Mini Project Report

FoodieFleet: A MERN Stack Food Delivery Website

Student Name	Roll No.	PRN	Batch	Panel
Shubham Waditake	40	1032232463	A1	CSF Panel A
Yash Darekar	16	1032231148	A1	CSF Panel A

Course: Full Stack Development

Course Code:

Submission Date: November 4, 2025

Abstract

FoodieFleet is a full-stack food delivery web application built using the MERN stack (MongoDB, Express.js, React.js, Node.js). The project's primary purpose is to apply and demonstrate full-stack development principles by creating a real-world, scalable, and user-centric platform. The application allows users to browse restaurants, view menus, add items to a cart, and place mock orders. Key features include secure user authentication with JSON Web Tokens (JWT), a responsive and interactive frontend, a robust RESTful API backend, and a flexible NoSQL database. This report details the project's objectives, architecture, implementation process, and final outcomes.

Introduction

Background and Motivation

In recent years, the online food delivery market has seen explosive growth, fundamentally changing consumer dining habits. Platforms like Zomato, Swiggy, and Uber Eats have set the standard for convenience, offering users a wide array of choices at their fingertips. This project is motivated by the desire to understand and replicate the core technologies that power these complex, high-demand applications. The MERN stack was chosen as it represents a modern, efficient, and popular technology stack for building such platforms, allowing for a seamless development experience using JavaScript across the entire stack.

Problem Statement

Many small to medium-sized restaurants lack the technical expertise or financial resources to develop and maintain their own online ordering and delivery applications. Customers, in turn, are inconvenienced by traditional, inefficient methods like phone-in orders, which are prone to human error and lack visual menus or transparent pricing.

The problem is to design and implement a centralized, web-based platform that connects users with multiple restaurants, providing a simple, secure, and streamlined process for browsing menus and ordering food online.

Project Objectives

The primary objectives for the FoodieFleet project are:

1. To design and build a full-stack MERN application from the ground up.
2. To create a responsive, component-based frontend using React.js for a seamless user experience (UX) on all devices.
3. To develop a secure and scalable RESTful API using Node.js and Express.js for handling all business logic, data, and user requests.
4. To implement a flexible database schema using MongoDB to store data for users, restaurants, menu items, and orders.
5. To integrate secure user authentication and authorization using JSON Web Tokens (JWT).
6. To implement core e-commerce functionalities: user registration/login, restaurant browsing, menu viewing, cart management (add, remove, update), and a mock checkout process.

Scope and Limitations

Scope:

User Module: User registration, login, and profile management.

Restaurant Module: Listing all available restaurants and viewing specific restaurant menus. Cart Module: Full cart functionality, including adding, removing, and adjusting item quantities. Order Module: Placing a mock order and saving it to the user's order history.

Limitations:

Payment Gateway: The project uses a mock checkout process. It does not integrate a real-time payment gateway like Stripe or Razorpay.

Live Order Tracking: There is no real-time GPS tracking for delivery personnel.

Admin & Restaurant Panel: The current scope focuses on the customer-facing application. A dedicated dashboard for restaurant owners or site administrators is not implemented.

Reviews and Ratings: The feature for users to add reviews and ratings to restaurants is not included.

Literature Review

The development of FoodieFleet is based on the MERN stack, a popular choice for full-stack web development.

Existing Systems: Platforms like Zomato and Uber Eats serve as the primary inspiration. These systems are built on a complex microservice architecture. They provide robust features like real- time tracking, payment integration, and sophisticated recommendation algorithms. While FoodieFleet aims for similar core functionality, it adopts a monolithic backend (for simplicity in a mini-project) and focuses on the essential order-flow process.

Technology Justification (MERN):

MongoDB: A NoSQL database, is chosen for its flexible schema-less design, which is ideal for handling diverse data like menus and user profiles. Its document-based (BSON) storage aligns perfectly with JavaScript objects, simplifying data transfer.

Express.js: A minimal and flexible Node.js web application framework, provides a robust set of features for building our RESTful API. Its middleware-based architecture simplifies tasks

like request handling, routing, and authentication.

React.js: A declarative, component-based JavaScript library for building user interfaces. Its use of a virtual DOM allows for high-performance Ul updates, crucial for a responsive application like a food delivery site where cart and item data change frequently.

Node.js: A JavaScript runtime environment that allows us to run JavaScript on the server. This enables a full-stack JavaScript experience, reducing context-switching for developers and allowing for code sharing between the frontend and backend.

Compared to traditional stacks like LAMP (Linux, Apache, MySQL, PHP), the MERN stack offers faster development, higher scalability (due to Node.js's non-blocking I/O), and a more unified development ecosystem.

Methodology

Choice of Technologies and Tools

Frontend: React 18 (with Hooks), React Router DOM (for page navigation), Axios (for API requests), CSS Modules (for styling).

Backend: Node.js, Express.js.

Database: MongoDB Atlas (cloud-hosted database), Mongoose (Object Data Modeling library).

Authentication: JSON Web Tokens (JWT), bcrypt.js (for password hashing).

Development Tools: VS Code, Postman (for API testing), Git & GitHub (for version control).

System Architecture

The project follows a 3-tier client-server architecture:

1. Client-Tier (Frontend): This is the React.js application that runs in the user's browser. It handles all Ul rendering and user interactions. It communicates with the backend via HTTP requests (using Axios) to a RESTful API.
2. Application-Tier (Backend): This is the Node.js/Express.js server. It acts as the middle layer, handling all business logic. It receives requests from the client, processes them, validates data, performs authentication/authorization, and interacts with the database.
3. Data-Tier (Database): This is the MongoDB database (hosted on Atlas). It stores all persistent data, including user credentials (hashed), restaurant information, menus, and order details.

Development Process

We adopted an Agile development methodology, breaking the project into small, manageable sprints:
Sprint 1: Planning and Backend Setup:

Defined database models (schemas) for Users, Restaurants, and Orders using Mongoose. Developed the core RESTful API endpoints for user authentication (register, login) using JWT. Set up API routes for fetching restaurant and menu data.

Tested all endpoints using Postman.

Sprint 2: Frontend Implementation:

Set up the React project structure with create-react-app.

Built static Ul components (Navbar, Restaurant Card, Menu Item).

Implemented routing using React Router.

Built the user registration and login forms.

Sprint 3: Integration and Core Features:

Integrated the frontend with the backend API for user authentication.

Fetched and displayed restaurant data on the home page.

Implemented the cart functionality using React's Context API for global state management.

Developed the mock checkout process, which sends the cart data to the backend to create an order.

Results and Discussion

Project Outcomes

The project was successfully completed, meeting all the primary objectives. The result is a functional and responsive food delivery web application, "FoodieFleet."

Key Implemented Features:

Secure User Authentication: Users can create an account, log in, and log out. Passwords are securely hashed using bcrypt, and sessions are managed using stateless JSON Web Tokens (JWTs) stored on the client.

Restaurant Browsing: The main page successfully fetches and displays a list of all available restaurants from the MongoDB database.

Dynamic Cart Management: Users can add items to their cart from different restaurant menus. The cart state is managed globally using React's Context API, allowing for real-time updates (add, remove, update quantity) without page reloads.

Mock Order Placement: On checkout, the cart details and user information are compiled and sent to the /api/orders endpoint, which validates the data and successfully creates a new order document in the database.

Discussion and Challenges

State Management: One of the main challenges on the frontend was managing the global state of the cart. We initially considered using simple props but quickly realized this led to "prop drilling." We adopted the React Context API as a lightweight and effective solution to share cart state and functions across the entire component tree.

Backend Authentication: Implementing the JWT authentication middleware was a critical task. Ensuring that protected routes (like order placement) were only accessible to logged-in users required careful setup of the middleware to verify the token from the Authorization header on incoming requests.

CORS: We encountered Cross-Origin Resource Sharing (CORS) errors when the React app (on localhost:3000) tried to communicate with the Express server (on localhost: 5000). This was resolved by implementing the cors middleware on the Express server to allow requests from the client's origin.

Conclusion

Summary of Findings

The "FoodieFleet" project successfully demonstrates the power and flexibility of the MERN stack for building modern, full-stack web applications. We successfully designed, developed, and deployed a core-feature food delivery platform, reinforcing our understanding of RESTful APIs, component-based frontend design, NoSQL databases, and secure authentication. The project meets all its stated objectives and provides a solid foundation for future development.

Recommendations for Future Work

While the current version of FoodieFleet is a successful prototype, its functionality can be greatly expanded. Future recommendations include:

1. Payment Gateway Integration: Integrate a real payment solution like Stripe or Razorpay to handle actual financial transactions.
2. Admin/Restaurant Panel: Create a separate dashboard for restaurant owners to manage their menus, view incoming orders, and update their information.

3. Live Order Tracking: Implement WebSockets (e.g., using Socket.io) to provide users with real- time updates on their order status.
4. Reviews and Ratings: Add a feature for users to rate and review restaurants and menu items, creating a more interactive community.
5. Search and Filtering: Implement advanced search functionality to allow users to filter restaurants by cuisine, location, or price.
6. References

[1] React.js Official Documentation. [Online]. Available: https://reactjs.org

[2] Node.js Official Documentation. [Online]. Available: https://nodejs.org

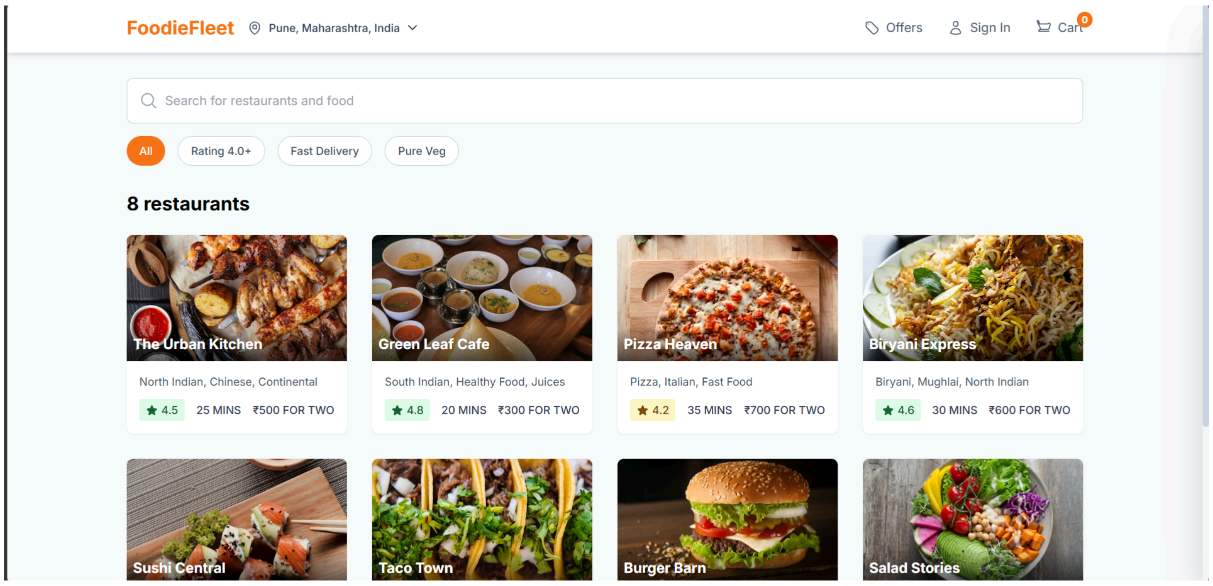
[3] Express.js Official Documentation. [Online]. Available: https://expressjs.com

[4] MongoDB Official Documentation. [Online]. Available: https://www.mongodb.com

[5] Mongoose.js Official Documentation. [Online]. Available: https://mongoosejs.com

[6] JSON Web Tokens Official Website. [Online]. Available: https://jwt.io

Appendices


FoodieFleet
Pune, Maharashtra, India
Offers
Sign In
Cart 0
Search for restaurants and food
All
Rating 4.0+
Fast Delivery
Pure Veg
8 restaurants
The Urban Kitchen
North Indian, Chinese, Continental
4.5 25 MINS ₹500 FOR TWO
Green Leaf Cafe
South Indian, Healthy Food, Juices
4.8 20 MINS ₹300 FOR TWO
Pizza Heaven
Pizza, Italian, Fast Food
4.2 35 MINS ₹700 FOR TWO
Biryani Express
Biryani, Mughlai, North Indian
4.6 30 MINS ₹600 FOR TWO
Sushi Central
Taco Town
Burger Barn
Salad Stories