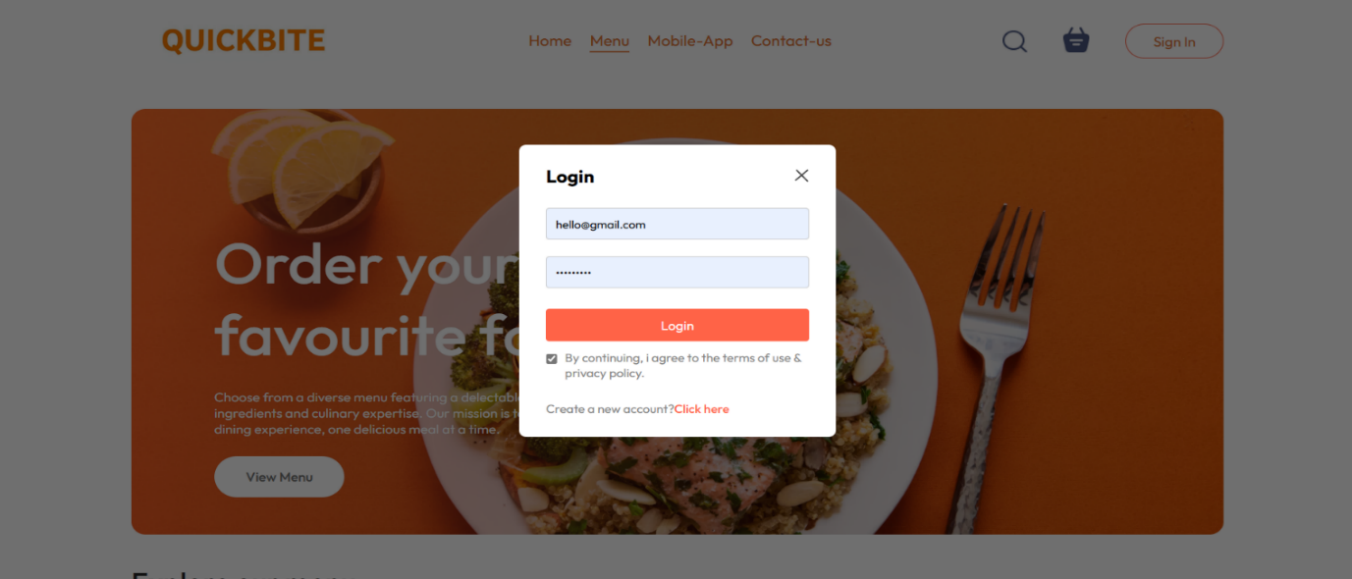
**Figure 6.2.2 Explore our menu page**



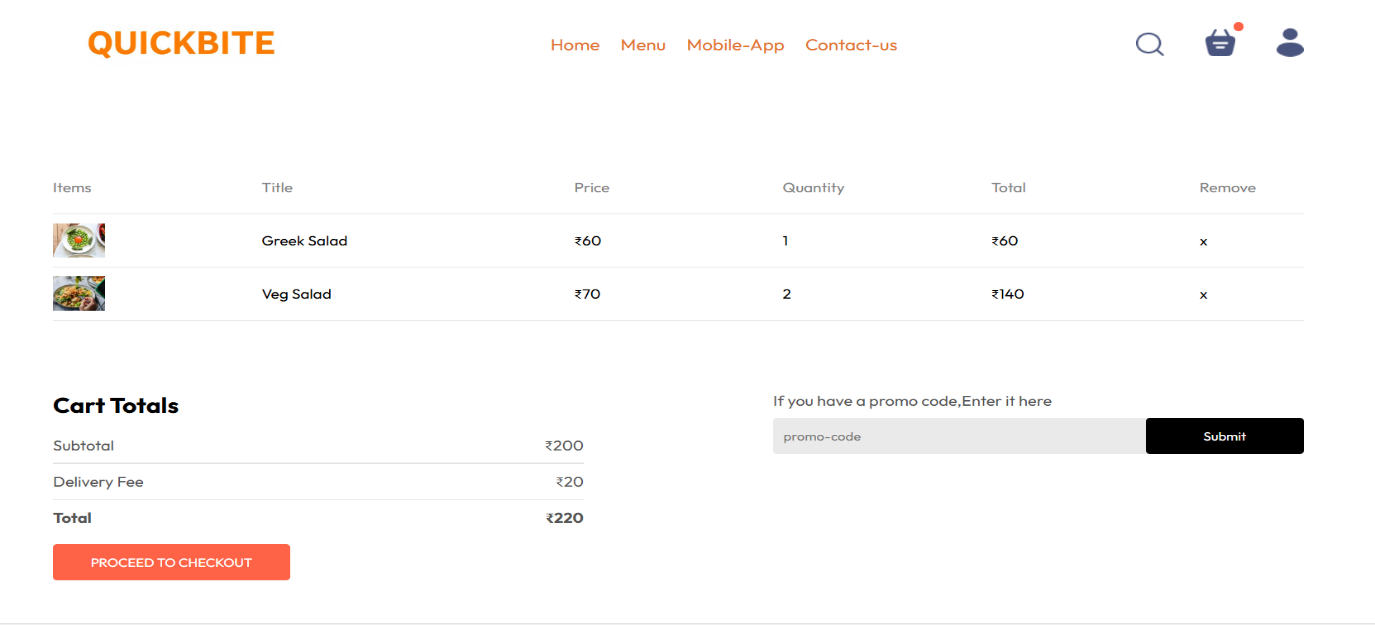
**Figure 6.2.3 Menu page**



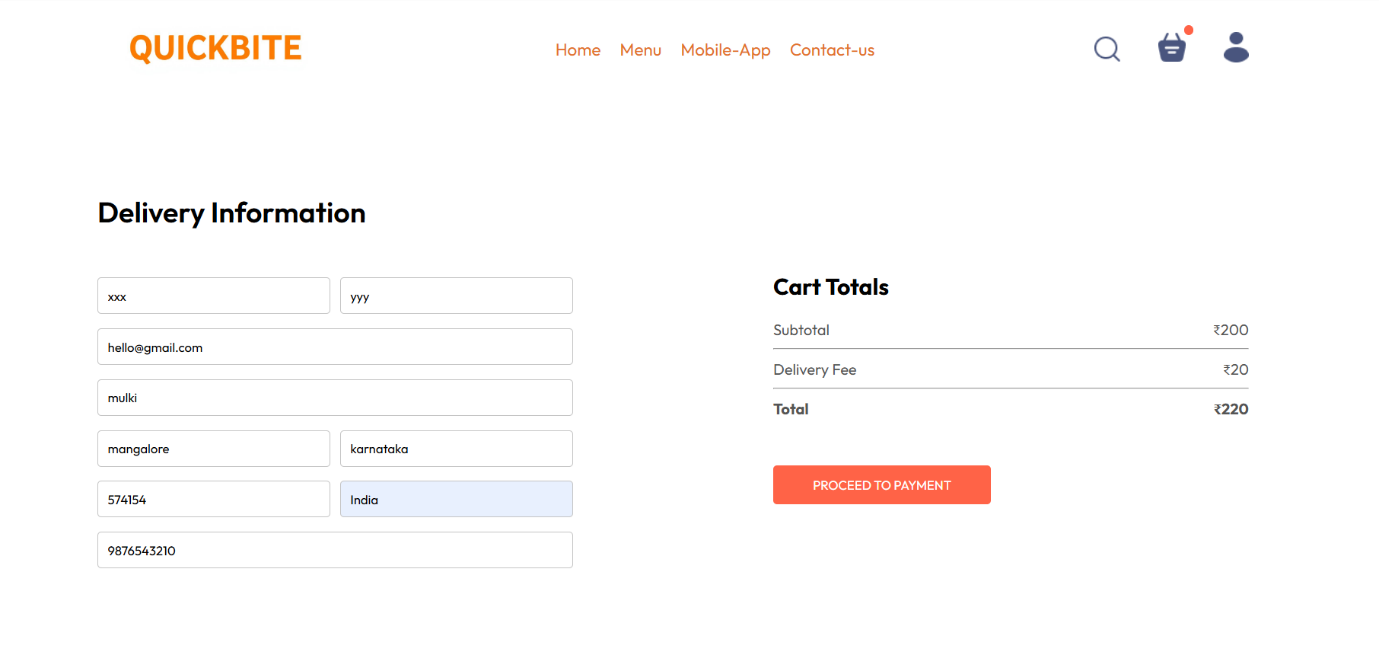
**Figure 6.2.4 User Sign Up page**



**Figure 6.2.5 User Login page**



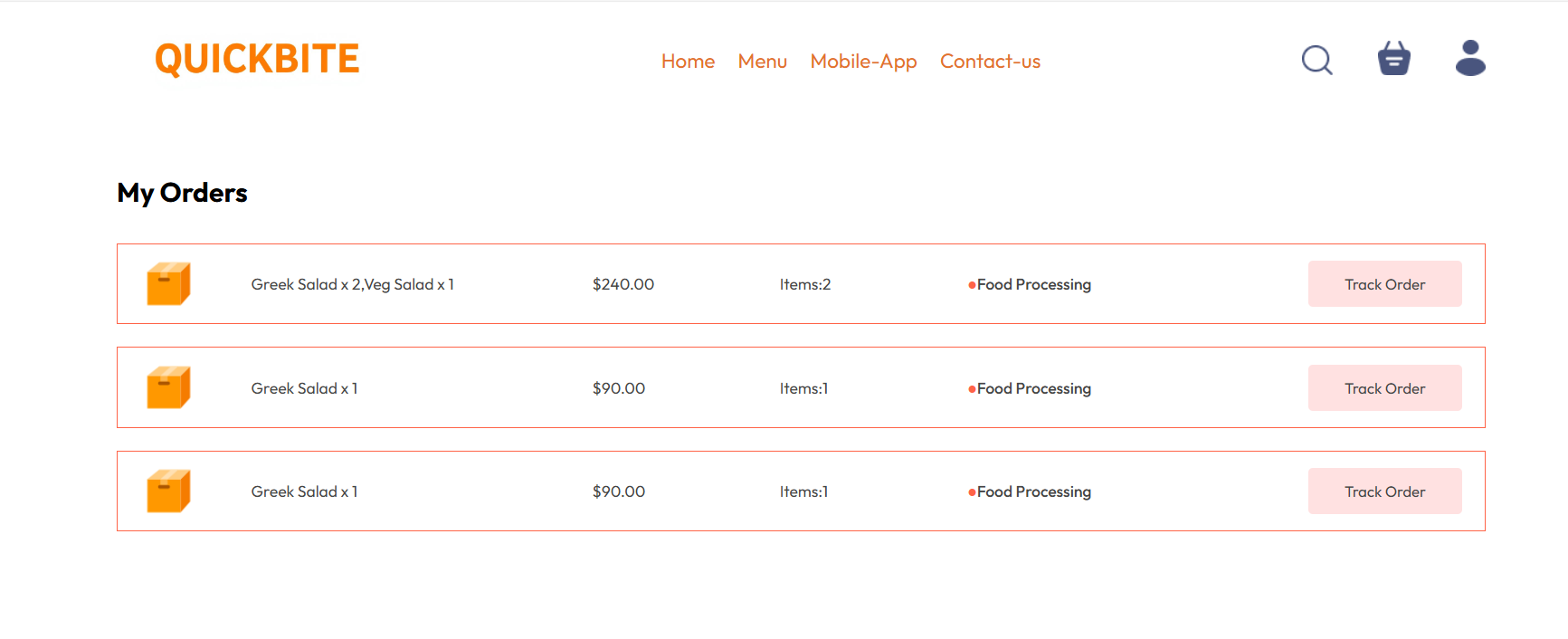
**Figure 6.2.6 Cart page**



**Figure 6.2.7 Delivery Information page**

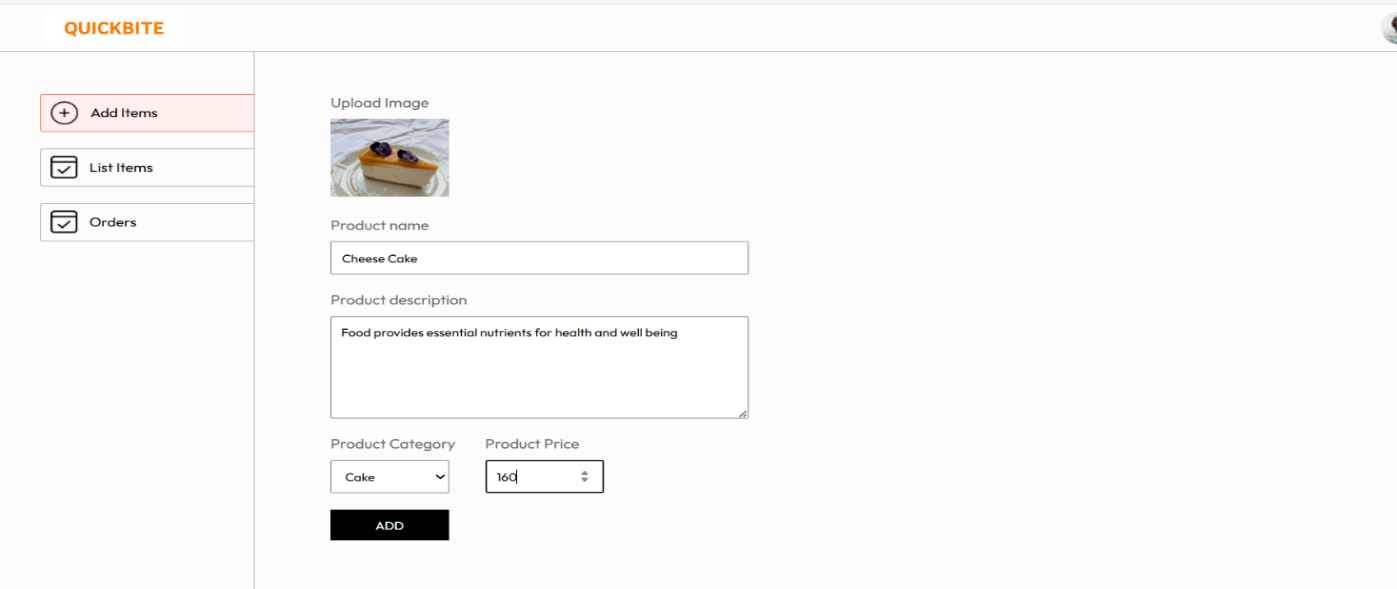


**Figure 6.2.8 Payment page**

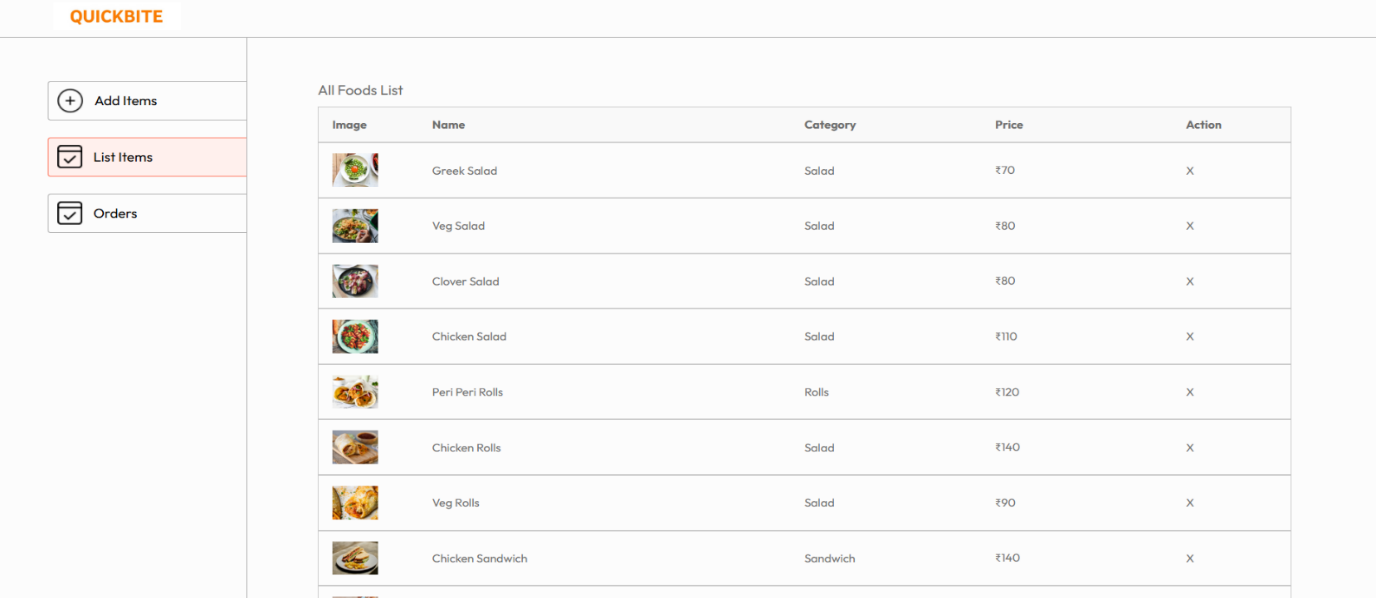


**Figure 6.2.9 My Orders page**

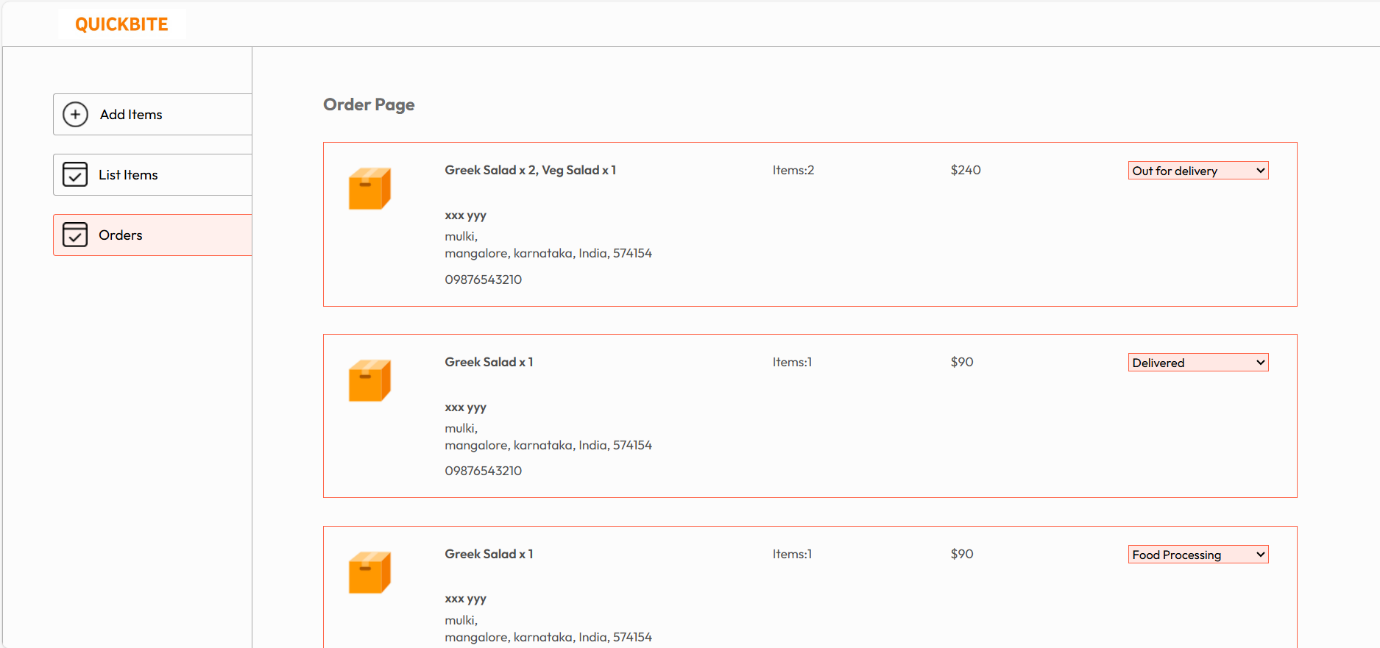
**Admin Module**



**Figure 6.2.10 Add Items page**



**Figure 6.2.11 List Items page**



**Figure 6.2.12 Order page**

## CHAPTER 5

**7. SYSTEM TESTING**

**7.1 Test Cases**

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.

**Unit testing for user Login Form**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **ID** | **Unit Test Cases** | **Expected Result** | **Actual Results** |
| 1 | Click Login In button without entering Email. | Display message  “Please fill out this field”. | Successful |
| 2 | Click Login button without entering Password. | Display message  “Please fill out this field”. | Successful |
| 3 | Click Login button by entering Email  address without ‘@’. | Display message  “Please include an @ in the email address”. | Successful |
| 4 | Click Login button by entering Email  address without ‘.’. | Display message  “Please enter a part following ‘@’”. | Successful. |
| 5 | Click Login button by entering invalid Password. | Display message  “Invalid credentials”. | Successful. |
| 6 | Click Login button by entering invalid Email. | Display message  “User Doesn’t exists”. | Successful. |

**Unit testing for Admin module Add Item form**

**Unit testing for User Sign Up form**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Unit Test Cases** | **Expected Result** | **Actual Results** |
| 1 | Click Create account button without entering  Name. | Display message  “Please fill out this field”. | Successful |
| 2 | Click Create account button without entering  Email. | Display message  “Please fill out this field”. | Successful |
| 3 | Click Create account button without entering Password. | Display message  “Please fill out this field”. | Successful |
| 4 | Click Create account button by entering Email  address without ‘@’ . | Display message  “Please include an ‘@’ in the email address”. | Successful |
| 5 | Click Create account button by entering Email  address without ‘.’ . | Display message  “Please enter a valid email”. | Successful |
| 6 | Click Create account button after entering Password less than 8 digits. | Display message  “Please enter a strong password”. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Unit Test Cases** | **Expected Result** | **Actual Results** |
| 1 | Click Add button without uploading Image. | Display message  “Please select a file”. | Food Added |
| 2 | Click Add button without entering Product Name. | Display message  “Please fill out this field”. | Food Added |
| 2 | Click Add button without entering Product Description. | Display message  “Please fill out this field”. | Food Added |
| 3 | Click Add button without entering Product Price. | Display message  “Please fill out this field”. | Food Added |

**Unit testing for User Delivery information form**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Unit Test Cases** | **Expected Result** | **Actual Results** |
| 1 | Click Proceed to Payment button without entering First name. | Display message  “Please fill out this field”. | Successful |
| 2 | Click Proceed to Payment button without entering Last name. | Display message  “Please fill out this field”. | Successful |
| 3 | Click Proceed to Payment button without entering Email address. | Display message  “Please fill out this field”. | Successful |
| 4 | Click Proceed to Payment button without entering Street. | Display message  “Please fill out this field”. | Successful |
| 5 | Click Proceed to Payment button without entering City. | Display message  “Please fill out this field”. | Successful |
| 6 | Click Proceed to Payment button without entering State. | Display message  “Please fill out this field”. | Successful |
| 7 | Click Proceed to Payment button without entering Zip code. | Display message  “Please fill out this field”. | Successful |
| 8 | Click Proceed to Payment button without entering Country. | Display message  “Please fill out this field”. | Successful |
| 9 | Click Proceed to Payment button without entering phone. | Display message  “Please fill out this field”. | Successful |
| 10 | Click Proceed to Payment button by entering Email  address without ‘@’ . | Display message  “Please include an ‘@’ in the email address”. | Successful |
| 11 | Click Proceed to Payment button after entering Phone Number less than 10 digits. | Display message  “Please match the requested format”. | Successful |

**CHAPTER 8**

**8.RESULTS AND DISCUSSION**

**8.1 Conclusions**

* **Functionality**:  
  All core features, including user registration, login, item browsing, cart management, order placement, and admin functionalities, worked as expected.
* **Performance**:  
  The system successfully handled simultaneous orders and requests without noticeable delays or downtime.
* **Security**:  
  No critical vulnerabilities were found. Secure authentication with **JWT** and password encryption using **bcrypt** were implemented to protect user data.
* **Usability**:  
  The interface is intuitive, with clear navigation for both users and admins, ensuring a smooth user experience across devices.

The **QuickBite Food Delivery** **Application** is a **reliable, user-friendly, and secure** solution for food delivery. It offers a smooth experience for customers to browse, order, and track food, while the admin module efficiently manages food items and orders. The system’s modular design makes it easy to maintain and scale for future growth.

**8.2 Future Enhancements**

1. AI-Powered Recommendations.

2. Chatbot Integration.

3. Enhanced Analytics.

4. Loyalty Programs.

5. Multi-Language Support.

**CHAPTER 9**

**9. REFERENCES**

**9.1 Text References:**

1. "Software Testing: Concepts and Operations" by Srinivasan Desikan and Gopalaswamy Ramesh.

2. "Introduction to Algorithms" by Thomas H. Cormen.

3. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.

**9.2 Web References:**

* <http://www.w3school.com/>
* <http://www.tutorialsapoint.com/>
* <http://www.geeksforgeeks.com/>
* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)