Project Report

FoodieConnect – Online Food Ordering & Delivery System

1. Introduction

The modern lifestyle has led to a significant rise in online food delivery systems. Customers prefer convenience, variety, and efficiency in ordering food from their favorite restaurants. FoodieConnect is a full-stack web application that provides a seamless food ordering experience, integrating features such as restaurant search, user authentication, cart management, order tracking, and real-time status updates.

2. Objectives

- To design and develop a scalable online food ordering platform.
- To implement secure user authentication and authorization.
- To integrate restaurant listings and filtering for optimized search.
- To provide real-time cart and order tracking.
- To ensure smooth deployment using Vercel (frontend) and Render (backend).

3. System Requirements

Hardware Requirements:
- Processor: Intel i5 or above
- RAM: Minimum 8 GB
- Storage: 500 GB HDD / 256 GB SSD
- Internet: Stable broadband connection

Software Requirements:
- Frontend: React.js, TailwindCSS
- Backend: Node.js, Express.js
- Database: MongoDB Atlas
- Deployment: Vercel (frontend), Render (backend)
- Version Control: Git & GitHub

4. System Architecture

The system follows a client–server architecture:

1. Frontend (Client):
- Developed with React.js.
- Handles UI, restaurant search, cart management, and placing orders.

2. Backend (Server):
- Built with Express.js.

- Provides REST APIs for authentication, restaurants, orders, and payments.
- Handles CORS policies for secure communication.

3. Database:
- MongoDB stores users, restaurants, menus, and orders.

5. Features Implemented

■ Authentication – User registration & login with JWT tokens
■ Restaurant Search – Search & filter restaurants by delivery time, location, name
■ Cart Management – Add/remove items, store cart in localStorage
■ Orders – Place orders, update order status, track orders
■ Admin Panel – Restaurant owners can update menu, order status
■ Deployment – Frontend on Vercel, Backend on Render

6. Database Design

Main Collections:
- Users: { name, email, passwordHash, role }
- Restaurants: { name, menuItems, deliveryTime }
- Orders: { userId, restaurantId, items, status, createdAt }
- Cart (client-side): stored in localStorage with { items[], restaurantId }

7. Implementation Details

Frontend:
- Built using React.js + TailwindCSS.
- Uses Context API for managing cart state.
- Axios used for API requests with JWT authentication.

Backend:
- Express.js REST API with modular routes.
- Authentication with JWT and password hashing (bcrypt).
- Middleware for validation, error handling, and CORS.

Deployment:
- Frontend deployed on Vercel.
- Backend deployed on Render with environment variables.
- MongoDB Atlas for scalable cloud database.

8. Security

- JWT authentication for secure login.
- Bcrypt for password hashing.
- Role-based access control (User / Admin).
- CORS restricted to production domain (https://foodie-connect.vercel.app).

9. Testing

- Unit Testing: Tested API endpoints with Postman.
- Integration Testing: Verified frontend-backend connectivity.
- User Testing: Checked flows like login → search → add to cart → order.

10. Challenges & Solutions

- CORS Issues – Solved by configuring allowed origins in Express.
- PATCH Requests Blocked – Fixed by adding PATCH in CORS methods.
- Local vs Production Config – Used environment variables for domain-specific settings.

11. Future Enhancements

- Real-time order tracking using WebSockets.
- Online payment gateway integration.
- AI-based restaurant recommendations.
- Mobile app version using React Native.

12. Conclusion

FoodieConnect successfully demonstrates the design and implementation of a full-stack online food ordering system. The platform is scalable, secure, and user-friendly, providing an end-to-end solution from restaurant discovery to order delivery.

13. References

1. React.js Official Documentation – https://react.dev/
2. Express.js Guide – https://expressjs.com/
3. MongoDB Atlas – https://www.mongodb.com/atlas
4. Vercel Deployment Docs – https://vercel.com/docs
5. Render Deployment Docs – https://render.com/docs