PROJECT REPORT

ON

ONLINE FOOD ORDERING SYSTEM (THE FOOD CITY)

SUBMITTED BY: **NAME:** KUNAL **PROJECT:** THE FOOD CITY - ONLINE FOOD ORDERING SYSTEM **DATE:** FEBRUARY 2026

<div style="page-break-after: always;"></div>

CERTIFICATE

This is to certify that the project report entitled **"THE FOOD CITY"** submitted by **KUNAL** in partial fulfillment of the requirements for the completion of the Software Development Course is a record of bona fide work carried out under my supervision and guidance.

The project demonstrates a comprehensive understanding of Full Stack Web Development principles, specifically React.js ecosystem, Modern UI/UX Design, and State Management.

Signature of Guide Signature of HOD

<div style="page-break-after: always;"></div>

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher/guide who gave me the golden opportunity to do this wonderful project on the topic **"THE FOOD CITY" (Online Food Ordering System)**, which also helped me in doing a lot of Research and I came to know about so many new things. I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

I am also grateful to the open-source community for providing excellent tools and libraries like React, Tailwind CSS, and various npm packages that made this project possible.

<div style="page-break-after: always;"></div>

EXECUTIVE SUMMARY

"The Food City" is a state-of-the-art, web-based online food ordering application designed to revolutionize the way customers interact with restaurants. In an era where convenience is paramount, this application bridges the gap between culinary desires and digital accessibility.

The project is built using a modern technology stack, primarily featuring **React.js** for the frontend, **Tailwind CSS** for styling, and **Context API** for sophisticated state management. The application follows a Single Page Application (SPA) architecture ensuring seamless transitions, rapid load times, and an app-like user experience on the web.

Key Features Include:

- **Dynamic User Interface:** A visually stunning, responsive design that adapts to all device sizes (Mobile, Tablet, Desktop).
- **Advanced Menu Management:** Capabilities for searching, filtering, and sorting food items based on various criteria (Cuisine, Price, Rating, Veg/Non-Veg).
- **Robust Cart & Checkout:** A persistent shopping cart using LocalStorage with a comprehensive checkout flow including address management and order summary.
- **Dual-Role Architecture:** Distinct interfaces for "Customers" (Ordering flow) and "Admins" (Management dashboard).
- **Admin Dashboard:** A powerful analytics hub providing real-time insights into sales, orders, and customer activity.
- **Security:** Implemented protected routes and role-based access control (RBAC) to ensure data integrity and authorized access.

The development process adhered to the Agile methodology, allowing for iterative improvements and rapid feedback integration. Extensive testing, including unit, integration, and user acceptance testing, has been conducted to ensure a bug-free and robust final product.

This report details every aspect of the project, from initial requirements gathering and system design to implementation details, testing strategies, and user manuals. It serves as comprehensive documentation for developers, stakeholders, and end-users alike.

<div style="page-break-after: always;"></div>

TABLE OF CONTENTS

<div style="page-break-after: always;"></div>

1. INTRODUCTION

1.1 Project Background

The food and beverage industry has seen a massive paradigm shift in the last decade. The traditional model of dining out or calling a restaurant for delivery is rapidly being supplemented, and in some cases replaced, by online ordering platforms. The convenience of browsing menus, customizing orders, and tracking deliveries from a smartphone has become an expectation rather than a luxury.

"The Food City" was conceived to address the need for a customizable, lightweight, and efficient food ordering solution for independent restaurants or food chains that wish to avoid the high commissions of aggregator platforms. By building a proprietary system, businesses can maintain direct relationships with their customers and retain control over their brand and data.

1.2 Problem Statement

Many small to medium-sized restaurants face significant challenges in the digital age:

1. **High Dependency on Aggregators:** Platforms like Swiggy and Zomato charge high commissions (up to 30%), significantly eating into profit margins.
2. **Lack of Customer Data:** Aggregators own the customer data. Restaurants often don't know who their loyal customers are.
3. **Limited Branding:** On third-party apps, a restaurant is just one of many cards in a list. There is limited scope for brand storytelling.
4. **Inefficient Phone Ordering:** Taking orders over the phone is prone to errors, time-consuming, and frustrating for customers during peak hours.

1.3 Objectives

The primary objectives of "The Food City" project are:

1. **To develop a user-friendly online ordering interface** that allows customers to browse menus and place orders intuitively.
2. **To implement a comprehensive administrative dashboard** for restaurant owners to manage menus, track orders, and view analytics.
3. **To ensure data persistence and state management** without a heavy backend initially, proving the viability of the frontend logic using LocalStorage and modern React patterns.
4. **To create a responsive design** that works flawlessly across desktops, tablets, and mobile phones, acknowledging the mobile-first nature of food ordering.
5. **To simulate real-world scenarios** such as authentication, role-based access control, and order lifecycle management.

1.4 Scope

In Scope:

- **Customer Module:** Registration, Login, Menu Browsing, Search/Filter, Cart Management, Checkout Process, Order History, User Profile.
- **Admin Module:** Admin Login, Dashboard (Analytics), Menu Management (Add/Edit/Delete/Toggle Availability), Order Management (Status Updates), Customer View.
- **System Features:** LocalStorage persistence, Toast notifications, Loading states, Error handling, Protected routes.

Out of Scope (Future Versions):

- **Live Payment Gateway Integration:** (Currently simulated).
- **Real-time GPS Tracking:** (Currently simulated with status updates).
- **Multi-vendor Support:** (Currently designed for a single entity).

1.5 Methodology

The project was executed using the **Agile functionality** with a focus on **Component-Driven Development (CDD)**.

1. **Requirement Gathering:** Analyzed competitors and interviewed potential users to list features.
2. **Design Phase:** created wireframes and defined the component hierarchy.
3. **Development:**
    - **Phase 1:** Core setup, routing, and basic layout.
    - **Phase 2:** Authentication and Context setup.
    - **Phase 3:** Customer modules (Menu, Cart).
    - **Phase 4:** Admin modules (Dashboard, Management).
4. **Testing:** Continuous unit testing of components and integration testing of flows.
5. **Deployment:** (Ready for deployment on Vercel/Netlify).

<div style="page-break-after: always;"></div>

2. SYSTEM ANALYSIS

2.1 Existing System

In the absence of this system, the business relies on:

- **Manual Phone Orders:** Staff writes down orders manually.
    - *Disadvantage:* High error rate, no visual menu for customers, waiting times on call.
- **Third-Party Aggregators:** Using Swiggy/Zomato.
    - *Disadvantage:* Loss of revenue due to commissions, loss of brand identity.
- **Walk-in Orders:** Physical presence required.
    - *Disadvantage:* Limited reach, customer inconvenience.

2.2 Proposed System

"The Food City" automates the entire flow:

- **Digital Menu:** Always up-to-date, visually appealing with images.
- **Automated Calculations:** Tax, delivery fees, and totals are calculated instantly, removing human error.
- **Order Tracking:** Customers see the status of their order in real-time (Pending -> Preparing -> Delivery -> Delivered).
- **Data Insights:** Owners get reports on "Top Selling Items" and "Peak Hours", allowing for better inventory planning.

2.3 Feasibility Study

A feasibility study is carried out to select the best system that meets performance requirements.

2.3.1 Technical Feasibility

The project uses standard web technologies (React, JavaScript, HTML5, CSS3) which are mature, well-documented, and supported by all modern browsers. The device requirements are minimal (any device with a browser), making it highly technically feasible.

2.3.2 Economic Feasibility

- **Development Cost:** Low (Open source libraries, free development tools like VS Code).
- **Maintenance Cost:** Low (Frontend hosting on platforms like Vercel is free for starter tiers).

- **ROI:** High, due to commission savings from third-party apps and increased operational efficiency.

2.3.3 Operational Feasibility

The User Interface is designed to be intuitive.

- **For Customers:** Similar to popular apps they already use; zero learning curve.
- **For Staff:** The Admin panel uses clear icons, color-coded statuses, and simple forms. Minimal training is required.

2.3.4 Legal Feasibility

The system handles user data (Names, Addresses, Phones). It is designed to be GDPR/Privacy-compliant by not storing sensitive payment data (simulated) and allowing users to manage their profiles.

2.4 Requirement Analysis

2.4.1 Functional Requirements

1. **Authentication:** Users must be able to Register and Login. System must distinguish between 'Admin' and 'Customer'.
2. **Menu Browsing:** Users should view items with images, prices, and descriptions.
3. **Search/Filter:** Users must be able to search for "Chicken" or filter by "Veg".
4. **Cart:** Users must be able to add items, change quantities, and remove items. Structure must persist on refresh.
5. **Order Placement:** Users must be able to input delivery details and confirm the order.
6. **Order Management:** Admins must be able to change order status.

2.4.2 Non-Functional Requirements

1. **Performance:** Page loads should be under 2 seconds.
2. **Reliability:** The system should not crash under normal load.
3. **Usability:** Mobile-responsive design is mandatory.
4. **Security:** Passwords must be validated (min length, complexity). API routes (simulated) must be protected.

<div style="page-break-after: always;"></div>

3. SYSTEM DESIGN

3.1 System Architecture

The application follows a **Component-Based Architecture** within a **Single Page Application (SPA)** framework.

- **Presentation Layer:** React Components (Pages, UI Elements). Handles user interaction and HTML rendering.
- **Logic Layer:** React Hooks (Custom Hooks, useEffect) & Context API. Handles business logic, state management, and side effects.
- **Data Layer:** LocalStorage Wrapper (Utils). Handles data persistence and retrieval, acting as a mock database.

3.2 Data Flow Diagrams (DFD)

Context Level (Level 0) DFD

- **User** -> [Requests/Orders] -> **System**
- **System** -> [Order Status/Menu] -> **User**
- **Admin** -> [Updates Menu/Status] -> **System**
- **System** -> [Reports/Orders] -> **Admin**

Level 1 DFD (Order Process)

1. **Customer** inputs login details -> **Auth Process** validates -> Token Granted.
2. **Customer** selects Items -> **Cart Process** updates State -> Storage updated.
3. **Customer** confirms Order -> **Order Process** creates Record -> **Order Database** (LocalStorage).
4. **Admin** views Order -> **Order Process** retrieves Record -> **Dashboard Display**.

3.3 Database Design (Mock Schema)

Although implemented in LocalStorage, the data is structured relationally.

Table 1: Users

Field	Type	Description
`id`	String (UUID)	Primary Key
`name`	String	Full Name
`email`	String	Unique Identifier
`password`	String	Hashed (Simulated)
`role`	Enum	'admin' or 'customer'
`addresses`	Array	List of address objects

Table 2: MenuItems

Field	Type	Description
`id`	String	Primary Key
`name`	String	Item Name
`price`	Number	Cost per unit
`category`	String	e.g., Indian, Chinese
`isAvailable`	Boolean	Stock status
`tags`	Array	Search keywords
`isVeg`	Boolean	Dietary classifier

Table 3: Orders

Field	Type	Description
`id`	String	Primary Key
`userId`	String	Foreign Key (Users)
`items`	Array	List of {itemId, qty, price}
`total`	Number	Financial total
`status`	Enum	pending, preparing, delivery, delivered
`timestamp`	Date	Order creation time

3.4 Algorithm Design

Cart Calculation Algorithm

```
FUNCTION calculateTotal(cartItems):
    subtotal = 0
    FOR EACH item IN cartItems:
        subtotal = subtotal + (item.price * item.quantity)
    deliveryFee = IF subtotal > 500 THEN 0 ELSE 40
    tax = subtotal * 0.05

    grandTotal = subtotal + deliveryFee + tax - discount
    RETURN grandTotal
END FUNCTION
```

Search Relevance Algorithm

The search feature uses a weighted scoring system (implicitly implemented via filtering):

1. Match in **Name** (High Priority).
2. Match in **Category** (Medium Priority).
3. Match in **Tags** (Low Priority).
4. Match in **Description**.

<div style="page-break-after: always;"></div>

4. IMPLEMENTATION DETAILS

4.1 Technology Stack

- **Frontend Framework:** React 18.x
  - *Why?* Virtual DOM for performance, huge ecosystem, component reusability.
- **Build Tool:** Vite
  - *Why?* Extremely fast HMR (Hot Module Replacement), optimized production builds.
- **Styling:** Tailwind CSS
  - *Why?* Utility-first approach allows for rapid UI development without context switching between CSS and JS files. Consistent design system.
- **Routing:** React Router DOM v6
  - *Why?* Standard solution for declarative routing in React SPAs.
- **Icons:** React Icons (Fa, Md, Bi)
  - *Why?* Lightweight SVG icons that are easily customizable via props.
- **State Management:** React Context API + useReducer
  - *Why?* sufficient for this scale, avoids boilerplates of Redux.

4.2 Module Description

Authentication Module (`src/context/AuthContext.jsx`)

This module handles the user identity. It exposes `login` , `register` , and `logout` functions. It uses `localStorage` to persist the session so users stay logged in on refresh.

- **Key Challenge:** Synchronizing state across multiple tabs.
- **Solution:** Using `window.addEventListener('storage')` to detect session changes.

Cart Module (`src/context/CartContext.jsx`)

This is the heart of the e-commerce functionality. It maintains the `items` array.

- **Features:** `addItem` (checks if item exists, increments qty if yes), `removeItem` , `updateQuantity` , `clearCart` .
- **Persistence:** Automatically saves to `localStorage` whenever the cart state changes via `useEffect` .

Admin Dashboard (`src/pages/admin/Dashboard.jsx`)

Visualizes data using charts and stats cards.

- **Implementation:** Calculates derived state from the raw `orders` array (e.g., `orders.reduce` for total revenue, `orders.filter` for today's orders).

4.3 Key Code Snippets Explanation

Protected Route Logic: This component acts as a gatekeeper. If a user tries to access `/admin` without the `admin` role, they are redirected.

```
 // Simplified logic
if (!isAuthenticated) return <Navigate to="/login" />;
if (requiredRole && user.role !== requiredRole) return <AccessDenied />;
return children;
```

<div style="page-break-after: always;"></div>

5. TESTING

5.1 Testing Strategy

We employed a "Black Box Testing" strategy where functionality was tested without looking at the internal code structure during the QA phase.

5.2 Test Cases

5.2.1 Authentication & User Profile

Test ID	Test Scenario	Steps	Expected Result	Status
TC_AUTH_01	Valid Login	1. Open Auth Modal<br>2. Enter valid email/pass<br>3. Click Login	Modal closes, Header shows User Name	PASS
TC_AUTH_02	Invalid Login	1. Enter wrong password<br>2. Click Login	Error message "Invalid credentials" appears	PASS
TC_AUTH_03	Registration	1. Switch to Sign Up<br>2. Enter details<br>3. Click Create Account	Account created, auto-logged in	PASS
TC_AUTH_04	Password Strength	1. Enter "123"<br>2. Submit	Validation error "Must be at least 8 chars..."	PASS

5.2.2 Cart Functionality

Test ID	Test Scenario	Steps	Expected Result	Status
TC_CART_01	Add Item	1. Click "Add" on Food Card	Item appears in cart, counter in header updates	PASS
TC_CART_02	Incr Quantity	1. Click "+" in Cart	Qty increases, Subtotal recalculates correcty	PASS
TC_CART_03	Remove Item	1. Click "Remove" icon	Item disappears from list	PASS
TC_CART_04	Empty Cart	1. Click "Clear Cart"	All items removed, "Cart Empty" message shown	PASS

5.2.3 Order Processing

Test ID	Test Scenario	Steps	Expected Result	Status
TC_ORD_01	Checkout Flow	1. Go to Cart<br>2. Click Checkout<br>3. Fill details<br>4. Pay	Order placed, redirect to Success Page	PASS
TC_ORD_02	Admin Update	1. Login as Admin<br>2. Go to Orders<br>3. Change status	Status updates in Admin and Customer View	PASS

<div style="page-break-after: always;"></div>

6. USER MANUAL

6.1 For Customers

Getting Started

1. **Launch the App:** Open the URL in your browser.
2. **Sign Up:** Click the "Sign Up" button in the top right. Fill in your Name, Email, Phone, and Password.
3. **Browse Menu:** Click "Menu" in the navigation bar. You can scroll through categories or use the search bar.

Placing an Order

1. **Select Items:** Click the "Add" button on any item you wish to eat.
2. **View Cart:** Click the Cart icon (top right) or "Cart" in the bottom nav (mobile).
3. **Checkout:** Review your items. Add any special instructions (e.g., "No onions"). Click "Proceed to Checkout".
4. **Payment:** Confirm your delivery address. Select Payment method (Cash or UPI). Click "Place Order".
5. **Track:** Go to "My Orders" in the profile menu to see the live status.

6.2 For Administrators

Accessing the Dashboard

1. **Login:** Use the dedicated Admin credentials (`admin@example.com` / `Admin123!`).
2. **Dashboard:** The landing page shows you today's snapshot: Revenue, Total Orders, and Pending Orders.

Managing the Menu

1. **Navigate:** Click "Menu Management" in the sidebar.
2. **Add Item:** Click the "+ Add New Item" button. Upload an image, set the price, and description.
3. **Toggle Availability:** If an ingredient is out of stock, click the "Toggle" switch on the item card to mark it as "Unavailable". The item will instantly disappear for customers.
4. **Delete:** Use the Trash icon to permanently remove an item.

Processing Orders

1. **Navigate:** Click "Orders" in the sidebar.
2. **View:** Filters are available for "Pending", "Preparing", etc.
3. **Update Status:**
    - When kitchen starts: Click "Mark as Preparing".
    - When food is ready: Click "Out for Delivery".
    - When driver returns: Click "Mark as Delivered".

<div style="page-break-after: always;"></div>

7. REQUIREMENTS TRACEABILITY MATRIX (RTM)

The RTM maps the user requirements to the specific design modules and test cases to ensure full coverage.

Req ID	Requirement Description	Module Implemented	Design Component	Test Case ID
R-01	User shall create an account	Auth Module	`AuthModal.jsx` , `AuthContext.jsx`	TC_AUTH_03
R-02	User shall login securely	Auth Module	`AuthModal.jsx` , `validation.js`	TC_AUTH_01
R-03	User shall view menu items	Customer Module	`MenuPage.jsx` , `FoodCard.jsx`	TC_MENU_01
R-04	User shall filter by Veg/NonVeg	Customer Module	`FilterPanel.jsx`	TC_MENU_02
R-05	System shall calculate totals	Cart Module	`CartContext.jsx`	TC_CART_02
R-06	Admin shall view sales stats	Admin Module	`Dashboard.jsx`	TC_ADM_01
R-07	Admin shall update order status	Admin Module	`OrderManagement.jsx`	TC_ORD_02

<div style="page-break-after: always;"></div>

8. CONCLUSION & FUTURE SCOPE

Conclusion "The Food City" successfully achieves its primary goals of creating a responsive, functional, and user-friendly food ordering system. By leveraging the power of React and LocalStorage, we have demonstrated that complex business logic can be handled efficiently on the client side for small to medium-scale applications. The project highlights the importance of clean code architecture, state management, and user-centric design.

Future Scope To make this a commercial-grade product, the following enhancements are planned:

1. **Backend Integration:** Moving from LocalStorage to a MongoDB/Node.js backend for scalable data storage.
2. **Payment Gateway:** Integrating Razorpay or Stripe for real money transactions.
3. **PWA Support:** Converting the web app into a Progressive Web App (PWA) for installability on mobile phones.
4. **Driver App:** A separate interface for delivery partners to accept orders and navigate using GPS.

<div style="page-break-after: always;"></div>

9. BIBLIOGRAPHY

1. **React Documentation:** https://react.dev/
2. **Tailwind CSS Documentation:** https://tailwindcss.com/docs
3. **MDN Web Docs:** For JavaScript and LocalStorage references.
4. **React Icons:** https://react-icons.github.io/react-icons/
5. **React Router:** https://reactrouter.com/

<div style="page-break-after: always;"></div>

10. APPENDIX A: SOURCE CODE

1. src/main.jsx

```
 import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

2. src/App.jsx

```
 import React from "react";
import { AuthProvider, useAuth } from "./context/AuthContext.jsx";
import { CartProvider } from "./context/CartContext.jsx";
import { ToastProvider } from "./components/shared/Toast.jsx";
import AppRouter from "./router/AppRouter.jsx";
import ErrorBoundary from "./components/shared/ErrorBoundary.jsx";
import ScrollToTop from "./components/shared/ScrollToTop.jsx";
import { FullScreenLoader } from "./components/shared/LoadingSpinner.jsx";

// App content component that waits for auth initialization
function AppContent() {
  const { isLoading } = useAuth();

  if (isLoading) {
    return <FullScreenLoader text="Initializing application..." />;
  }

  return (
    <CartProvider>
      <AppRouter />
      <ScrollToTop />
    </CartProvider>
  );
}

function App() {
  return (
    <ErrorBoundary>
      <ToastProvider>
        <AuthProvider>
          <AppContent />
        </AuthProvider>
      </ToastProvider>
    </ErrorBoundary>
  );
}

export default App;
```

3. src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Custom utilities */
@layer utilities {
  .scrollbar-hide {
    -ms-overflow-style: none;
    scrollbar-width: none;
  }

  .scrollbar-hide::-webkit-scrollbar {
    display: none;
  }

  .line-clamp-1 {
    display: -webkit-box;
    -webkit-line-clamp: 1;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }

  .line-clamp-2 {
    display: -webkit-box;
    -webkit-line-clamp: 2;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }

  .line-clamp-3 {
    display: -webkit-box;
    -webkit-line-clamp: 3;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }
}
```

4. src/context/AuthContext.jsx

```
import React, { createContext, useContext, useReducer, useEffect } from "react";
import { mockAPI } from "../services/mockApi.js";
import { storageManager } from "../utils/localStorage.js";

// Auth context
const AuthContext = createContext();

// Auth actions
const AUTH_ACTIONS = {
  LOGIN_START: "LOGIN_START",
  LOGIN_SUCCESS: "LOGIN_SUCCESS",
  LOGIN_FAILURE: "LOGIN_FAILURE",
  REGISTER_START: "REGISTER_START",
  REGISTER_SUCCESS: "REGISTER_SUCCESS",
  REGISTER_FAILURE: "REGISTER_FAILURE",
  LOGOUT: "LOGOUT",
  CLEAR_ERROR: "CLEAR_ERROR",
  SET_USER: "SET_USER",
  INIT_COMPLETE: "INIT_COMPLETE",
};

// Initial state
const initialState = {
  user: null,
  isAuthenticated: false,
  isLoading: true, // Start with loading true
  error: null,
};
```

```
// Auth reducer
```

```
function authReducer(state, action) {
  switch (action.type) {
    case AUTH_ACTIONS.LOGIN_START:
    case AUTH_ACTIONS.REGISTER_START:
      return {
        ...state,
        isLoading: true,
        error: null,
      };

    case AUTH_ACTIONS.LOGIN_SUCCESS:
    case AUTH_ACTIONS.REGISTER_SUCCESS:
      return {
        ...state,
        user: action.payload,
        isAuthenticated: true,
        isLoading: false,
        error: null,
      };

    case AUTH_ACTIONS.LOGIN_FAILURE:
    case AUTH_ACTIONS.REGISTER_FAILURE:
      return {
        ...state,
        user: null,
        isAuthenticated: false,
        isLoading: false,
        error: action.payload,
      };

    case AUTH_ACTIONS.LOGOUT:
      return {
        ...state,
        user: null,
        isAuthenticated: false,
        isLoading: false,
        error: null,
      };

    case AUTH_ACTIONS.SET_USER:
      return {
        ...state,
        user: action.payload,
        isAuthenticated: !!action.payload,
        isLoading: false,
      };

    case AUTH_ACTIONS.INIT_COMPLETE:
      return {
        ...state,
        isLoading: false,
      };

    case AUTH_ACTIONS.CLEAR_ERROR:
      return {
        ...state,
        error: null,
      };

    default:
      return state;
  }
}

// Auth provider component
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);

  // Check for existing session on mount
  useEffect(() => {
    const initializeAuth = () => {
      try {
        const savedUser = storageManager.getUserSession();
        if (savedUser) {
          dispatch({ type: AUTH_ACTIONS.SET_USER, payload: savedUser });
        } else {
          dispatch({ type: AUTH_ACTIONS.INIT_COMPLETE });
        }
      } catch (error) {
        console.error("Error initializing auth:", error);
        dispatch({ type: AUTH_ACTIONS.INIT_COMPLETE });
      }
    };

    // Add a small delay to ensure localStorage is ready
    setTimeout(initializeAuth, 100);
  }, []);

  // Login function
  const login = async (credentials) => {
    dispatch({ type: AUTH_ACTIONS.LOGIN_START });

    try {
      const response = await mockAPI.login(credentials);

      if (response.success) {
        storageManager.setUserSession(response.user);
        dispatch({ type: AUTH_ACTIONS.LOGIN_SUCCESS, payload: response.user });
        return { success: true, message: response.message };
      } else {
        dispatch({
          type: AUTH_ACTIONS.LOGIN_FAILURE,
          payload: response.message,
        });
        return { success: false, message: response.message };
      }
    } catch (error) {
      const errorMessage = "Login failed. Please try again.";
      dispatch({ type: AUTH_ACTIONS.LOGIN_FAILURE, payload: errorMessage });
      return { success: false, message: errorMessage };
    }
  };

  // Register function
  const register = async (userData) => {
    dispatch({ type: AUTH_ACTIONS.REGISTER_START });

    try {
      const response = await mockAPI.register(userData);

      if (response.success) {
        storageManager.setUserSession(response.user);
        dispatch({
          type: AUTH_ACTIONS.REGISTER_SUCCESS,
          payload: response.user,
        });
        return { success: true, message: response.message };
      } else {
        dispatch({
          type: AUTH_ACTIONS.REGISTER_FAILURE,
          payload: response.message,
        });
        return { success: false, message: response.message };
      }
    } catch (error) {
      const errorMessage = "Registration failed. Please try again.";
      dispatch({ type: AUTH_ACTIONS.REGISTER_FAILURE, payload: errorMessage });
      return { success: false, message: errorMessage };
    }
  };

  // Logout function
  const logout = async () => {
    try {
      await mockAPI.logout();
      dispatch({ type: AUTH_ACTIONS.LOGOUT });
      return { success: true };
    } catch (error) {
      console.error("Logout error:", error?.message || String(error));
      // Force logout even if API call fails
      dispatch({ type: AUTH_ACTIONS.LOGOUT });
      return { success: true };
    }
  };

  // Clear error function
  const clearError = () => {
    dispatch({ type: AUTH_ACTIONS.CLEAR_ERROR });
  };

  // Update user function
  const updateUser = (userData) => {
    const updatedUser = { ...state.user, ...userData };
    storageManager.setUserSession(updatedUser);
    dispatch({ type: AUTH_ACTIONS.SET_USER, payload: updatedUser });
  };

  // Check if user is admin
  const isAdmin = () => {
    return state.user?.role === "admin";
  };

  // Check if user is customer
  const isCustomer = () => {
    return state.user?.role === "customer";
  };

  const value = {
    ...state,
    login,
    register,
    logout,
    clearError,
    updateUser,
    isAdmin,
    isCustomer,
  };

  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
}

// Custom hook to use auth context
export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error("useAuth must be used within an AuthProvider");
  }
  return context;
}

// HOC for protected routes
export function withAuth(Component, requiredRole = null) {
  return function AuthenticatedComponent(props) {
    const { isAuthenticated, user, isLoading } = useAuth();

    if (isLoading) {
      return (
        <div className="min-h-screen flex items-center justify-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-dark-red"></div>
        </div>
      );
    }

    if (!isAuthenticated) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4">
            <h2 className="text-2xl font-bold text-dark-red text-center mb-4">
              Access Denied
            </h2>
            <p className="text-gray-600 text-center mb-6">
              Please log in to access this page.
          </p>
          <button
            onClick={() => (window.location.href = "/login")}
            className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-red trans
          >
            Go to Login
          </button>
        </div>
      </div>
    );
  }

  if (requiredRole && user?.role !== requiredRole) {
    return (
      <div className="min-h-screen flex items-center justify-center bg-light-gray">
        <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4">
          <h2 className="text-2xl font-bold text-error-red text-center mb-4">
            Unauthorized
          </h2>
          <p className="text-gray-600 text-center mb-6">
            You don't have permission to access this page.
          </p>
          <button
            onClick={() => window.history.back()}
            className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-600 transi
          >
            Go Back
          </button>
        </div>
      </div>
    );
  }

  return <Component {...props} />;
 };
}
```

5. src/context/CartContext.jsx

```
 import React, { createContext, useContext, useReducer, useEffect } from "react";
import { storageManager } from "../utils/localStorage.js";
import { vibrationUtils } from "../utils/vibration.js";

// Cart context
const CartContext = createContext();

// Cart actions
const CART_ACTIONS = {
  ADD_ITEM: "ADD_ITEM",
  REMOVE_ITEM: "REMOVE_ITEM",
  UPDATE_QUANTITY: "UPDATE_QUANTITY",
  CLEAR_CART: "CLEAR_CART",
  APPLY_PROMO: "APPLY_PROMO",
  REMOVE_PROMO: "REMOVE_PROMO",
  SET_DELIVERY_ADDRESS: "SET_DELIVERY_ADDRESS",
  SET_PAYMENT_METHOD: "SET_PAYMENT_METHOD",
  LOAD_CART: "LOAD_CART",
};

// Initial state
const initialState = {
  items: [],
  subtotal: 0,
  discount: 0,
  deliveryFee: 30,
  total: 0,
  promoCode: null,
  deliveryAddress: null,
  paymentMethod: "cod",
  itemCount: 0,
};

// Cart reducer
function cartReducer(state, action) {
  switch (action.type) {
    case CART_ACTIONS.ADD_ITEM: {
      const existingItemIndex = state.items.findIndex(
        (item) => item.id === action.payload.id
      );

      let newItems;
      if (existingItemIndex >= 0) {
        // Update existing item quantity
        newItems = state.items.map((item, index) =>
          index === existingItemIndex
            ? { ...item, quantity: item.quantity + 1 }
            : item
        );
      } else {
        // Add new item
        newItems = [...state.items, { ...action.payload, quantity: 1 }];
      }

      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.REMOVE_ITEM: {
      const newItems = state.items.filter((item) => item.id !== action.payload);
      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.UPDATE_QUANTITY: {
      const { itemId, quantity } = action.payload;

      if (quantity <= 0) {
        const newItems = state.items.filter((item) => item.id !== itemId);
        return calculateTotals({ ...state, items: newItems });
      }

      const newItems = state.items.map((item) =>
        item.id === itemId ? { ...item, quantity } : item
      );
      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.CLEAR_CART:
      return {
        ...initialState,
        deliveryAddress: state.deliveryAddress,
        paymentMethod: state.paymentMethod,
      };

    case CART_ACTIONS.APPLY_PROMO:
      return calculateTotals({
        ...state,
        promoCode: action.payload.promoCode,
        discount: action.payload.discount,
      });

    case CART_ACTIONS.REMOVE_PROMO:
      return calculateTotals({
        ...state,
        promoCode: null,
        discount: 0,
      });

    case CART_ACTIONS.SET_DELIVERY_ADDRESS:
      return { ...state, deliveryAddress: action.payload };

    case CART_ACTIONS.SET_PAYMENT_METHOD:
      return { ...state, paymentMethod: action.payload };

    case CART_ACTIONS.LOAD_CART:
      return calculateTotals(action.payload);
    default:
      return state;
  }
}

// Helper function to calculate totals
function calculateTotals(state) {
  // Ensure items is an array
  const items = Array.isArray(state.items) ? state.items : [];

  const subtotal = items.reduce((sum, item) => {
    const price = Number(item.price) || 0;
    const quantity = Number(item.quantity) || 0;
    return sum + price * quantity;
  }, 0);

  const itemCount = items.reduce((sum, item) => {
    const quantity = Number(item.quantity) || 0;
    return sum + quantity;
  }, 0);

  const discount = Number(state.discount) || 0;
  const deliveryFee = Number(state.deliveryFee) || 0;
  const total = subtotal - discount + deliveryFee;

  return {
    ...state,
    items,
    subtotal: Number(subtotal.toFixed(2)),
    total: Number(Math.max(0, total).toFixed(2)),
    itemCount,
    discount,
    deliveryFee,
  };
}

// Cart provider component
export function CartProvider({ children }) {
  const [state, dispatch] = useReducer(cartReducer, initialState);

  // Load cart from localStorage on mount
  useEffect(() => {
    const savedCart = storageManager.getCartData();
    if (savedCart && savedCart.items && savedCart.items.length > 0) {
      dispatch({ type: CART_ACTIONS.LOAD_CART, payload: savedCart });
    }
  }, []);

  // Save cart to localStorage whenever state changes
  useEffect(() => {
    storageManager.setCartData(state);
  }, [state]);

  // Add item to cart
  const addItem = (item) => {
    dispatch({ type: CART_ACTIONS.ADD_ITEM, payload: item });

    // Trigger vibration on mobile for haptic feedback
    vibrationUtils.addToCart();
  };

  // Remove item from cart
  const removeItem = (itemId) => {
    dispatch({ type: CART_ACTIONS.REMOVE_ITEM, payload: itemId });

    // Light vibration for remove action
    vibrationUtils.light();
  };

  // Update item quantity
  const updateQuantity = (itemId, quantity) => {
    dispatch({
      type: CART_ACTIONS.UPDATE_QUANTITY,
      payload: { itemId, quantity },
    });
  };

  // Clear entire cart
  const clearCart = () => {
    dispatch({ type: CART_ACTIONS.CLEAR_CART });
  };

  // Apply promo code
  const applyPromo = (promoCode, discount) => {
    dispatch({
      type: CART_ACTIONS.APPLY_PROMO,
      payload: { promoCode, discount },
    });

    // Success vibration for promo code applied
    vibrationUtils.success();
  };

  // Remove promo code
  const removePromo = () => {
    dispatch({ type: CART_ACTIONS.REMOVE_PROMO });
  };

  // Set delivery address
  const setDeliveryAddress = (address) => {
    dispatch({ type: CART_ACTIONS.SET_DELIVERY_ADDRESS, payload: address });
  };

  // Set payment method
  const setPaymentMethod = (method) => {
    dispatch({ type: CART_ACTIONS.SET_PAYMENT_METHOD, payload: method });
  };

  // Get item quantity in cart
  const getItemQuantity = (itemId) => {
    const item = state.items.find((item) => item.id === itemId);
    return item ? item.quantity : 0;
  };

  // Check if item is in cart
  const isItemInCart = (itemId) => {
    return state.items.some((item) => item.id === itemId);
  };

  // Get cart summary for order
  const getOrderSummary = () => {
    return {
      items: state.items.map((item) => ({
        menuItemId: item.id,
        name: item.name,
        price: item.price,
        quantity: item.quantity,
        specialInstructions: item.specialInstructions || "",
      })),
      subtotal: state.subtotal,
      discount: state.discount,
      deliveryFee: state.deliveryFee,
      total: state.total,
      promoCode: state.promoCode?.code || "",
      deliveryAddress: state.deliveryAddress,
      paymentMethod: state.paymentMethod,
    };
  };

  const value = {
    ...state,
    addItem,
    removeItem,
    updateQuantity,
    clearCart,
    applyPromo,
    removePromo,
    setDeliveryAddress,
    setPaymentMethod,
    getItemQuantity,
    isItemInCart,
    getOrderSummary,
  };

  return <CartContext.Provider value={value}>{children}</CartContext.Provider>;
}

// Custom hook to use cart context
export function useCart() {
  const context = useContext(CartContext);
  if (!context) {
    throw new Error("useCart must be used within a CartProvider");
  }
  return context;
}
```

6. src/components/auth/AuthModal.jsx

```
 import React, { useState, useEffect, useRef } from "react";
import { useAuth } from "../../context/AuthContext.jsx";

export default function AuthModal({ isOpen, onClose, initialMode = "login" }) {
  const [mode, setMode] = useState(initialMode);
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
    phone: "",
  });
  const [showPassword, setShowPassword] = useState(false);
  const clearErrorRef = useRef();

  const { login, register, isLoading, error, clearError } = useAuth();

  // Store clearError in ref to avoid dependency issues
  clearErrorRef.current = clearError;

  // Sync mode with initialMode prop when it changes
  useEffect(() => {
    setMode(initialMode);
  }, [initialMode]);

  // Clear form data and errors when modal opens
  useEffect(() => {
    if (isOpen) {
      setFormData({
        name: "",
        email: "",
        password: "",
        phone: "",
      });
      clearErrorRef.current();
    }
  }, [isOpen]); // Removed clearError from dependencies

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData((prev) => ({
      ...prev,
      [name]: value,
    }));

    // Clear error when user starts typing
    if (error) {
      clearErrorRef.current();
    }
  };
  const handleSubmit = async (e) => {
    e.preventDefault();

    let result;
    if (mode === "login") {
      result = await login({
        email: formData.email,
        password: formData.password,
      });
    } else {
      result = await register({
        name: formData.name,
        email: formData.email,
        password: formData.password,
        phone: formData.phone,
      });
    }

    if (result.success) {
      onClose();
      // Reset form
      setFormData({
        name: "",
        email: "",
        password: "",
        phone: "",
      });
    }
  };

  const switchMode = () => {
    setMode(mode === "login" ? "register" : "login");
    clearErrorRef.current();
    setFormData({
      name: "",
      email: "",
      password: "",
      phone: "",
    });
  };

  const fillDemoCredentials = (role) => {
    if (role === "customer") {
      setFormData({
        ...formData,
        email: "mohit.kumar@example.com",
        password: "Password123!",
      });
    } else if (role === "admin") {
      setFormData({
        ...formData,
        email: "admin@example.com",
        password: "Admin123!",
      });
    }
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">
      <div className="bg-white rounded-lg shadow-lg max-w-md w-full max-h-[90vh] overflow-y-auto">
        <div className="p-6">
          {/* Header */}
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-2xl font-bold text-dark-red">
              {mode === "login" ? "Welcome Back" : "Create Account"}
            </h2>
            <button
              onClick={onClose}
              className="text-gray-400 hover:text-gray-600 text-2xl"
            >
              ×
            </button>
          </div>

          {/* Demo Credentials */}
          <div className="mb-6 p-4 bg-light-gray rounded-lg">
            <p className="text-sm text-gray-600 mb-2">Demo Credentials:</p>
            <div className="flex gap-2">
              <button
                type="button"
                onClick={() => fillDemoCredentials("customer")}
                className="text-xs bg-info-blue text-white px-2 py-1 rounded hover:bg-blue-600"
              >
                Customer
              </button>
              <button
                type="button"
                onClick={() => fillDemoCredentials("admin")}
                className="text-xs bg-spicy-orange text-white px-2 py-1 rounded hover:bg-orange-600"
              >
                Admin
              </button>
            </div>
          </div>

          {/* Error Message */}
          {error && (
            <div className="mb-4 p-3 bg-red-100 border border-red-400 text-red-700 rounded">
              {error}
            </div>
          )}

          {/* Form */}
          <form onSubmit={handleSubmit} className="space-y-4">
            {mode === "register" && (
              <div>
                <label className="block text-sm font-medium text-gray-700 mb-1">
                  Full Name
                </label>
                <input
                  type="text"
                  name="name"
                  value={formData.name}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none f
                  placeholder="Enter your full name"
                />
              </div>
            )}

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-1">
                Email Address
              </label>
              <input
                type="email"
                name="email"
                value={formData.email}
                onChange={handleInputChange}
                required
                className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none foc
                placeholder="Enter your email"
              />
            </div>

            {mode === "register" && (
              <div>
                <label className="block text-sm font-medium text-gray-700 mb-1">
                  Phone Number
                </label>
                <input
                  type="tel"
                  name="phone"
                  value={formData.phone}
```

```
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none f
                  placeholder="Enter your phone number"
                />
              </div>
            )}

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-1">
                Password
              </label>
              <div className="relative">
                <input
                  type={showPassword ? "text" : "password"}
                  name="password"
                  value={formData.password}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none f
                  placeholder="Enter your password"
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-400 hover
                >
                  {showPassword ? "👁️" : "👁️‍🗨️"}
                </button>
              </div>
            </div>

            <button
              type="submit"
              disabled={isLoading}
              className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-red trans
            >
              {isLoading ? (
                <div className="flex items-center justify-center">
                  <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-white mr-2"></
                  {mode === "login" ? "Signing In..." : "Creating Account..."}
                </div>
              ) : mode === "login" ? (
                "Sign In"
              ) : (
                "Create Account"
              )}
            </button>
          </form>

          {/* Switch Mode */}
          <div className="mt-6 text-center">
            <p className="text-gray-600">
              {mode === "login"
                ? "Don't have an account? "
                : "Already have an account? "}
              <button
                onClick={switchMode}
                className="text-dark-red hover:text-hover-red font-medium"
              >
                {mode === "login" ? "Sign Up" : "Sign In"}
              </button>
            </p>
          </div>
        </div>
      </div>
    </div>
  );
}
```

7. src/utils/localStorage.js

```
 // Local storage utilities for The Food City

const STORAGE_KEYS = {
  USER_SESSION: "foodcity_user_session",
  CART_DATA: "foodcity_cart",
  MENU_ITEMS: "foodcity_menu_items",
  ORDERS: "foodcity_orders",
  PROMO_CODES: "foodcity_promo_codes",
  DELIVERY_BOYS: "foodcity_delivery_boys",
  REVIEWS: "foodcity_reviews",
  USERS: "foodcity_users",
  ANALYTICS: "foodcity_analytics",
};

class LocalStorageManager {
  // Generic methods
  setItem(key, value) {
    try {
      localStorage.setItem(key, JSON.stringify(value));
      return true;
    } catch (error) {
      console.error(
        "Error saving to localStorage:",
        error?.message || String(error)
      );
      return false;
    }
  }

  getItem(key) {
    try {
      const item = localStorage.getItem(key);
      return item ? JSON.parse(item) : null;
    } catch (error) {
      console.error(
        "Error reading from localStorage:",
        error?.message || String(error)
      );
      return null;
    }
  }

  removeItem(key) {
    try {
      localStorage.removeItem(key);
      return true;
    } catch (error) {
      console.error(
        "Error removing from localStorage:",
        error?.message || String(error)
      );
      return false;
    }
  }

  // User session management
  setUserSession(user) {
    return this.setItem(STORAGE_KEYS.USER_SESSION, user);
  }

  getUserSession() {
    return this.getItem(STORAGE_KEYS.USER_SESSION);
  }

  clearUserSession() {
    return this.removeItem(STORAGE_KEYS.USER_SESSION);
  }

  // Cart management
  setCartData(cartData) {
    return this.setItem(STORAGE_KEYS.CART_DATA, cartData);
  }
  getCartData() {
    return this.getItem(STORAGE_KEYS.CART_DATA) || { items: [], total: 0 };
  }

  clearCartData() {
    return this.removeItem(STORAGE_KEYS.CART_DATA);
  }

  // Menu items management
  setMenuItems(menuItems) {
    return this.setItem(STORAGE_KEYS.MENU_ITEMS, menuItems);
  }

  getMenuItems() {
    return this.getItem(STORAGE_KEYS.MENU_ITEMS) || [];
  }

  // Orders management
  setOrders(orders) {
    return this.setItem(STORAGE_KEYS.ORDERS, orders);
  }

  getOrders() {
    return this.getItem(STORAGE_KEYS.ORDERS) || [];
  }

  addOrder(order) {
    const orders = this.getOrders();
    orders.push(order);
    return this.setOrders(orders);
  }

  updateOrder(orderId, updates) {
    const orders = this.getOrders();
    const orderIndex = orders.findIndex((order) => order.id === orderId);
    if (orderIndex !== -1) {
      orders[orderIndex] = { ...orders[orderIndex], ...updates };
      return this.setOrders(orders);
    }
    return false;
  }

  // Promo codes management
  setPromoCodes(promoCodes) {
    return this.setItem(STORAGE_KEYS.PROMO_CODES, promoCodes);
  }

  getPromoCodes() {
    return this.getItem(STORAGE_KEYS.PROMO_CODES) || [];
  }

  // Delivery boys management
  setDeliveryBoys(deliveryBoys) {
    return this.setItem(STORAGE_KEYS.DELIVERY_BOYS, deliveryBoys);
  }

  getDeliveryBoys() {
    return this.getItem(STORAGE_KEYS.DELIVERY_BOYS) || [];
  }

  // Reviews management
  setReviews(reviews) {
    return this.setItem(STORAGE_KEYS.REVIEWS, reviews);
  }

  getReviews() {
    return this.getItem(STORAGE_KEYS.REVIEWS) || [];
  }

  addReview(review) {
    const reviews = this.getReviews();
    reviews.push(review);
    return this.setReviews(reviews);
  }
  // Users management
  setUsers(users) {
    return this.setItem(STORAGE_KEYS.USERS, users);
  }

  getUsers() {
    return this.getItem(STORAGE_KEYS.USERS) || [];
  }

  // Analytics management
  setAnalytics(analytics) {
    return this.setItem(STORAGE_KEYS.ANALYTICS, analytics);
  }

  getAnalytics() {
    return this.getItem(STORAGE_KEYS.ANALYTICS) || {};
  }

  // Initialize with default data
  initializeDefaultData(defaultData) {
    // Always update menu items to get latest additions
    this.setMenuItems(defaultData.menuItems);

    // Check if users contain old data (John Doe) or missing new data (Mohit Kumar) and force update
    const currentUsers = this.getUsers();
    const hasOldData = currentUsers.some(
      (u) => u.name === "John Doe" || u.name === "Jane Smith"
    );
    const missingNewData = !currentUsers.some((u) => u.name === "Mohit Kumar");

    if (!currentUsers.length || hasOldData || missingNewData) {
      this.setUsers(defaultData.users);
      // Also reset orders if we reset users to avoid ID mismatches
      this.setOrders(defaultData.orders);
    }

    // Check for expired promo codes and force update
    const currentPromos = this.getPromoCodes();
    const hasExpiredPromos = currentPromos.some(
      (p) => new Date(p.expiryDate) < new Date()
    );

    if (!currentPromos.length || hasExpiredPromos) {
      this.setPromoCodes(defaultData.promoCodes);
    }
    if (!this.getDeliveryBoys().length) {
      this.setDeliveryBoys(defaultData.deliveryBoys);
    }

    // Check analytics for old data
    const analytics = this.getAnalytics();
    const hasOldAnalytics =
      analytics.topCustomers &&
      analytics.topCustomers.some((c) => c.name === "John Doe");

    if (!Object.keys(analytics).length || hasOldAnalytics) {
      this.setAnalytics(defaultData.analytics);
    }

    if (!this.getReviews().length) {
      this.setReviews(defaultData.reviews);
    }
  }

  // Clear all data (for testing/reset)
  clearAllData() {
    Object.values(STORAGE_KEYS).forEach((key) => {
      this.removeItem(key);
    });
  }
}

export const storageManager = new LocalStorageManager();
export { STORAGE_KEYS };
```

8. src/data/mockData.js

```
 // Mock data for The Food City platform

export const mockUsers = [
  {
    id: "1",
    name: "Mohit Kumar",
    email: "mohit.kumar@example.com",
    password: "Password123!",
    phone: "+919876543210",
    role: "customer",
    addresses: [
      {
        id: "1",
        userId: "1",
        label: "Home",
        street: "Flat 402, Sunshine Apartments, Andheri West",
        city: "Mumbai",
        state: "Maharashtra",
        pinCode: "400001",
        landmark: "Near Central Mall",
        isDefault: true,
      },
    ],
    createdAt: new Date("2024-01-15"),
    isActive: true,
  },
  {
    id: "2",
    name: "Fatima Shaikh",
    email: "fatima.shaikh@example.com",
    password: "Password123!",
    phone: "+919876543211",
    role: "customer",
    addresses: [
      {
        id: "2",
        userId: "2",
        label: "Home",
        street: "Block C, Lajpat Nagar",
        city: "Delhi",
        state: "Delhi",
        pinCode: "110001",
        landmark: "Opposite Metro Station",
        isDefault: true,
      },
    ],
    createdAt: new Date("2024-02-10"),
    isActive: true,
  },
  {
    id: "admin1",
    name: "Admin User",
    email: "admin@example.com",
    password: "Admin123!",
    phone: "+919876543212",
    role: "admin",
    addresses: [],
    createdAt: new Date("2024-01-01"),
    isActive: true,
  },
];



export const mockMenuItems = [
  // Indian Category
  {
    id: "1",
    name: "Butter Chicken",
    description: "Creamy tomato-based curry with tender chicken pieces",
    price: 250,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1603894584373-5ac82b2ae398?w=400",
    rating: 4.5,
    reviewCount: 128,
    isAvailable: true,
    preparationTime: 25,
    tags: ["spicy", "creamy", "popular"],
  },
  {
    id: "2",
    name: "Paneer Tikka",
    description: "Grilled cottage cheese marinated in aromatic spices",
    price: 200,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=400",
    rating: 4.3,
    reviewCount: 95,
    isAvailable: true,
    preparationTime: 20,
    tags: ["grilled", "spicy", "vegetarian"],
  },
  {
    id: "3",
    name: "Dal Makhani",
    description: "Rich and creamy black lentils cooked overnight",
    price: 180,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1546833999-b9f581a1996d?w=400",
    rating: 4.4,
    reviewCount: 87,
    isAvailable: true,
    preparationTime: 15,
    tags: ["creamy", "comfort food", "vegetarian"],
  },
  {
    id: "13",
    name: "Chicken Biryani",
    description: "Aromatic basmati rice cooked with spiced chicken and saffron",
    price: 320,
    category: "Indian",
    type: "Non-Veg",
    image: "https://www.licious.in/blog/wp-content/uploads/2022/06/chicken-hyderabadi-biryani-01.jpg
    rating: 4.8,
    reviewCount: 245,
    isAvailable: true,
    preparationTime: 45,
    tags: ["aromatic", "rice", "festive"],
  },
  {
    id: "14",
    name: "Palak Paneer",
    description: "Cottage cheese cubes in creamy spinach gravy",
    price: 210,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1601050690597-df0568f70950?w=400",
    rating: 4.5,
    reviewCount: 134,
    isAvailable: true,
    preparationTime: 20,
    tags: ["creamy", "healthy", "vegetarian"],
  },
  {
    id: "15",
    name: "Mutton Rogan Josh",
    description: "Tender mutton cooked in aromatic Kashmiri spices",
    price: 380,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1585937421612-70a008356fbe?w=400",
    rating: 4.6,
    reviewCount: 98,
    isAvailable: true,
    preparationTime: 50,
    tags: ["aromatic", "tender", "kashmiri"],
  },
  {
    id: "16",
    name: "Chole Bhature",
    description: "Spicy chickpea curry served with fluffy fried bread",
    price: 160,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1606491956689-2ea866880c84?w=400",
    rating: 4.3,
    reviewCount: 167,
    isAvailable: true,
    preparationTime: 25,
    tags: ["spicy", "traditional", "filling"],
  },
  {
    id: "17",
    name: "Fish Curry",
    description: "Fresh fish cooked in coconut-based curry",
    price: 290,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1631452180519-c014fe946bc7?w=400",
    rating: 4.4,
    reviewCount: 76,
    isAvailable: true,
    preparationTime: 30,
    tags: ["coconut", "fresh", "coastal"],
  },

  // Chinese Category
  {
    id: "4",
    name: "Chicken Fried Rice",
    description: "Wok-tossed rice with chicken and vegetables",
    price: 180,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1512058564366-18510be2db19?w=400",
    rating: 4.2,
    reviewCount: 156,
    isAvailable: true,
    preparationTime: 18,
    tags: ["fried", "savory", "filling"],
  },
  {
    id: "5",
    name: "Veg Manchurian",
    description: "Deep-fried vegetable balls in tangy sauce",
    price: 150,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1626645738196-c2a7c87a8f58?w=400",
    rating: 4.1,
    reviewCount: 73,
    isAvailable: true,
    preparationTime: 22,
    tags: ["tangy", "crispy", "vegetarian"],
  },
  {
    id: "6",
    name: "Hakka Noodles",
    description: "Stir-fried noodles with vegetables and sauces",
    price: 160,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1585032226651-759b368d7246?w=400",
    rating: 4.0,
    reviewCount: 92,
    isAvailable: true,
    preparationTime: 16,
    tags: ["stir-fried", "savory", "noodles"],
  },
  {
    id: "18",
    name: "Sweet and Sour Chicken",
    description: "Crispy chicken in tangy sweet and sour sauce",
    price: 220,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1559847844-5315695dadae?w=400",
    rating: 4.3,
    reviewCount: 89,
    isAvailable: true,
    preparationTime: 20,
    tags: ["crispy", "tangy", "sweet"],
  },
  {
    id: "19",
    name: "Chilli Garlic Fried Rice",
    description: "Spicy fried rice with garlic and vegetables",
    price: 170,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1512058564366-18510be2db19?w=400",
    rating: 4.1,
    reviewCount: 112,
    isAvailable: true,
    preparationTime: 15,
    tags: ["spicy", "garlic", "rice"],
  },
  {
    id: "20",
    name: "Honey Chilli Potato",
    description: "Crispy potato fingers tossed in honey chilli sauce",
    price: 140,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1598103442097-8b74394b95c6?w=400",
    rating: 4.2,
    reviewCount: 156,
    isAvailable: true,
    preparationTime: 18,
    tags: ["crispy", "honey", "appetizer"],
  },
  {
    id: "21",
    name: "Chicken Chowmein",
    description: "Stir-fried noodles with chicken and mixed vegetables",
    price: 190,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1585032226651-759b368d7246?w=400",
    rating: 4.4,
    reviewCount: 145,
    isAvailable: true,
    preparationTime: 22,
    tags: ["stir-fried", "noodles", "protein"],
  },
  {
    id: "22",
    name: "Dragon Chicken",
    description: "Spicy Indo-Chinese chicken with bell peppers",
    price: 250,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1559847844-5315695dadae?w=400",
    rating: 4.5,
    reviewCount: 103,
    isAvailable: true,
    preparationTime: 25,
    tags: ["spicy", "indo-chinese", "bell peppers"],
  },

  // South Category
  {
    id: "7",
    name: "Masala Dosa",
    description: "Crispy rice crepe filled with spiced potato curry",
    price: 120,
    category: "South",
    type: "Veg",
    image: "https://vismaifood.com/storage/app/uploads/public/45a/29b/a17/thumb__700_0_0_0_auto.jpg"
    rating: 4.6,
    reviewCount: 203,
    isAvailable: true,
    preparationTime: 20,
    tags: ["crispy", "traditional", "vegetarian"],
  },
  {
    id: "8",
    name: "Chicken Chettinad",
    description: "Spicy South Indian chicken curry with aromatic spices",
    price: 220,
    category: "South",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1565557623262-b51c2513a641?w=400",
    rating: 4.4,
    reviewCount: 67,
    isAvailable: true,
    preparationTime: 30,
    tags: ["spicy", "traditional", "aromatic"],
  },
  {
    id: "9",
    name: "Sambar Rice",
    description: "Steamed rice served with lentil-based vegetable stew",
    price: 140,
    category: "South",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1596797038530-2c107229654b?w=400",
    rating: 4.2,
    reviewCount: 45,
    isAvailable: true,
    preparationTime: 15,
    tags: ["healthy", "comfort food", "vegetarian"],
  },
  {
    id: "23",
    name: "Idli Sambar",
    description:
      "Steamed rice cakes served with lentil curry and coconut chutney",
    price: 100,
    category: "South",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1589301760014-d929f3979dbc?w=400",
    rating: 4.5,
    reviewCount: 189,
    isAvailable: true,
    preparationTime: 12,
    tags: ["healthy", "steamed", "traditional"],
  },
  {
    id: "24",
    name: "Medu Vada",
    description: "Crispy lentil donuts served with sambar and chutney",
    price: 80,
    category: "South",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1589301760014-d929f3979dbc?w=400",
    rating: 4.3,
    reviewCount: 234,
    isAvailable: true,
    preparationTime: 15,
    tags: ["crispy", "lentil", "snack"],
  },
  {
    id: "25",
    name: "Rava Upma",
    description: "Savory semolina porridge with vegetables and spices",
    price: 90,
    category: "South",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1606491956689-2ea866880c84?w=400",
    rating: 4.0,
    reviewCount: 67,
    isAvailable: true,
    preparationTime: 10,
    tags: ["savory", "healthy", "breakfast"],
  },
  {
    id: "26",
    name: "Chicken 65",
    description: "Spicy deep-fried chicken with curry leaves and chilies",
    price: 260,
    category: "South",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1565557623262-b51c2513a641?w=400",
    rating: 4.6,
    reviewCount: 178,
    isAvailable: true,
    preparationTime: 20,
    tags: ["spicy", "fried", "appetizer"],
  },
  {
    id: "27",
    name: "Coconut Rice",
    description: "Fragrant rice cooked with coconut and South Indian spices",
    price: 130,
    category: "South",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1596797038530-2c107229654b?w=400",
    rating: 4.2,
    reviewCount: 92,
    isAvailable: true,
    preparationTime: 18,
    tags: ["coconut", "fragrant", "rice"],
  },

  // Tandoor Category
  {
    id: "10",
    name: "Chicken Tikka",
    description: "Marinated chicken pieces grilled in tandoor oven",
    price: 240,
    category: "Tandoor",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
    rating: 4.7,
    reviewCount: 189,
    isAvailable: true,
    preparationTime: 25,
    tags: ["grilled", "smoky", "protein-rich"],
  },
  {
    id: "11",
    name: "Naan",
    description: "Soft leavened bread baked in tandoor",
    price: 40,
    category: "Tandoor",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1586190848861-99aa4a171e90?w=400",
    rating: 4.3,
    reviewCount: 234,
    isAvailable: true,
    preparationTime: 10,
    tags: ["bread", "soft", "traditional"],
  },
  {
    id: "12",
    name: "Seekh Kebab",
    description: "Spiced minced meat skewers grilled to perfection",
    price: 280,
    category: "Tandoor",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
    rating: 4.5,
    reviewCount: 112,
    isAvailable: true,
    preparationTime: 22,
    tags: ["grilled", "spiced", "kebab"],
  },
  {
    id: "28",
    name: "Tandoori Chicken",
    description:
      "Whole chicken marinated in yogurt and spices, roasted in tandoor",
    price: 420,
    category: "Tandoor",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
    rating: 4.8,
    reviewCount: 201,
    isAvailable: true,
    preparationTime: 35,
    tags: ["grilled", "yogurt", "traditional"],
  },
  {
    id: "29",
    name: "Paneer Tikka",
    description: "Marinated cottage cheese cubes grilled with vegetables",
    price: 220,
    category: "Tandoor",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=400",
    rating: 4.4,
    reviewCount: 145,
    isAvailable: true,
    preparationTime: 18,
    tags: ["grilled", "vegetarian", "cottage cheese"],
  },
  {
    id: "30",
    name: "Garlic Naan",
    description: "Soft bread topped with garlic and herbs, baked in tandoor",
    price: 60,
    category: "Tandoor",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1586190848861-99aa4a171e90?w=400",
    rating: 4.6,
    reviewCount: 298,
    isAvailable: true,
    preparationTime: 12,
    tags: ["bread", "garlic", "herbs"],
  },
  {
    id: "31",
    name: "Tandoori Fish",
    description: "Fresh fish marinated in tandoori spices and grilled",
    price: 320,
    category: "Tandoor",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1631452180519-c014fe946bc7?w=400",
    rating: 4.5,
    reviewCount: 87,
    isAvailable: true,
    preparationTime: 25,
    tags: ["grilled", "fish", "spiced"],
  },
  {
    id: "32",
    name: "Reshmi Kebab",
    description: "Creamy chicken kebabs made with cashews and cream",
    price: 300,
    category: "Tandoor",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
    rating: 4.7,
    reviewCount: 156,
    isAvailable: true,
    preparationTime: 28,
    tags: ["creamy", "cashews", "kebab"],
  },
];

export const mockPromoCodes = [
  {
    id: "1",
    code: "FIRST50",
    description: "50% off on your first order",
    discountType: "percentage",
    discountValue: 50,
    minOrderValue: 200,
    maxDiscount: 150,
    expiryDate: new Date("2030-12-31"),
    usageLimit: 1000,
    usedCount: 245,
    isActive: true,
  },
  {
    id: "2",
    code: "WELCOME20",
    description: "20% off for new customers",
    discountType: "percentage",
    discountValue: 20,
    minOrderValue: 150,
    maxDiscount: 100,
    expiryDate: new Date("2030-11-30"),
    usageLimit: 500,
    usedCount: 123,
    isActive: true,
  },
  {
    id: "3",
    code: "FLAT100",
    description: "Flat ₹100 off on orders above ₹500",
    discountType: "fixed",
    discountValue: 100,
    minOrderValue: 500,
    maxDiscount: 100,
    expiryDate: new Date("2030-10-15"),
    usageLimit: 200,
    usedCount: 67,
    isActive: true,
  },
];

export const mockDeliveryBoys = [
  {
    id: "1",
    name: "Amit Sharma",
    phone: "+919876543210",
    isAvailable: true,
    currentOrders: [],
    rating: 4.8,
    totalDeliveries: 1247,
  },
  {
    id: "2",
    name: "Rajesh Kumar",
    phone: "+919876543211",
    isAvailable: true,
    currentOrders: [],
    rating: 4.6,
    totalDeliveries: 892,
  },
  {
    id: "3",
    name: "Suresh Patel",
    phone: "+919876543212",
    isAvailable: false,
    currentOrders: ["order_1"],
    rating: 4.7,
    totalDeliveries: 1056,
  },
];

export const mockOrders = [
  {
    id: "order_1",
    customerId: "1",
    items: [
      {
        menuItemId: "1",
        name: "Butter Chicken",
        price: 250,
        quantity: 1,
        specialInstructions: "Medium spicy",
      },
      {
        menuItemId: "11",
        name: "Naan",
        price: 40,
        quantity: 2,
        specialInstructions: "",
      },
    ],
    subtotal: 330,
    discount: 66,
    deliveryFee: 30,
    total: 294,
    status: "delivered",
    deliveryAddress: {
      id: "1",
      userId: "1",
      label: "Home",
      street: "123 Main St",
      city: "Mumbai",
      state: "Maharashtra",
      pinCode: "400001",
      landmark: "Near Central Mall",
      isDefault: true,
    },
    paymentMethod: "razorpay",
    promoCode: "WELCOME20",
    deliveryBoyId: "3",
    estimatedDeliveryTime: new Date("2024-07-18T13:30:00"),
    createdAt: new Date("2024-07-18T12:15:00"),
    updatedAt: new Date("2024-07-18T13:25:00"),
  },
];

export const mockReviews = [
  {
    id: "1",
    orderId: "order_1",
    customerId: "1",
    menuItemId: "1",
    rating: 5,
    comment:
      "Absolutely delicious! The butter chicken was perfectly creamy and flavorful.",
    createdAt: new Date("2024-07-18T14:00:00"),
    isModerated: false,
  },
];

// Analytics mock data
export const mockAnalytics = {
  totalViews: 15420,
  totalRevenue: 125000,
  totalOrders: 1247,
  averageRating: 4.3,
  todaysSales: 8500,
  todaysOrders: 45,
  cancelledOrders: 23,
  topCustomers: [
    { name: "Mohit Kumar", orders: 15, spent: 3200 },
    { name: "Fatima Shaikh", orders: 12, spent: 2800 },
    { name: "Sadat Alam", orders: 10, spent: 2400 },
  ],
  topItems: [
    { name: "Butter Chicken", orders: 89, revenue: 22250 },
    { name: "Masala Dosa", orders: 76, revenue: 9120 },
    { name: "Chicken Tikka", orders: 65, revenue: 15600 },
  ],
  revenueByTime: [
    { time: "9 AM", revenue: 1200 },
    { time: "12 PM", revenue: 3500 },
    { time: "3 PM", revenue: 2100 },
    { time: "6 PM", revenue: 4200 },
    { time: "9 PM", revenue: 3800 },
  ],
};
```

9. src/pages/admin/Dashboard.jsx

```
 import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import { Card } from "../../components/shared/Layout.jsx";
import {
  FaShoppingBag,
  FaRupeeSign,
  FaCalendarAlt,
  FaChartLine,
  FaArrowUp,
  FaArrowDown,
  FaHistory,
  FaClock,
  FaUsers,
  FaUtensils,
  FaHourglassHalf,
  FaCheckCircle,
  FaStar,
  FaTrophy,
  FaEye,
  FaTruck,
} from "react-icons/fa";

export default function Dashboard() {
  const [selectedDate, setSelectedDate] = useState(
    new Date().toISOString().split("T")[0]
  );
  const [stats, setStats] = useState({
    todayOrders: 0,
    todayRevenue: 0,
    allTimeOrders: 0,
    allTimeRevenue: 0,
    selectedDayOrders: 0,
    selectedDayRevenue: 0,
    totalCustomers: 0,
    menuItems: 0,
    pendingOrders: 0,
    deliveredOrders: 0,
    averageRating: 0,
    totalDeliveryStaffs: 0,
  });
  const [recentOrders, setRecentOrders] = useState([]);
  const [topSellingItems, setTopSellingItems] = useState([]);
  const [loading, setLoading] = useState(true);
const loadDashboardData = React.useCallback(() => {
  setLoading(true);

  // Mock data - replace with actual API calls
  setTimeout(() => {
    const today = new Date().toISOString().split("T")[0];
    const isToday = selectedDate === today;

    setStats({
      todayOrders: 43,
      todayRevenue: 12850,
      allTimeOrders: 2847,
      allTimeRevenue: 485600,
      selectedDayOrders: isToday ? 43 : Math.floor(Math.random() * 60) + 10,
      selectedDayRevenue: isToday
        ? 12850
        : Math.floor(Math.random() * 20000) + 5000,
      totalCustomers: 856,
      menuItems: 124,
      pendingOrders: 12,
      deliveredOrders: 2835,
      averageRating: 4.6,
      totalDeliveryStaffs: 18,
    });

    // Mock recent orders
    setRecentOrders([
      {
        id: "ORD001",
        customerName: "Vikram Malhotra",
        items: 3,
        total: 850,
        status: "preparing",
        time: "10 mins ago",
      },
      {
        id: "ORD002",
        customerName: "Anjali Gupta",
        items: 2,
        total: 650,
        status: "delivered",
        time: "25 mins ago",
      },
      {
        id: "ORD003",
        customerName: "Rohan Das",
        items: 5,
        total: 1200,
        status: "delivery",
        time: "32 mins ago",
      },
      {
        id: "ORD004",
        customerName: "Meera Iyer",
        items: 1,
        total: 320,
        status: "pending",
        time: "45 mins ago",
      },
      {
        id: "ORD005",
        customerName: "Arjun Singh",
        items: 4,
        total: 980,
        status: "delivered",
        time: "1 hour ago",
      },
    ]);

    // Mock top selling items
    setTopSellingItems([
      { name: "Butter Chicken", orders: 45, revenue: 11250 },
      { name: "Pizza Margherita", orders: 38, revenue: 9500 },
      { name: "Chicken Biryani", orders: 32, revenue: 8000 },
      { name: "Paneer Tikka", orders: 28, revenue: 5600 },
      { name: "Masala Dosa", orders: 25, revenue: 3750 },
    ]);

    setLoading(false);
  }, 500);
}, [selectedDate]);

useEffect(() => {
  loadDashboardData();
}, [loadDashboardData]);

const formatCurrency = (amount) => {
  return new Intl.NumberFormat("en-IN", {
    style: "currency",
    currency: "INR",
    maximumFractionDigits: 0,
  }).format(amount);
};

const formatDate = (dateString) => {
  const date = new Date(dateString);
  const today = new Date();
  const yesterday = new Date(today);
  yesterday.setDate(yesterday.getDate() - 1);

  if (dateString === today.toISOString().split("T")[0]) {
    return "Today";
  } else if (dateString === yesterday.toISOString().split("T")[0]) {
    return "Yesterday";
  } else {
    return date.toLocaleDateString("en-IN", {
      weekday: "short",
      day: "numeric",
      month: "short",
      year: "numeric",
    });
  }
};

const getOrderStatusColor = (status) => {
  switch (status) {
    case "pending":
      return "bg-gray-100 text-gray-800";
    case "preparing":
      return "bg-yellow-100 text-yellow-800";
    case "delivery":
      return "bg-blue-100 text-blue-800";
    case "delivered":
      return "bg-green-100 text-green-800";
    default:
      return "bg-gray-100 text-gray-800";
  }
};

const StatCard = ({ title, value, icon: IconComponent, color, subtext }) => (
  <Card className="p-6 hover:shadow-md transition-all duration-200 border border-gray-100">
    <div className="flex items-start justify-between">
      <div className="flex-1">
        <div className="flex items-center gap-3 mb-3">
          <div className={`p-2.5 rounded-lg ${color} flex-shrink-0`}>
            {IconComponent && (
              <IconComponent className="text-lg text-white" />
            )}
          </div>
          <p className="text-sm font-medium text-gray-600 leading-tight">
            {title}
          </p>
        </div>
        <div className="space-y-1">
          <p className="text-2xl font-bold text-gray-900 leading-none">
            {value}
          </p>
          {subtext && (
            <p className="text-xs text-gray-500 leading-relaxed">{subtext}</p>
          )}
        </div>
      </div>
    </div>
  </Card>
);

return (
  <div className="space-y-6">
    {/* Header */}
    <div>
      <h1 className="text-2xl sm:text-3xl font-bold text-gray-900">
        Dashboard
      </h1>
      <p className="text-sm sm:text-base text-gray-600 mt-1">
        Monitor your restaurant's orders and revenue
      </p>
    </div>

    {/* Date Selector */}
    <Card className="p-4 sm:p-6">
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
        <div className="flex items-center gap-3">
          <FaCalendarAlt className="text-red-600 text-xl" />
          <div>
            <h3 className="text-base sm:text-lg font-semibold text-gray-900">
              Select Date
            </h3>
            <p className="text-xs sm:text-sm text-gray-600">
              View orders and revenue for any specific day
            </p>
          </div>
        </div>
        <div className="flex flex-col sm:flex-row sm:items-center gap-2 sm:gap-3">
          <label
            htmlFor="date-picker"
            className="text-sm font-medium text-gray-700"
          >
            Choose Date:
          </label>
          <input
            id="date-picker"
            type="date"
            value={selectedDate}
            onChange={(e) => setSelectedDate(e.target.value)}
            max={new Date().toISOString().split("T")[0]}
            className="px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-red-500
          />
          <span className="text-sm font-medium text-red-600">
            {formatDate(selectedDate)}
          </span>
        </div>
      </div>
    </Card>

    {/* Today's Stats */}
    <div>
      <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center gap-2">
        <FaClock className="text-red-600" />
        Today's Performance
      </h2>
      <div className="grid grid-cols-2 gap-3 md:gap-6">
        <StatCard
          title="Today's Orders"
          value={loading ? "..." : stats.todayOrders}
          icon={FaShoppingBag}
          color="bg-blue-500"
          subtext={`Orders received today (${new Date().toLocaleDateString(
            "en-IN"
          )})`}
        />
        <StatCard
```

```
          title="Today's Revenue"
          value={loading ? "..." : formatCurrency(stats.todayRevenue)}
          icon={FaRupeeSign}
          color="bg-green-500"
          subtext="Total earnings today"
        />
      </div>
    </div>

    {/* Selected Day Stats */}
    {selectedDate !== new Date().toISOString().split("T")[0] && (
      <div>
        <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center gap-2
          <FaCalendarAlt className="text-red-600" />
          {formatDate(selectedDate)} Performance
        </h2>
        <div className="grid grid-cols-2 gap-3 md:gap-6">
          <StatCard
            title={`${formatDate(selectedDate)} Orders`}
            value={loading ? "..." : stats.selectedDayOrders}
            icon={FaShoppingBag}
            color="bg-purple-500"
            subtext={`Orders on ${new Date(selectedDate).toLocaleDateString(
              "en-IN"
            )}`}
          />
          <StatCard
            title={`${formatDate(selectedDate)} Revenue`}
            value={loading ? "..." : formatCurrency(stats.selectedDayRevenue)}
            icon={FaRupeeSign}
            color="bg-orange-500"
            subtext="Total earnings for selected day"
          />
        </div>
      </div>
    )}

    {/* All Time Stats */}
    <div>
      <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center gap-2">
        <FaHistory className="text-red-600" />
        All Time Performance
      </h2>
      <div className="grid grid-cols-2 gap-3 md:gap-6">
        <StatCard
          title="Total Orders"
          value={
            loading ? "..." : stats.allTimeOrders.toLocaleString("en-IN")
          }
          icon={FaChartLine}
          color="bg-red-600"
          subtext="All orders since restaurant started"
        />
        <StatCard
          title="Total Revenue"
          value={loading ? "..." : formatCurrency(stats.allTimeRevenue)}
          icon={FaRupeeSign}
          color="bg-emerald-600"
          subtext="Total earnings since inception"
        />
      </div>
    </div>

    {/* Additional Business Metrics */}
    <div>
      <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center gap-2">
        <FaChartLine className="text-red-600" />
        Business Metrics
      </h2>
      <div className="grid grid-cols-2 md:grid-cols-3 gap-3 md:gap-6">
        <StatCard
          title="Total Customers"
          value={
            loading ? "..." : stats.totalCustomers.toLocaleString("en-IN")
          }
          icon={FaUsers}
          color="bg-blue-600"
          subtext="Registered customers"
        />
        <StatCard
          title="Total Delivery Staffs"
          value={loading ? "..." : stats.totalDeliveryStaffs}
          icon={FaTruck}
          color="bg-indigo-600"
          subtext="Active delivery personnel"
        />
        <StatCard
          title="Menu Items"
          value={loading ? "..." : stats.menuItems}
          icon={FaUtensils}
          color="bg-orange-500"
          subtext="Available dishes"
        />
        <StatCard
          title="Pending Orders"
          value={loading ? "..." : stats.pendingOrders}
          icon={FaHourglassHalf}
          color="bg-yellow-500"
          subtext="Awaiting preparation"
        />
        <StatCard
          title="Delivered Orders"
          value={
            loading ? "..." : stats.deliveredOrders.toLocaleString("en-IN")
          }
          icon={FaCheckCircle}
          color="bg-green-600"
          subtext="Successfully completed"
        />
        <StatCard
          title="Average Rating"
          value={loading ? "..." : `${stats.averageRating}⭐`}
          icon={FaStar}
          color="bg-amber-500"
          subtext="Customer satisfaction"
          trend={5}
        />
      </div>
    </div>

    {/* Recent Orders and Top Selling Items */}
    <div className="grid grid-cols-1 xl:grid-cols-2 gap-4 sm:gap-6">
      {/* Recent Orders */}
      <Card className="p-4 sm:p-6">
        <div className="flex items-center justify-between mb-4">
          <h3 className="text-base sm:text-lg font-semibold text-gray-900 flex items-center gap-2"
            <FaShoppingBag className="text-red-600" />
            Recent Orders
          </h3>
          <Link
            to="/admin/orders"
            className="text-red-600 hover:text-red-700 text-xs sm:text-sm font-medium flex items-c
          >
            <FaEye />
            <span className="hidden sm:inline">View All</span>
          </Link>
        </div>
        <div className="space-y-3">
          {recentOrders.slice(0, 5).map((order) => (
            <div
              key={order.id}
              className="flex items-center justify-between p-3 bg-gray-50 rounded-lg"
            >
              <div className="flex-1 min-w-0">
                <p className="font-medium text-gray-900 text-sm sm:text-base truncate">
                  {order.customerName}
                </p>
                <p className="text-xs sm:text-sm text-gray-600">
                {order.items} items • ₹{order.total}
              </p>
            </div>
            <div className="text-right flex-shrink-0 ml-2">
              <span
                className={`px-2 py-1 rounded-full text-xs font-medium ${getOrderStatusColor(
                  order.status
                )}`}
              >
                {order.status}
              </span>
              <p className="text-xs text-gray-500 mt-1">{order.time}</p>
            </div>
          </div>
        ))}
      </div>
    </Card>

    {/* Top Selling Items */}
    <Card className="p-4 sm:p-6">
      <div className="flex items-center justify-between mb-4">
        <h3 className="text-base sm:text-lg font-semibold text-gray-900 flex items-center gap-2"
          <FaTrophy className="text-red-600" />
          Top Selling Items
        </h3>
        <Link
          to="/admin/menu"
          className="text-red-600 hover:text-red-700 text-xs sm:text-sm font-medium flex items-c
        >
          <FaEye />
          <span className="hidden sm:inline">View Menu</span>
        </Link>
      </div>
      <div className="space-y-3">
        {topSellingItems.map((item, index) => (
          <div
            key={item.name}
            className="flex items-center justify-between p-3 bg-gray-50 rounded-lg"
          >
            <div className="flex items-center gap-3 flex-1 min-w-0">
              <div className="w-7 h-7 sm:w-8 sm:h-8 bg-red-100 rounded-full flex items-center ju
                <span className="text-xs sm:text-sm font-bold text-red-600">
                  #{index + 1}
                </span>
              </div>
              <div className="min-w-0 flex-1">
                <p className="font-medium text-gray-900 text-sm sm:text-base">
                  <span className="truncate">{item.name}</span>
                </p>
                <p className="text-xs sm:text-sm text-gray-600">
                  {item.orders} orders
                </p>
              </div>
            </div>
            <div className="text-right flex-shrink-0 ml-2">
              <p className="font-medium text-gray-900 text-sm sm:text-base">
                ₹{item.revenue.toLocaleString("en-IN")}
              </p>
              <p className="text-xs text-gray-500">Revenue</p>
            </div>
          </div>
        ))}
      </div>
    </Card>
  </div>
</div>
);
}
```

10. src/pages/customer/MenuPage.jsx

```
 import React, { useState, useEffect, useMemo } from "react";
import { useSearchParams } from "react-router-dom";
import { Container, Grid } from "../../components/shared/Layout.jsx";
import { mockAPI } from "../../services/mockApi.js";
import SearchBar from "../../components/customer/SearchBar.jsx";
import FilterPanel from "../../components/customer/FilterPanel.jsx";
import SortOptions from "../../components/customer/SortOptions.jsx";
import FoodCard from "../../components/customer/FoodCard.jsx";
import { InlineLoader } from "../../components/shared/LoadingSpinner.jsx";

const CATEGORIES = ["All", "Indian", "Chinese", "South", "Tandoor"];

export default function MenuPage() {
  const [searchParams] = useSearchParams();
  const [menuItems, setMenuItems] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Search and filter state
  const [searchQuery, setSearchQuery] = useState("");
  const [filters, setFilters] = useState({});
  const [sortBy, setSortBy] = useState("relevance");
  const [activeCategory, setActiveCategory] = useState("");

  useEffect(() => {
    loadMenuItems();
  }, []);

  useEffect(() => {
    // Set category from URL parameter
    const categoryParam = searchParams.get("category");
    const searchParam = searchParams.get("search");

    if (categoryParam && CATEGORIES.includes(categoryParam)) {
      setActiveCategory(categoryParam);
    }

    if (searchParam) {
      setSearchQuery(searchParam);
    }
  }, [searchParams]);

  const loadMenuItems = async () => {
    try {
      setLoading(true);
      const response = await mockAPI.getMenuItems();
      if (response.success) {
        setMenuItems(response.data);
      } else {
        setError("Failed to load menu items");
      }
    } catch (err) {
      setError("Something went wrong while loading menu items");
      console.error("Error loading menu items:", err?.message || String(err));
    } finally {
      setLoading(false);
    }
  };

  // Generate search suggestions
  const searchSuggestions = useMemo(() => {
    const suggestions = new Set();

    menuItems.forEach((item) => {
      // Add item names
      suggestions.add(item.name);
      // Add categories
      suggestions.add(item.category);
      // Add tags
      if (item.tags) {
        item.tags.forEach((tag) => suggestions.add(tag));
      }
    });
    return Array.from(suggestions);
  }, [menuItems]);

  // Filter and sort menu items
  const filteredAndSortedItems = useMemo(() => {
    let filtered = [...menuItems];

    // Apply category filter from tabs
    if (activeCategory && activeCategory !== "All") {
      filtered = filtered.filter((item) => item.category === activeCategory);
    }

    // Apply search query
    if (searchQuery.trim()) {
      const query = searchQuery.toLowerCase();
      filtered = filtered.filter(
        (item) =>
          item.name.toLowerCase().includes(query) ||
          item.description.toLowerCase().includes(query) ||
          item.category.toLowerCase().includes(query) ||
          (item.tags &&
            item.tags.some((tag) => tag.toLowerCase().includes(query)))
      );
    }

    // Apply filters
    if (filters.category) {
      filtered = filtered.filter((item) => item.category === filters.category);
    }

    if (filters.type) {
      filtered = filtered.filter((item) => item.type === filters.type);
    }

    if (filters.minPrice !== undefined && filters.maxPrice !== undefined) {
      filtered = filtered.filter(
        (item) =>
          item.price >= filters.minPrice && item.price <= filters.maxPrice
      );
    }

    // Apply sorting
    switch (sortBy) {
      case "rating":
        filtered.sort((a, b) => b.rating - a.rating);
        break;
      case "price-low":
        filtered.sort((a, b) => a.price - b.price);
        break;
      case "price-high":
        filtered.sort((a, b) => b.price - a.price);
        break;
      case "name":
        filtered.sort((a, b) => a.name.localeCompare(b.name));
        break;
      case "delivery-time":
        filtered.sort((a, b) => a.preparationTime - b.preparationTime);
        break;
      default:
        // Relevance - keep original order or sort by rating
        filtered.sort((a, b) => b.rating - a.rating);
    }

    return filtered;
  }, [menuItems, searchQuery, filters, sortBy, activeCategory]);

  const handleSearch = (query) => {
    setSearchQuery(query);
  };

  const handleFiltersChange = (newFilters) => {
    setFilters(newFilters);
  };
  const handleClearFilters = () => {
    setFilters({});
  };

  const handleSortChange = (newSortBy) => {
    setSortBy(newSortBy);
  };

  const handleCategoryChange = (category) => {
    setActiveCategory(category);
    // Clear category filter when using tabs
    if (filters.category) {
      const newFilters = { ...filters };
      delete newFilters.category;
      setFilters(newFilters);
    }
  };

  if (loading) {
    return (
      <Container className="py-8">
        <InlineLoader text="Loading menu..." />
      </Container>
    );
  }

  if (error) {
    return (
      <Container className="py-8">
        <div className="text-center">
          <div className="bg-error-red text-white p-4 rounded-lg max-w-md mx-auto">
            <h3 className="font-semibold mb-2">Oops! Something went wrong</h3>
            <p className="text-sm">{error}</p>
            <button
              onClick={loadMenuItems}
              className="mt-3 bg-white text-error-red px-4 py-2 rounded font-medium hover:bg-gray-10
            >
              Try Again
            </button>
          </div>
        </div>
      </Container>
    );
  }

  return (
    <Container className="py-4 md:py-8">
      {/* Header */}
      <div className="mb-6">
        <h1 className="text-3xl font-bold text-dark-red mb-2">Our Menu</h1>
        <p className="text-gray-600">
          Discover delicious dishes from our kitchen
        </p>
      </div>

      {/* Search Bar */}
      <div className="mb-6 flex justify-center">
        <SearchBar
          onSearch={handleSearch}
          suggestions={searchSuggestions}
          placeholder="Search for dishes, cuisines, or ingredients..."
        />
      </div>

      {/* Category Tabs */}
      <div className="mb-6">
        <div className="flex space-x-2 overflow-x-auto scrollbar-hide pb-2">
          {CATEGORIES.map((category) => (
            <button
              key={category}
              onClick={() =>
                handleCategoryChange(category === "All" ? "" : category)
              }
          className={`
            flex-shrink-0 px-4 py-2 rounded-full font-medium transition-colors
            ${
              activeCategory === category ||
              (category === "All" && !activeCategory)
                ? "bg-dark-red text-white"
                : "bg-gray-100 text-gray-700 hover:bg-gray-200"
            }
          `}
        >
          {category}
        </button>
      ))}
    </div>
  </div>

  {/* Controls */}
  <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4 mb-6">
    <div className="flex items-center space-x-4">
      <FilterPanel
        filters={filters}
        onFiltersChange={handleFiltersChange}
        onClear={handleClearFilters}
      />
      <SortOptions sortBy={sortBy} onSortChange={handleSortChange} />
    </div>

    <div className="text-sm text-gray-600">
      {filteredAndSortedItems.length} item
      {filteredAndSortedItems.length !== 1 ? "s" : ""} found
    </div>
  </div>

  {/* Results */}
  {filteredAndSortedItems.length === 0 ? (
    <div className="text-center py-12">
      <div className="w-24 h-24 bg-gray-100 rounded-full flex items-center justify-center mx-aut
        <svg
          className="w-12 h-12 text-gray-400"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z"
          />
        </svg>
      </div>
      <h3 className="text-xl font-semibold text-gray-800 mb-2">
        No items found
      </h3>
      <p className="text-gray-600 mb-4">
        Try adjusting your search or filters to find what you're looking
        for.
      </p>
      <button
        onClick={() => {
          setSearchQuery("");
          setFilters({});
          setActiveCategory("");
        }}
        className="bg-dark-red text-white px-6 py-2 rounded-lg hover:bg-hover-red transition-col
      >
        Clear All Filters
      </button>
    </div>
  ) : (
    <Grid cols={4} gap={6}>
      {filteredAndSortedItems.map((item) => (
        <FoodCard key={item.id} item={item} />
      ))}
```

```
        </Grid>
      )}
    </Container>
  );
}
```

11. src/pages/admin/OrderManagement.jsx

```
 import React, { useState, useEffect } from "react";
import { Container, Card } from "../../components/shared/Layout.jsx";
import ConfirmDialog from "../../components/shared/ConfirmDialog.jsx";
import { useToast } from "../../components/shared/Toast.jsx";
import {
  FaSearch,
  FaFilter,
  FaClock,
  FaCheck,
  FaTruck,
  FaEye,
  FaPhoneAlt,
  FaMapMarkerAlt,
} from "react-icons/fa";

export default function OrderManagement() {
  const [orders, setOrders] = useState([]);
  const [filteredOrders, setFilteredOrders] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [statusFilter, setStatusFilter] = useState("all");
  const [loading, setLoading] = useState(true);

  // Professional confirmation dialog state
  const [confirmDialog, setConfirmDialog] = useState({
    isOpen: false,
    title: "",
    message: "",
    onConfirm: null,
    type: "warning",
  });

  // Toast notifications
  const { addToast } = useToast();

  // Mock orders data
  useEffect(() => {
    const mockOrders = [
      {
        id: "ORD001",
        customerName: "Rahul Sharma",
        customerPhone: "+91-9876543210",
        items: [
          { name: "Butter Chicken", quantity: 2, price: 250 },
          { name: "Naan", quantity: 3, price: 60 },
        ],
        total: 680,
        status: "pending",
        orderTime: "2025-01-22T14:30:00",
        address: "123 MG Road, Bangalore",
        paymentMethod: "Online",
      },
      {
        id: "ORD002",
        customerName: "Priya Patel",
        customerPhone: "+91-9876543211",
        items: [
          { name: "Pizza Margherita", quantity: 1, price: 299 },
          { name: "Garlic Bread", quantity: 2, price: 120 },
        ],
        total: 539,
        status: "preparing",
        orderTime: "2025-01-22T14:15:00",
        address: "456 Brigade Road, Bangalore",
        paymentMethod: "Cash on Delivery",
      },
      {
        id: "ORD003",
        customerName: "Amit Kumar",
        customerPhone: "+91-9876543212",
        items: [
          { name: "Chicken Biryani", quantity: 1, price: 320 },
          { name: "Raita", quantity: 1, price: 80 },
        ],
        total: 400,
        status: "delivery",
        orderTime: "2025-01-22T13:45:00",
        address: "789 Koramangala, Bangalore",
        paymentMethod: "UPI",
      },
      {
        id: "ORD004",
        customerName: "Sneha Reddy",
        customerPhone: "+91-9876543213",
        items: [
          { name: "Paneer Tikka", quantity: 1, price: 220 },
          { name: "Dal Tadka", quantity: 1, price: 180 },
          { name: "Rice", quantity: 2, price: 120 },
        ],
        total: 520,
        status: "delivered",
        orderTime: "2025-01-22T12:30:00",
        address: "321 Whitefield, Bangalore",
        paymentMethod: "Credit Card",
      },
    ];

    setTimeout(() => {
      setOrders(mockOrders);
      setFilteredOrders(mockOrders);
      setLoading(false);
    }, 500);
  }, []);

  // Filter orders based on search and status
  useEffect(() => {
    let filtered = orders;

    if (searchTerm) {
      filtered = filtered.filter(
        (order) =>
          order.id.toLowerCase().includes(searchTerm.toLowerCase()) ||
          order.customerName.toLowerCase().includes(searchTerm.toLowerCase()) ||
          order.customerPhone.includes(searchTerm)
      );
    }

    if (statusFilter !== "all") {
      filtered = filtered.filter((order) => order.status === statusFilter);
    }

    setFilteredOrders(filtered);
  }, [searchTerm, statusFilter, orders]);

  const getStatusColor = (status) => {
    switch (status) {
      case "pending":
        return "bg-yellow-100 text-yellow-800";
      case "preparing":
        return "bg-orange-100 text-orange-800";
      case "delivery":
        return "bg-red-100 text-red-800";
      case "delivered":
        return "bg-green-100 text-green-800";
      default:
        return "bg-gray-100 text-gray-800";
    }
  };

  const getStatusIcon = (status) => {
    switch (status) {
      case "pending":
        return <FaClock className="mr-1" />;
      case "preparing":
        return <FaClock className="mr-1" />;
      case "delivery":
        return <FaTruck className="mr-1" />;
      case "delivered":
        return <FaCheck className="mr-1" />;
      default:
        return <FaClock className="mr-1" />;
    }
  };

  const updateOrderStatus = (orderId, newStatus) => {
    setOrders((prevOrders) =>
      prevOrders.map((order) =>
        order.id === orderId ? { ...order, status: newStatus } : order
      )
    );
  };

  const handleStatusUpdate = (orderId, newStatus, customerName) => {
    let dialogConfig = {};

    switch (newStatus) {
      case "preparing":
        dialogConfig = {
          title: "Mark Order as Preparing",
          message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Preparing"?
          type: "info",
        };
        break;
      case "delivery":
        dialogConfig = {
          title: "Send Order for Delivery",
          message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Out for Del
          type: "warning",
        };
        break;
      case "delivered":
        dialogConfig = {
          title: "Mark Order as Delivered",
          message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Delivered"?
          type: "danger",
        };
        break;
      default:
        dialogConfig = {
          title: "Update Order Status",
          message: `Are you sure you want to update the status of order ${orderId}?`,
          type: "warning",
        };
    }

    setConfirmDialog({
      isOpen: true,
      ...dialogConfig,
      onConfirm: () => {
        updateOrderStatus(orderId, newStatus);

        // Show success toast
        const successMessages = {
          preparing: "Order marked as preparing successfully!",
          delivery: "Order sent for delivery successfully!",
          delivered: "Order marked as delivered successfully!",
        };

        addToast(
          successMessages[newStatus] || "Order status updated successfully!",
          "success",
          4000
        );
      },
    });
  };

  const formatTime = (timeString) => {
    return new Date(timeString).toLocaleString("en-IN", {
      hour: "2-digit",
      minute: "2-digit",
      day: "2-digit",
      month: "short",
    });
  };

  return (
    <Container className="py-6">
      <div className="space-y-6">
        {/* Header */}
        <div>
          <h1 className="text-2xl sm:text-3xl font-bold text-gray-900 mb-2">
            Order Management
          </h1>
          <p className="text-gray-600">Manage and track all customer orders</p>
        </div>

        {/* Search and Filters */}
        <Card className="p-4 sm:p-6">
          <div className="flex flex-col sm:flex-row gap-4">
            {/* Search */}
            <div className="flex-1 relative">
              <FaSearch className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400"
              <input
                type="text"
                placeholder="Search by Order ID, Customer Name, or Phone..."
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded-lg focus:outline-no
              />
            </div>

            {/* Status Filter */}
            <div className="relative">
              <FaFilter className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400"
              <select
                value={statusFilter}
                onChange={(e) => setStatusFilter(e.target.value)}
                className="pl-10 pr-8 py-2 border border-gray-300 rounded-lg focus:outline-none focu
              >
                <option value="all">All Orders</option>
                <option value="pending">Pending</option>
                <option value="preparing">Preparing</option>
                <option value="delivery">Out for Delivery</option>
                <option value="delivered">Delivered</option>
              </select>
            </div>
          </div>
        </Card>

        {/* Orders List */}
        <div className="space-y-4">
          {loading ? (
            <Card className="p-6 text-center">
              <p className="text-gray-600">Loading orders...</p>
            </Card>
          ) : filteredOrders.length === 0 ? (
            <Card className="p-6 text-center">
              <p className="text-gray-600">No orders found</p>
            </Card>
          ) : (
            filteredOrders.map((order) => (
              <Card key={order.id} className="p-4 sm:p-6">
                <div className="space-y-4">
                  {/* Order Header */}
          <div className="flex flex-col sm:flex-row sm:items-center justify-between gap-3">
            <div className="flex items-center gap-3">
              <h3 className="text-lg font-semibold text-gray-900">
                {order.id}
              </h3>
              <span
                className={`px-3 py-1 rounded-full text-xs font-medium flex items-center ${g
                  order.status
                )}`}
              >
                {getStatusIcon(order.status)}
                {order.status.charAt(0).toUpperCase() +
                  order.status.slice(1)}
              </span>
            </div>
            <div className="text-sm text-gray-500">
              {formatTime(order.orderTime)}
            </div>
          </div>

          {/* Customer Info */}
          <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
            <div>
              <h4 className="font-medium text-gray-900 mb-2">
                Customer Details
              </h4>
              <div className="space-y-1 text-sm">
                <p className="flex items-center gap-2">
                  <FaEye className="text-gray-400" />
                  {order.customerName}
                </p>
                <p className="flex items-center gap-2">
                  <FaPhoneAlt className="text-gray-400" />
                  {order.customerPhone}
                </p>
                <p className="flex items-start gap-2">
                  <FaMapMarkerAlt className="text-gray-400 mt-0.5" />
                  <span>{order.address}</span>
                </p>
              </div>
            </div>

            <div>
              <h4 className="font-medium text-gray-900 mb-2">
                Order Items
              </h4>
              <div className="space-y-1 text-sm">
                {order.items.map((item, index) => (
                  <div
                    key={index}
                    className="flex justify-between items-center"
                  >
                    <span>
                      {item.name} x{item.quantity}
                    </span>
                    <span>₹{item.price * item.quantity}</span>
                  </div>
                ))}
                <div className="border-t pt-1 mt-2 font-medium">
                  <div className="flex justify-between">
                    <span>Total Amount</span>
                    <span>₹{order.total}</span>
                  </div>
                  <div className="flex justify-between text-xs text-gray-600">
                    <span>Payment</span>
                    <span>{order.paymentMethod}</span>
                  </div>
                </div>
              </div>
            </div>
          </div>

          {/* Action Buttons */}
          {order.status !== "delivered" && (
                    <div className="flex flex-wrap gap-2 pt-3 border-t">
                      {order.status !== "preparing" && (
                        <button
                          onClick={() =>
                            handleStatusUpdate(
                              order.id,
                              "preparing",
                              order.customerName
                            )
                          }
                          className="px-4 py-2 bg-orange-600 text-white rounded-lg hover:bg-orange-7
                        >
                          <FaClock />
                          Mark as Preparing
                        </button>
                      )}
                      {order.status !== "delivery" && (
                        <button
                          onClick={() =>
                            handleStatusUpdate(
                              order.id,
                              "delivery",
                              order.customerName
                            )
                          }
                          className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 tex
                        >
                          <FaTruck />
                          Out for Delivery
                        </button>
                      )}
                      <button
                        onClick={() =>
                          handleStatusUpdate(
                            order.id,
                            "delivered",
                            order.customerName
                          )
                        }
                        className="px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 t
                      >
                        <FaCheck />
                        Mark as Delivered
                      </button>
                    </div>
                  )}
                </div>
              </Card>
            ))
          )}
        </div>
      </div>

      {/* Professional Confirmation Dialog */}
      <ConfirmDialog
        isOpen={confirmDialog.isOpen}
        onClose={() => setConfirmDialog((prev) => ({ ...prev, isOpen: false }))}
        onConfirm={confirmDialog.onConfirm}
        title={confirmDialog.title}
        message={confirmDialog.message}
        type={confirmDialog.type}
        confirmText="Yes, Update Status"
        cancelText="Cancel"
      />
    </Container>
  );
}
```

12. src/pages/admin/MenuManagement.jsx

```
 import React, { useState, useEffect } from "react";
import { Container, Card } from "../../components/shared/Layout.jsx";
import ConfirmDialog from "../../components/shared/ConfirmDialog.jsx";
import { useToast } from "../../components/shared/Toast.jsx";
import {
  FaPlus,
  FaEdit,
  FaTrash,
  FaSearch,
  FaFilter,
  FaToggleOn,
  FaToggleOff,
  FaRupeeSign,
  FaClock,
  FaUtensils,
} from "react-icons/fa";

export default function MenuManagement() {
  const [menuItems, setMenuItems] = useState([]);
  const [categories, setCategories] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [selectedCategory, setSelectedCategory] = useState("all");
  const [loading, setLoading] = useState(true);
  const [showAddForm, setShowAddForm] = useState(false);
  const [editingItem, setEditingItem] = useState(null);
  const [confirmDialog, setConfirmDialog] = useState({
    isOpen: false,
    title: "",
    message: "",
    onConfirm: null,
    type: "warning",
  });

  const { addToast } = useToast();

  // Mock data for menu items
  useEffect(() => {
    const mockCategories = [
      { id: 1, name: "Trending Now", items: 8 },
      { id: 2, name: "Indian Cuisine", items: 15 },
      { id: 3, name: "Chinese Cuisine", items: 10 },
      { id: 4, name: "South Cuisine", items: 6 },
      { id: 5, name: "Tandoor Cuisine", items: 4 },
    ];

    const mockMenuItems = [
      {
        id: 1,
        name: "Butter Chicken",
        category: "Indian Cuisine",
        price: 250,
        description: "Creamy tomato-based curry with tender chicken pieces",
        prepTime: 25,
        isAvailable: true,
        isVeg: false,
        tags: ["comfort food", "creamy", "popular"],
        image:
          "https://images.unsplash.com/photo-1588166524941-3bf61a9c41db?w=150&h=150&fit=crop",
      },
      {
        id: 2,
        name: "Paneer Tikka",
        category: "Tandoor Cuisine",
        price: 180,
        description: "Grilled cottage cheese cubes with aromatic spices",
        prepTime: 15,
        isAvailable: true,
        isVeg: true,
        tags: ["vegetarian", "healthy", "grilled"],
        image:
          "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=150&h=150&fit=crop",
      },
      {
        id: 3,
        name: "Chicken Manchurian",
        category: "Chinese Cuisine",
        price: 220,
        description: "Spicy chicken balls in tangy sauce",
        prepTime: 20,
        isAvailable: true,
        isVeg: false,
        tags: ["spicy", "tangy", "chinese"],
        image:
          "https://images.unsplash.com/photo-1571934811356-5cc061b6821f?w=150&h=150&fit=crop",
      },
      {
        id: 4,
        name: "Masala Dosa",
        category: "South Cuisine",
        price: 120,
        description: "Crispy rice crepe with spiced potato filling",
        prepTime: 15,
        isAvailable: false,
        isVeg: true,
        tags: ["healthy", "vegetarian", "south indian"],
        image:
          "https://vismaifood.com/storage/app/uploads/public/45a/29b/a17/thumb__700_0_0_0_auto.jpg",
      },
      {
        id: 5,
        name: "Chicken Biryani",
        category: "Trending Now",
        price: 320,
        description: "Aromatic basmati rice with spiced chicken",
        prepTime: 35,
        isAvailable: true,
        isVeg: false,
        tags: ["trending", "aromatic", "biryani"],
        image:
          "https://www.licious.in/blog/wp-content/uploads/2022/06/chicken-hyderabadi-biryani-01.jpg"
      },
    ];

    setTimeout(() => {
      setCategories(mockCategories);
      setMenuItems(mockMenuItems);
      setLoading(false);
    }, 500);
  }, []);

  // Filter menu items
  const filteredItems = menuItems.filter((item) => {
    const matchesSearch =
      item.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
      item.description.toLowerCase().includes(searchTerm.toLowerCase());
    const matchesCategory =
      selectedCategory === "all" || item.category === selectedCategory;
    return matchesSearch && matchesCategory;
  });

  // Toggle item availability
  const toggleAvailability = (itemId) => {
    setMenuItems((prev) =>
      prev.map((item) =>
        item.id === itemId ? { ...item, isAvailable: !item.isAvailable } : item
      )
    );

    const item = menuItems.find((item) => item.id === itemId);
    const newStatus = !item.isAvailable;
    addToast(
      `${item.name} ${newStatus ? "enabled" : "disabled"} successfully!`,
      "success"
    );
  };
```

```
  // Delete item with confirmation
  const handleDeleteItem = (item) => {
    setConfirmDialog({
      isOpen: true,
      title: "Delete Menu Item",
      message: `Are you sure you want to delete "${item.name}"?\n\nThis action cannot be undone and
      type: "danger",
      onConfirm: () => {
        setMenuItems((prev) =>
          prev.filter((menuItem) => menuItem.id !== item.id)
        );
        addToast(`${item.name} deleted successfully!`, "success");
      },
    });
  };

  // Handle save item (add or edit)
  const handleSaveItem = (itemData) => {
    if (editingItem) {
      // Edit existing item
      setMenuItems((prev) =>
        prev.map((item) =>
          item.id === editingItem.id ? { ...item, ...itemData } : item
        )
      );
      addToast(`${itemData.name} updated successfully!`, "success");
    } else {
      // Add new item
      const newItem = {
        ...itemData,
        id: Math.max(...menuItems.map((item) => item.id), 0) + 1,
      };
      setMenuItems((prev) => [...prev, newItem]);
      addToast(`${itemData.name} added successfully!`, "success");
    }

    // Close form
    setShowAddForm(false);
    setEditingItem(null);
  };

  return (
    <Container className="py-6">
      <div className="space-y-6">
        {/* Header */}
        <div className="flex flex-col sm:flex-row justify-between items-start sm:items-center gap-4"
          <div>
            <h1 className="text-2xl sm:text-3xl font-bold text-gray-900">
              Menu Management
            </h1>
            <p className="text-gray-600 mt-1">
              Manage your restaurant menu items and categories
            </p>
          </div>
          <button
            onClick={() => setShowAddForm(true)}
            className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 flex items-center
          >
            <FaPlus />
            Add New Item
          </button>
        </div>

        {/* Statistics Cards */}
        <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
          <Card className="p-4">
            <div className="flex items-center gap-3">
              <div className="p-2 bg-blue-100 rounded-lg">
                <FaUtensils className="text-blue-600" />
              </div>
              <div>
                <p className="text-sm text-gray-600">Total Items</p>
                <p className="text-xl font-bold text-gray-900">
                  {menuItems.length}
                </p>
              </div>
            </div>
          </Card>

          <Card className="p-4">
            <div className="flex items-center gap-3">
              <div className="p-2 bg-green-100 rounded-lg">
                <FaToggleOn className="text-green-600" />
              </div>
              <div>
                <p className="text-sm text-gray-600">Available</p>
                <p className="text-xl font-bold text-gray-900">
                  {menuItems.filter((item) => item.isAvailable).length}
                </p>
              </div>
            </div>
          </Card>

          <Card className="p-4">
            <div className="flex items-center gap-3">
              <div className="p-2 bg-red-100 rounded-lg">
                <FaToggleOff className="text-red-600" />
              </div>
              <div>
                <p className="text-sm text-gray-600">Unavailable</p>
                <p className="text-xl font-bold text-gray-900">
                  {menuItems.filter((item) => !item.isAvailable).length}
                </p>
              </div>
            </div>
          </Card>

          <Card className="p-4">
            <div className="flex items-center gap-3">
              <div className="p-2 bg-orange-100 rounded-lg">
                <FaFilter className="text-orange-600" />
              </div>
              <div>
                <p className="text-sm text-gray-600">Categories</p>
                <p className="text-xl font-bold text-gray-900">
                  {categories.length}
                </p>
              </div>
            </div>
          </Card>
        </div>

        {/* Search and Filters */}
        <Card className="p-4 sm:p-6">
          <div className="flex flex-col sm:flex-row gap-4">
            {/* Search */}
            <div className="flex-1 relative">
              <FaSearch className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400"
              <input
                type="text"
                placeholder="Search menu items..."
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded-lg focus:outline-no
              />
            </div>

            {/* Category Filter */}
            <div className="relative">
              <FaFilter className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400"
              <select
                value={selectedCategory}
                onChange={(e) => setSelectedCategory(e.target.value)}
                className="pl-10 pr-8 py-2 border border-gray-300 rounded-lg focus:outline-none focu
              >
                <option value="all">All Categories</option>
                {categories.map((category) => (
                  <option key={category.id} value={category.name}>
                    {category.name}
              </option>
            ))}
          </select>
        </div>
      </div>
</Card>

{/* Menu Items List */}
<div className="space-y-4">
  {loading ? (
    <Card className="p-6 text-center">
      <p className="text-gray-600">Loading menu items...</p>
    </Card>
  ) : filteredItems.length === 0 ? (
    <Card className="p-6 text-center">
      <p className="text-gray-600">No menu items found</p>
    </Card>
  ) : (
    filteredItems.map((item) => (
      <Card key={item.id} className="p-4 sm:p-6">
        <div className="flex flex-col sm:flex-row gap-4">
          {/* Item Image */}
          <div className="w-full sm:w-24 h-48 sm:h-24 rounded-lg overflow-hidden bg-gray-200
            <img
              src={item.image}
              alt={item.name}
              className="w-full h-full object-cover"
              referrerPolicy="no-referrer"
              onError={(e) => {
                e.target.onerror = null;
                e.target.src = "https://images.unsplash.com/photo-1546833999-b9f581a1996d?w=
              }}
            />
          </div>

          {/* Item Details */}
          <div className="flex-1 space-y-2">
            <div className="flex flex-col sm:flex-row sm:items-start justify-between gap-2">
              <div>
                <div className="flex items-center gap-2">
                  <h3 className="text-lg font-semibold text-gray-900">
                    {item.name}
                  </h3>
                  <span
                    className={`px-2 py-1 rounded-full text-xs font-medium ${item.isVeg
                        ? "bg-green-100 text-green-800"
                        : "bg-red-100 text-red-800"
                      }`}
                  >
                    {item.isVeg ? "Veg" : "Non-Veg"}
                  </span>
                  <span
                    className={`px-2 py-1 rounded-full text-xs font-medium ${item.isAvailabl
                        ? "bg-green-100 text-green-800"
                        : "bg-red-100 text-red-800"
                      }`}
                  >
                    {item.isAvailable ? "Available" : "Unavailable"}
                  </span>
                </div>
                <p className="text-sm text-gray-600 mt-1">
                  {item.description}
                </p>
                <div className="flex items-center gap-4 mt-2 text-sm text-gray-500">
                  <span className="flex items-center gap-1">
                    <FaRupeeSign />₹{item.price}
                  </span>
                  <span className="flex items-center gap-1">
                    <FaClock />
                    {item.prepTime} mins
                  </span>
                  <span className="px-2 py-1 bg-gray-100 rounded">
                    {item.category}
                  </span>
```

```
                    {item.spiceLevel !== "None" && (
                      <span className="px-2 py-1 bg-orange-100 text-orange-800 rounded text-xs
                        {item.spiceLevel}
                      </span>
                    )}
                  </div>
                </div>

                {/* Action Buttons */}
                <div className="flex items-center gap-2">
                  <button
                    onClick={() => toggleAvailability(item.id)}
                    className={`p-2 rounded-lg ${item.isAvailable
                        ? "bg-green-100 text-green-600 hover:bg-green-200"
                        : "bg-red-100 text-red-600 hover:bg-red-200"
                      }`}
                    title={
                      item.isAvailable ? "Disable Item" : "Enable Item"
                    }
                  >
                    {item.isAvailable ? <FaToggleOn /> : <FaToggleOff />}
                  </button>
                  <button
                    onClick={() => setEditingItem(item)}
                    className="p-2 bg-blue-100 text-blue-600 rounded-lg hover:bg-blue-200"
                    title="Edit Item"
                  >
                    <FaEdit />
                  </button>
                  <button
                    onClick={() => handleDeleteItem(item)}
                    className="p-2 bg-red-100 text-red-600 rounded-lg hover:bg-red-200"
                    title="Delete Item"
                  >
                    <FaTrash />
                  </button>
                </div>
              </div>
            </div>
          </div>
        </Card>
      ))
    )}
  </div>
</div>

{/* Add/Edit Item Modal */}
{(showAddForm || editingItem) && (
  <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-
    <div className="bg-white rounded-lg max-w-2xl w-full max-h-[90vh] overflow-y-auto">
      <div className="p-6">
        <div className="flex justify-between items-center mb-6">
          <h2 className="text-xl font-bold text-gray-900">
            {editingItem ? "Edit Menu Item" : "Add New Menu Item"}
          </h2>
          <button
            onClick={() => {
              setShowAddForm(false);
              setEditingItem(null);
            }}
            className="text-gray-400 hover:text-gray-600"
          >
            <svg
              className="w-6 h-6"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M6 18L18 6M6 6l12 12
              />
```

```
                </svg>
              </button>
            </div>

            <ItemForm
              item={editingItem}
              categories={categories}
              onSave={handleSaveItem}
              onCancel={() => {
                setShowAddForm(false);
                setEditingItem(null);
              }}
            />
          </div>
        </div>
      </div>
    )}

    {/* Confirmation Dialog */}
    <ConfirmDialog
      isOpen={confirmDialog.isOpen}
      onClose={() => setConfirmDialog((prev) => ({ ...prev, isOpen: false }))}
      onConfirm={confirmDialog.onConfirm}
      title={confirmDialog.title}
      message={confirmDialog.message}
      type={confirmDialog.type}
      confirmText="Yes, Delete"
      cancelText="Cancel"
    />
  </Container>
  );
}

// ItemForm component for adding/editing menu items
function ItemForm({ item, categories, onSave, onCancel }) {
  const [formData, setFormData] = useState({
    name: item?.name || "",
    category: item?.category || "Trending Now",
    price: item?.price || "",
    description: item?.description || "",
    prepTime: item?.prepTime || "",
    isVeg: item?.isVeg || false,
    isAvailable: item?.isAvailable !== undefined ? item.isAvailable : true,
    image: item?.image || "",
    tags: item?.tags || [],
  });

  const [errors, setErrors] = useState({});
  const [imagePreview, setImagePreview] = useState(item?.image || "");
  const [uploadingImage, setUploadingImage] = useState(false);
  const [newTag, setNewTag] = useState("");

  // Handle form field changes
  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;
    let newValue = type === "checkbox" ? checked : value;

    // Limit description to 60 characters
    if (name === "description" && newValue.length > 60) {
      newValue = newValue.substring(0, 60);
    }

    setFormData((prev) => ({
      ...prev,
      [name]: newValue,
    }));

    // Clear error when user starts typing
    if (errors[name]) {
      setErrors((prev) => ({ ...prev, [name]: "" }));
    }
  };
```

```
  // Handle tag operations
  const addTag = () => {
    if (
      newTag.trim() &&
      formData.tags.length < 3 &&
      !formData.tags.includes(newTag.trim())
    ) {
      setFormData((prev) => ({
        ...prev,
        tags: [...prev.tags, newTag.trim()],
      }));
      setNewTag("");
    }
  };

  const removeTag = (tagToRemove) => {
    setFormData((prev) => ({
      ...prev,
      tags: prev.tags.filter((tag) => tag !== tagToRemove),
    }));
  };

  const handleTagInput = (e) => {
    const value = e.target.value;
    if (value.length <= 12) {
      setNewTag(value);
    }
  };

  // Handle image upload
  const handleImageUpload = (e) => {
    const file = e.target.files[0];
    if (!file) return;

    // Validate file type
    if (!file.type.startsWith("image/")) {
      setErrors((prev) => ({
        ...prev,
        image: "Please select a valid image file",
      }));
      return;
    }

    // Validate file size (max 2MB)
    if (file.size > 2 * 1024 * 1024) {
      setErrors((prev) => ({
        ...prev,
        image: "Image size must be less than 2MB",
      }));
      return;
    }

    setUploadingImage(true);

    // Create preview URL
    const reader = new FileReader();
    reader.onload = (e) => {
      const imageUrl = e.target.result;
      setImagePreview(imageUrl);
      setFormData((prev) => ({ ...prev, image: imageUrl }));
      setUploadingImage(false);

      // Clear any previous errors
      if (errors.image) {
        setErrors((prev) => ({ ...prev, image: "" }));
      }
    };
    reader.readAsDataURL(file);
  };

  const validateForm = () => {
    const newErrors = {};

    if (!formData.name.trim()) newErrors.name = "Name is required";
    if (!formData.category.trim()) newErrors.category = "Category is required";
    if (!formData.price || formData.price <= 0)
      newErrors.price = "Valid price is required";
    if (!formData.description.trim())
      newErrors.description = "Description is required";
    if (!formData.prepTime || formData.prepTime <= 0)
      newErrors.prepTime = "Valid prep time is required";

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (validateForm()) {
      onSave({
        ...formData,
        price: parseFloat(formData.price),
        prepTime: parseInt(formData.prepTime),
      });
    }
  };

  return (
    <form onSubmit={handleSubmit} className="space-y-6">
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        {/* Name */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Item Name *
          </label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleChange}
            className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 focus:bor
              }`}
            placeholder="Enter item name"
          />
          {errors.name && (
            <p className="text-red-500 text-sm mt-1">{errors.name}</p>
          )}
        </div>

        {/* Category */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Category *
          </label>
          <select
            name="category"
            value={formData.category}
            onChange={handleChange}
            className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 focus:bor
              }`}
          >
            {categories.map((cat) => (
              <option key={cat.id} value={cat.name}>
                {cat.name}
              </option>
            ))}
          </select>
          {errors.category && (
            <p className="text-red-500 text-sm mt-1">{errors.category}</p>
          )}
        </div>

        {/* Price */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Price (₹) *
          </label>
          <input
            type="number"
            name="price"
            value={formData.price}
            onChange={handleChange}
            min="0"
            step="0.01"
            className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 focus:bor
              }`}
            placeholder="Enter price"
          />
          {errors.price && (
            <p className="text-red-500 text-sm mt-1">{errors.price}</p>
          )}
        </div>

        {/* Prep Time */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Prep Time (minutes) *
          </label>
          <input
            type="number"
            name="prepTime"
            value={formData.prepTime}
            onChange={handleChange}
            min="1"
            className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 focus:bor
              }`}
            placeholder="Enter prep time"
          />
          {errors.prepTime && (
            <p className="text-red-500 text-sm mt-1">{errors.prepTime}</p>
          )}
        </div>

        {/* Tags Management */}
        <div className="md:col-span-2">
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Tags (Max 3 tags, 12 chars each)
          </label>
          <div className="space-y-3">
            {/* Add Tag Input */}
            <div className="flex gap-2">
              <input
                type="text"
                value={newTag}
                onChange={handleTagInput}
                onKeyPress={(e) =>
                  e.key === "Enter" && (e.preventDefault(), addTag())
                }
                maxLength={12}
                className="flex-1 px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:rin
                placeholder="Enter tag (max 12 chars)"
                disabled={formData.tags.length >= 3}
              />
              <button
                type="button"
                onClick={addTag}
                disabled={
                  !newTag.trim() ||
                  formData.tags.length >= 3 ||
                  formData.tags.includes(newTag.trim())
                }
                className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 disabled:bg-g
              >
                Add
              </button>
            </div>

            {/* Character Counter */}
            <div className="text-sm text-gray-400">
              {newTag.length}/12 characters • {formData.tags.length}/3 tags
            </div>

            {/* Display Tags */}
            {formData.tags.length > 0 && (
              <div className="flex flex-wrap gap-2">
                {formData.tags.map((tag, index) => (
                  <span
                    key={index}
                    className="inline-flex items-center gap-1 px-3 py-1 bg-blue-100 text-blue-800 ro
                  >
                    {tag}
                    <button
                      type="button"
                      onClick={() => removeTag(tag)}
                      className="ml-1 text-blue-600 hover:text-blue-800"
                    >
                      ×
                    </button>
                  </span>
                ))}
              </div>
            )}
          </div>
        </div>

        {/* Image Upload */}
        <div className="md:col-span-2">
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Product Image
          </label>
          <div className="space-y-4">
            {/* Upload Button */}
            <div className="flex items-center gap-4">
              <label className="cursor-pointer bg-gray-100 hover:bg-gray-200 px-4 py-2 rounded-lg bo
                <input
                  type="file"
                  accept="image/*"
                  onChange={handleImageUpload}
                  className="hidden"
                />
                <div className="flex items-center gap-2 text-gray-600">
                  <svg
                    className="w-5 h-5"
                    fill="none"
                    stroke="currentColor"
                    viewBox="0 0 24 24"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M7 16a4 4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M15 13l-3-3m0 0l-
                    />
                  </svg>
                  {uploadingImage ? "Uploading..." : "Choose Image"}
                </div>
              </label>
              <span className="text-sm text-gray-500">
                Max 2MB • JPG, PNG, GIF
              </span>
            </div>

            {/* Image Preview */}
            {imagePreview && (
              <div className="relative inline-block">
                <img
                  src={imagePreview}
                  alt="Preview"
                  className="w-32 h-32 object-cover rounded-lg border"
                />
                <button
                  type="button"
                  onClick={() => {
                    setImagePreview("");
                    setFormData((prev) => ({ ...prev, image: "" }));
                  }}
                  className="absolute -top-2 -right-2 bg-red-500 text-white rounded-full w-6 h-6 fle
              >
                ×
              </button>
            </div>
          )}

          {errors.image && (
            <p className="text-red-500 text-sm">{errors.image}</p>
          )}
        </div>
      </div>
    </div>

    {/* Description */}
    <div>
      <label className="block text-sm font-medium text-gray-700 mb-2">
        Description * (Max 60 characters)
      </label>
      <div className="relative">
        <textarea
          name="description"
          value={formData.description}
          onChange={handleChange}
          rows={3}
          maxLength={60}
          className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 focus:bor
            }`}
          placeholder="Enter item description (max 60 characters)"
        />
        <div className="absolute bottom-2 right-2 text-sm text-gray-400">
          {formData.description.length}/60
        </div>
      </div>
      {errors.description && (
        <p className="text-red-500 text-sm mt-1">{errors.description}</p>
      )}
    </div>

    {/* Checkboxes */}
    <div className="flex flex-wrap gap-6">
      <label className="flex items-center gap-2">
        <input
          type="checkbox"
          name="isVeg"
          checked={formData.isVeg}
          onChange={handleChange}
          className="rounded text-red-600 focus:ring-red-500"
        />
        <span className="text-sm font-medium text-gray-700">Vegetarian</span>
      </label>

      <label className="flex items-center gap-2">
        <input
          type="checkbox"
          name="isAvailable"
          checked={formData.isAvailable}
          onChange={handleChange}
          className="rounded text-red-600 focus:ring-red-500"
        />
        <span className="text-sm font-medium text-gray-700">Available</span>
      </label>
    </div>

    {/* Form Actions */}
    <div className="flex gap-4 pt-4 border-t">
      <button
        type="submit"
        className="flex-1 px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 font-medium"
      >
        {item ? "Update Item" : "Add Item"}
      </button>
      <button
        type="button"
        onClick={onCancel}
```

```
          className="flex-1 px-4 py-2 bg-gray-200 text-gray-800 rounded-lg hover:bg-gray-300 font-me
        >
          Cancel
        </button>
      </div>
    </form>
  );
}
```

13. src/components/auth/ProtectedRoute.jsx

```
import React from "react";
import { useAuth } from "../../context/AuthContext.jsx";

export default function ProtectedRoute({
  children,
  requiredRole = null,
  fallback = null,
}) {
  const { isAuthenticated, user, isLoading } = useAuth();

  // Show loading spinner while checking authentication
  if (isLoading) {
    return (
      <div className="min-h-screen flex items-center justify-center bg-light-gray">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-dark-red mx-auto mb-
          <p className="text-gray-600">Loading...</p>
        </div>
      </div>
    );
  }

  // Check if user is authenticated
  if (!isAuthenticated) {
    return (
      fallback || (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4 text-center">
            <div className="mb-6">
              <div className="w-16 h-16 bg-error-red rounded-full flex items-center justify-center m
                <span className="text-white text-2xl">🔒</span>
              </div>
              <h2 className="text-2xl font-bold text-dark-red mb-2">
                Access Denied
              </h2>
              <p className="text-gray-600">
                Please log in to access this page.
              </p>
            </div>
            <button
              onClick={() => (window.location.href = "/")}
              className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-red trans
            >
              Go to Home
            </button>
          </div>
        </div>
      )
    );
  }

  // Check role-based access
  if (requiredRole && user?.role !== requiredRole) {
    return (
      fallback || (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4 text-center">
            <div className="mb-6">
              <div className="w-16 h-16 bg-warn-yellow rounded-full flex items-center justify-center
```

```
                <span className="text-white text-2xl">⚠️</span>
              </div>
              <h2 className="text-2xl font-bold text-error-red mb-2">
                Unauthorized Access
              </h2>
              <p className="text-gray-600">
                You don't have permission to access this page.
              </p>
            </div>
            <div className="space-y-2">
              <button
                onClick={() => window.history.back()}
                className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-600 tran
              >
                Go Back
              </button>
              <button
                onClick={() => (window.location.href = "/")}
                className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-red tra
              >
                Go to Home
              </button>
            </div>
          </div>
        </div>
      )
    );
  }

  // User is authenticated and authorized
  return children;
}
```

14. src/components/customer/FoodCard.jsx

```
 import React from "react";
import { useCart } from "../../context/CartContext.jsx";
import { Button } from "../shared/Layout.jsx";

export default function FoodCard({ item }) {
  const { addItem, getItemQuantity, updateQuantity } = useCart();
  const quantity = getItemQuantity(item.id);

  const handleAddToCart = () => {
    addItem(item);
  };

  const handleIncrement = () => {
    updateQuantity(item.id, quantity + 1);
  };

  const handleDecrement = () => {
    if (quantity > 1) {
      updateQuantity(item.id, quantity - 1);
    } else {
      updateQuantity(item.id, 0);
    }
  };

  return (
    <div className="bg-white rounded-lg shadow-subtle overflow-hidden hover:shadow-lg transition-sha
      {/* Image */}
      <div className="relative h-48 sm:h-52 md:h-48 lg:h-52 overflow-hidden flex-shrink-0">
        <img
          src={item.image}
          alt={item.name}
          className="w-full h-full object-cover hover:scale-105 transition-transform duration-300"
          referrerPolicy="no-referrer"
          onError={(e) => {
            e.target.onerror = null;
            e.target.src = "https://images.unsplash.com/photo-1546833999-b9f581a1996d?w=400"; // Fal
          }}
        />

        {/* Type Badge */}
        <div className="absolute top-2 left-2">
          <span
            className={`
            px-2 py-1 rounded-full text-xs font-medium
            ${item.type === "Veg"
                ? "bg-success-green text-white"
                : "bg-error-red text-white"
              }
          `}
          >
            {item.type}
          </span>
        </div>

        {/* Rating Badge */}
        <div className="absolute top-2 right-2 bg-black bg-opacity-70 text-white px-2 py-1 rounded-f
          <span className="text-golden-yellow mr-1">★</span>
          {item.rating}
        </div>
      </div>

      {/* Content */}
      <div className="p-4 flex-1 flex flex-col">
        <div className="mb-3 flex-1">
          <h3 className="text-lg font-semibold text-gray-800 mb-2 line-clamp-1">
            {item.name}
          </h3>
          <p className="text-gray-600 text-sm line-clamp-2 min-h-[2.5rem]">
            {item.description}
          </p>
        </div>

        {/* Tags */}
        {item.tags && item.tags.length > 0 && (
          <div className="flex flex-wrap gap-1 mb-3 min-h-[1.5rem]">
            {item.tags.slice(0, 3).map((tag, index) => (
              <span
                key={index}
                className="bg-light-gray text-gray-600 px-2 py-1 rounded-full text-xs"
              >
                {tag}
              </span>
            ))}
          </div>
        )}

        {/* Price and Actions */}
        <div className="flex items-center justify-between mt-auto">
          <div className="flex-1">
            <span className="text-xl font-bold text-dark-red">
              ₹{item.price}
            </span>
            {item.preparationTime && (
              <p className="text-xs text-gray-500 mt-1">
                {item.preparationTime} mins
              </p>
            )}
          </div>

          {/* Add to Cart / Quantity Controls */}
          <div className="flex items-center ml-2">
            {quantity === 0 ? (
              <Button
                onClick={handleAddToCart}
                variant="primary"
                size="sm"
                className="px-3 py-1 text-sm min-w-[60px]"
              >
                Add
            </Button>
          ) : (
            <div className="flex items-center space-x-2">
              <button
                onClick={handleDecrement}
                className="w-7 h-7 rounded-full bg-gray-200 hover:bg-gray-300 flex items-center ju
              >
                -
              </button>
              <span className="font-medium text-dark-red min-w-[20px] text-center text-sm">
                {quantity}
              </span>
              <button
                onClick={handleIncrement}
                className="w-7 h-7 rounded-full bg-dark-red hover:bg-hover-red flex items-center j
              >
                +
              </button>
            </div>
          )}
        </div>
      </div>

      {/* Reviews */}
      {item.reviewCount > 0 && (
        <div className="mt-3 pt-2 border-t border-gray-100">
          <p className="text-xs text-gray-500">
            {item.reviewCount} review{item.reviewCount !== 1 ? "s" : ""}
          </p>
        </div>
      )}
    </div>
  </div>
  );
}
```

15. src/components/customer/CartItem.jsx

```
 import React, { useState } from "react";
import { useCart } from "../../context/CartContext.jsx";

export default function CartItem({ item }) {
  const { updateQuantity, removeItem } = useCart();
  const [specialInstructions, setSpecialInstructions] = useState(
    item.specialInstructions || ""
  );
  const [showInstructions, setShowInstructions] = useState(false);

  const handleQuantityChange = (newQuantity) => {
    if (newQuantity <= 0) {
      removeItem(item.id);
    } else {
      updateQuantity(item.id, newQuantity);
    }
  };

  const handleInstructionsChange = (instructions) => {
    setSpecialInstructions(instructions);
    // Update the item with special instructions
    // This would typically be handled by the cart context
  };

  return (
    <div className="bg-white rounded-lg shadow-sm border border-gray-200 p-3 md:p-4 mb-3 md:mb-4">
      <div className="flex items-start space-x-3 md:space-x-4">
        {/* Item Image */}
        <div className="flex-shrink-0">
          <img
            src={item.image}
            alt={item.name}
            className="w-16 h-16 md:w-20 md:h-20 object-cover rounded-lg"
          />
        </div>

        {/* Item Details */}
        <div className="flex-1 min-w-0">
          <div className="flex items-start justify-between">
            <div className="flex-1 min-w-0">
              <h3 className="text-base md:text-lg font-semibold text-gray-800 mb-1 truncate">
                {item.name}
              </h3>

              {/* Type Badge */}
              <div className="flex items-center space-x-1 md:space-x-2 mb-2">
                <span
                  className={`
                  inline-flex items-center px-1.5 md:px-2 py-1 rounded-full text-xs font-medium
                  ${
                    item.type === "Veg"
                      ? "bg-success-green text-white"
                      : "bg-error-red text-white"
                  }
                `}
                >
                  <span
                    className={`
                    w-1.5 h-1.5 md:w-2 md:h-2 rounded-full mr-1
                    ${item.type === "Veg" ? "bg-white" : "bg-white"}
                  `}
                  ></span>
                  {item.type}
                </span>

                {item.category && (
                  <span className="text-xs text-gray-500 bg-gray-100 px-1.5 md:px-2 py-1 rounded-ful
                    {item.category}
                  </span>
                )}
              </div>

              {/* Price */}
              <div className="flex items-center space-x-2 mb-2 md:mb-3">
                <span className="text-lg md:text-xl font-bold text-dark-red">
                  ₹{item.price}
                </span>
                <span className="text-xs md:text-sm text-gray-500">each</span>
              </div>
            </div>

            {/* Remove Button */}
            <button
              onClick={() => removeItem(item.id)}
              className="text-gray-400 hover:text-error-red transition-colors p-1 flex-shrink-0"
              title="Remove item"
            >
              <svg
                className="w-4 h-4 md:w-5 md:h-5"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M19 7l-.867 12.142A2 2 0 0116.138 21H7.862a2 2 0 01-1.995-1.858L5 7m5 4v6m4-6v6
                />
              </svg>
            </button>
          </div>

          {/* Quantity Controls */}
          <div className="flex flex-col sm:flex-row sm:items-center justify-between gap-3 sm:gap-0">
            <div className="flex items-center space-x-2 md:space-x-3">
              <span className="text-xs md:text-sm font-medium text-gray-700">
                Quantity:
              </span>
              <div className="flex items-center space-x-2">
                <button
                  onClick={() => handleQuantityChange(item.quantity - 1)}
                  className="w-7 h-7 md:w-8 md:h-8 rounded-full bg-gray-200 hover:bg-gray-300 flex i
                  disabled={item.quantity <= 1}
                >
                  -
                </button>
                <span className="font-medium text-dark-red min-w-[30px] text-center text-sm md:text-
                  {item.quantity}
                </span>
                <button
                  onClick={() => handleQuantityChange(item.quantity + 1)}
                  className="w-7 h-7 md:w-8 md:h-8 rounded-full bg-dark-red hover:bg-hover-red flex
                >
                  +
                </button>
              </div>
            </div>

            {/* Subtotal */}
            <div className="text-left sm:text-right">
              <div className="text-base md:text-lg font-bold text-dark-red">
                ₹{(item.price * item.quantity).toFixed(2)}
              </div>
              <div className="text-xs text-gray-500">
                {item.quantity} × ₹{item.price}
              </div>
            </div>
          </div>

          {/* Special Instructions */}
          <div className="mt-3 pt-3 border-t border-gray-100">
            <button
              onClick={() => setShowInstructions(!showInstructions)}
              className="text-xs md:text-sm text-dark-red hover:text-hover-red transition-colors fle
            >
              <svg
                className="w-3 h-3 md:w-4 md:h-4 mr-1"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2h11a2 2 0 002-2v-5m-1.414-9.414a2 2 0 112.82
                />
              </svg>
              {showInstructions ? "Hide" : "Add"} special instructions
            </button>

            {showInstructions && (
              <div className="mt-2">
                <textarea
                  value={specialInstructions}
                  onChange={(e) => handleInstructionsChange(e.target.value)}
                  placeholder="Any special requests for this item? (e.g., less spicy, extra sauce)"
                  className="w-full px-2 md:px-3 py-2 border border-gray-300 rounded-lg focus:outlin
                  rows={2}
                />
              </div>
            )}

            {specialInstructions && !showInstructions && (
              <div className="mt-2 text-xs md:text-sm text-gray-600 bg-gray-50 p-2 rounded">
                <strong>Note:</strong> {specialInstructions}
              </div>
            )}
          </div>
```

```
          </div>
        </div>
      </div>
    );
  }
```

16. src/components/customer/layout/Header.jsx

```
 import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "../../../context/AuthContext.jsx";
import { useCart } from "../../../context/CartContext.jsx";

export default function Header({ onOpenAuth }) {
  const [isProfileDropdownOpen, setIsProfileDropdownOpen] = useState(false);
  const { isAuthenticated, user, logout } = useAuth();
  const { itemCount } = useCart();
  const navigate = useNavigate();

  const handleLogout = async () => {
    await logout();
    setIsProfileDropdownOpen(false);
    navigate("/");
  };

  return (
    <header className="bg-white shadow-sm sticky top-0 z-40">
      <div className="container mx-auto px-4">
        <div className="flex items-center justify-between h-16">
          {/* Logo */}
          <Link to="/" className="flex items-center space-x-2">
            <img
              src="https://cornflowerblue-lion-884998.hostingersite.com/wp-content/uploads/2025/07/l
              alt="The Food City"
              className="h-10 w-10 object-contain"
            />
            <span className="text-xl font-bold text-dark-red hidden sm:block">
              The Food City
            </span>
          </Link>

          {/* Desktop Navigation */}
          <nav className="hidden md:flex items-center space-x-6">
            <Link
              to="/"
              className="text-gray-700 hover:text-dark-red transition-colors"
            >
              Home
            </Link>
            <Link
              to="/menu"
              className="text-gray-700 hover:text-dark-red transition-colors"
            >
              Menu
            </Link>
            <Link
              to="/deals"
              className="text-gray-700 hover:text-dark-red transition-colors"
            >
              Deals
            </Link>
            <Link
              to="/about"
              className="text-gray-700 hover:text-dark-red transition-colors"
            >
              About
            </Link>
            <Link
              to="/contact"
              className="text-gray-700 hover:text-dark-red transition-colors"
        >
          Contact
        </Link>
      </nav>

      {/* Right side actions */}
      <div className="flex items-center space-x-4">
        {/* Cart Icon - Hidden on mobile since it's in bottom nav */}
        <Link
          to="/cart"
          className="relative p-2 text-gray-700 hover:text-dark-red transition-colors hidden md:
        >
          <svg
            className="w-6 h-6"
            fill="none"
            stroke="currentColor"
            viewBox="0 0 24 24"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M3 3h2l.4 2M7 13h10l4-8H5.4m0 0L7 13m0 0l-2.5 5M7 13l2.5 5m0 0h8"
            />
          </svg>
          {itemCount > 0 && (
            <span className="absolute -top-1 -right-1 bg-error-red text-white text-xs rounded-fu
              {itemCount > 99 ? "99+" : itemCount}
            </span>
          )}
        </Link>

        {/* User Menu */}
        {isAuthenticated ? (
          <div className="relative">
            <button
              onClick={() =>
                setIsProfileDropdownOpen(!isProfileDropdownOpen)
              }
              className="flex items-center space-x-2 p-2 rounded-lg hover:bg-gray-100 transition
            >
              <div className="w-8 h-8 bg-dark-red rounded-full flex items-center justify-center"
                <span className="text-white text-sm font-medium">
                  {user?.name?.charAt(0).toUpperCase()}
                </span>
              </div>
              <span className="hidden sm:block text-gray-700">
                {user?.name}
              </span>
              <svg
                className="w-4 h-4 text-gray-500"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M19 9l-7 7-7-7"
                />
              </svg>
            </button>

            {/* Dropdown Menu */}
            {isProfileDropdownOpen && (
              <div className="absolute right-0 mt-2 w-48 bg-white rounded-lg shadow-lg border bo
                <Link
                  to="/profile"
                  onClick={() => setIsProfileDropdownOpen(false)}
                  className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
                >
                  Profile
                </Link>
                <Link
                  to="/orders"
                  onClick={() => setIsProfileDropdownOpen(false)}
                  className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
                >
                  My Orders
                </Link>
                {user?.role === "admin" && (
                  <Link
                    to="/admin"
                    onClick={() => setIsProfileDropdownOpen(false)}
                    className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
                  >
                    Admin Panel
                  </Link>
                )}
                <hr className="my-1" />
                <button
                  onClick={handleLogout}
                  className="block w-full text-left px-4 py-2 text-sm text-gray-700 hover:bg-gra
                >
                  Logout
                </button>
              </div>
            )}
          </div>
        ) : (
          <div className="flex items-center space-x-2">
            <button
              onClick={() => onOpenAuth("login")}
              className="text-gray-700 hover:text-dark-red transition-colors"
            >
              Login
            </button>
            <button
              onClick={() => onOpenAuth("register")}
              className="bg-dark-red text-white px-4 py-2 rounded-lg hover:bg-hover-red transiti
            >
              Sign Up
            </button>
          </div>
        )}
      </div>
    </div>
  </div>
</header>
);
}
```

17. src/components/customer/layout/Footer.jsx

```
 import React from "react";
import { Link } from "react-router-dom";

export default function Footer() {
  return (
    <footer className="bg-gray-800 text-white">
      <div className="container mx-auto px-4 py-8">
        <div className="grid grid-cols-1 md:grid-cols-4 gap-8">
          {/* Company Info */}
          <div>
            <div className="flex items-center space-x-2 mb-4">
              <img
                src="https://cornflowerblue-lion-884998.hostingersite.com/wp-content/uploads/2025/07
                alt="The Food City"
                className="h-8 w-8 object-contain"
              />
              <span className="text-lg font-bold">The Food City</span>
            </div>
            <p className="text-gray-300 text-sm mb-4">
            Delicious food delivered to your doorstep. Experience the best
            flavors from around the world.
          </p>
          <div className="flex space-x-4">
            <a
              href="#"
              className="text-gray-300 hover:text-white transition-colors"
            >
              <svg
                className="w-5 h-5"
                fill="currentColor"
                viewBox="0 0 24 24"
              >
                <path d="M24 4.557c-.883.392-1.832.656-2.828.775 1.017-.609 1.798-1.574 2.165-2.72
              </svg>
            </a>
            <a
              href="#"
              className="text-gray-300 hover:text-white transition-colors"
            >
              <svg
                className="w-5 h-5"
                fill="currentColor"
                viewBox="0 0 24 24"
              >
                <path d="M22.46 6c-.77.35-1.6.58-2.46.69.88-.53 1.56-1.37 1.88-2.38-.83.5-1.75.85-
              </svg>
            </a>
            <a
              href="#"
              className="text-gray-300 hover:text-white transition-colors"
            >
              <svg
                className="w-5 h-5"
                fill="currentColor"
                viewBox="0 0 24 24"
              >
                <path d="M12.017 0C5.396 0 .029 5.367.029 11.987c0 5.079 3.158 9.417 7.618 11.174-
              </svg>
            </a>
          </div>
        </div>

        {/* Quick Links */}
        <div>
          <h3 className="text-lg font-semibold mb-4">Quick Links</h3>
          <ul className="space-y-2">
            <li>
              <Link
                to="/"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Home
              </Link>
            </li>
            <li>
              <Link
                to="/menu"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Menu
              </Link>
            </li>
            <li>
              <Link
                to="/deals"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Deals & Offers
              </Link>
            </li>
            <li>
              <Link
                to="/about"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                About Us
              </Link>
            </li>
            <li>
              <Link
                to="/contact"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Contact Us
              </Link>
            </li>
          </ul>
        </div>

        {/* Legal */}
        <div>
          <h3 className="text-lg font-semibold mb-4">Legal</h3>
          <ul className="space-y-2">
            <li>
              <Link
                to="/terms"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Terms & Conditions
              </Link>
            </li>
            <li>
              <Link
                to="/privacy"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Privacy Policy
              </Link>
            </li>
            <li>
              <Link
                to="/refund"
                className="text-gray-300 hover:text-white transition-colors text-sm"
              >
                Refund Policy
              </Link>
            </li>
          </ul>
        </div>

        {/* Contact Info */}
        <div>
          <h3 className="text-lg font-semibold mb-4">Contact Info</h3>
          <div className="space-y-2 text-sm text-gray-300">
            <div className="flex items-center space-x-2">
              <svg
                className="w-4 h-4"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M17.657 16.657L13.414 20.9a1.998 1.998 0 01-2.827 0l-4.244-4.243a8 8 0 1111.3
                />
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M15 11a3 3 0 11-6 0 3 3 0 016 0z"
                />
              </svg>
              <span>123 Food Street, Mumbai, India</span>
            </div>
            <div className="flex items-center space-x-2">
```

```
                <svg
                  className="w-4 h-4"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M3 5a2 2 0 012-2h3.28a1 1 0 01.948.684l1.498 4.493a1 1 0 01-.502 1.21l-2.257
                  />
                </svg>
                <span>+91 98765 43210</span>
              </div>
              <div className="flex items-center space-x-2">
                <svg
                  className="w-4 h-4"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M3 8l7.89 5.26a2 2 0 002.22 0L21 8M5 19h14a2 2 0 002-2V7a2 2 0 00-2-2H5a2 2 0
                  />
                </svg>
                <span>info@thefoodcity.com</span>
              </div>
            </div>
          </div>
        </div>

        {/* Bottom Bar */}
        <div className="border-t border-gray-700 mt-8 pt-8 flex flex-col md:flex-row justify-between
          <p className="text-gray-300 text-sm">
            © 2025 The Food City. All rights reserved.
          </p>
          <p className="text-gray-300 text-sm mt-2 md:mt-0">
            Designed and developed by{" "}
            <a
              href="https://enegixwebsolutions.com/"
              target="_blank"
              rel="noopener noreferrer"
              className="text-white hover:text-dark-red transition-colors font-medium"
            >
              Enegix Web Solutions
            </a>
          </p>
        </div>
      </div>
    </footer>
  );
}
```

18. src/components/customer/layout/CustomerLayout.jsx

```
 import React, { useState } from "react";
import { Outlet } from "react-router-dom";
import Header from "./Header.jsx";
import BottomNavigation from "./BottomNavigation.jsx";
import Footer from "./Footer.jsx";
import AuthModal from "../../auth/AuthModal.jsx";
import { ToastProvider } from "../../shared/Toast.jsx";

export default function CustomerLayout() {
  const [showAuthModal, setShowAuthModal] = useState(false);
  const [authMode, setAuthMode] = useState("login");
  const openAuthModal = (mode = "login") => {
    setAuthMode(mode);
    setShowAuthModal(true);
  };

  return (
    <ToastProvider>
      <div className="min-h-screen bg-light-gray flex flex-col">
        {/* Header */}
        <Header onOpenAuth={openAuthModal} />

        {/* Main Content */}
        <main className="flex-1 pb-20 md:pb-8">
          <Outlet />
        </main>

        {/* Footer - Hidden on mobile, shown on desktop */}
        <div className="hidden md:block">
          <Footer />
        </div>

        {/* Bottom Navigation - Mobile only */}
        <div className="md:hidden">
          <BottomNavigation onOpenAuth={openAuthModal} />
        </div>

        {/* Auth Modal */}
        <AuthModal
          isOpen={showAuthModal}
          onClose={() => setShowAuthModal(false)}
          initialMode={authMode}
        />
      </div>
    </ToastProvider>
  );
}
```

19. src/pages/customer/HomePage.jsx

```
 import React, { useState, useEffect } from "react";
import { useSearchParams } from "react-router-dom";
import { Container } from "../../components/shared/Layout.jsx";
import { mockAPI } from "../../services/mockApi.js";
import HeroSection from "../../components/customer/HeroSection.jsx";
import SimpleSearchBox from "../../components/customer/SimpleSearchBox.jsx";
import SpecialOffersSlider from "../../components/customer/SpecialOffersSlider.jsx";
import CategorySlider from "../../components/customer/CategorySlider.jsx";
import { InlineLoader } from "../../components/shared/LoadingSpinner.jsx";
import {
  FaFire,
  FaBowlFood,
  FaHotjar,
  FaDrumstickBite,
  FaUtensils,
} from "react-icons/fa6";
import { FaPepperHot, FaClock, FaStar, FaDollarSign } from "react-icons/fa";

export default function HomePage() {
  const [menuItems, setMenuItems] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchParams] = useSearchParams();

  useEffect(() => {
    loadMenuItems();
  }, [searchParams]);

  const loadMenuItems = async () => {
    try {
      setLoading(true);
      const response = await mockAPI.getMenuItems();
      if (response.success) {
        setMenuItems(response.data);
      } else {
        setError("Failed to load menu items");
      }
    } catch (err) {
      setError("Something went wrong while loading menu items");
      console.error("Error loading menu items:", err?.message || String(err));
    } finally {
      setLoading(false);
    }
  };

  // Group items by category
  const groupedItems = menuItems.reduce((acc, item) => {
    if (!acc[item.category]) {
      acc[item.category] = [];
    }
    acc[item.category].push(item);
    return acc;
  }, {});

  // Get trending items (highest rated items)
  const trendingItems = menuItems
    .filter((item) => item.rating >= 4.3)
    .sort((a, b) => b.rating - a.rating)
    .slice(0, 8);

  if (loading) {
    return (
      <Container className="py-8">
        <InlineLoader text="Loading delicious food..." />
      </Container>
    );
  }

  if (error) {
    return (
      <Container className="py-8">
        <div className="text-center">
          <div className="bg-error-red text-white p-4 rounded-lg max-w-md mx-auto">
            <h3 className="font-semibold mb-2">Oops! Something went wrong</h3>
            <p className="text-sm">{error}</p>
            <button
              onClick={loadMenuItems}
              className="mt-3 bg-white text-error-red px-4 py-2 rounded font-medium hover:bg-gray-10
            >
              Try Again
            </button>
          </div>
        </div>
      </Container>
    );
  }

  return (
    <Container className="py-4 md:py-8">
      {/* Hero Section */}
      <HeroSection />

      {/* Simple Search Box */}
      <SimpleSearchBox />

      {/* Special Offers Slider */}
      <SpecialOffersSlider />

      {/* Trending Items */}
      {trendingItems.length > 0 && (
        <CategorySlider
          category="trending"
          title={
            <span className="flex items-center gap-2">
              <FaFire className="text-dark-red" />
              Trending Now
            </span>
          }
          items={trendingItems}
        />
      )}

      {/* Category Sections */}
      {Object.entries(groupedItems).map(([category, items]) => {
        // Category icons
        const categoryIcons = {
          Indian: <FaBowlFood className="text-dark-red" />,
          Chinese: <FaHotjar className="text-dark-red" />,
          South: <FaPepperHot className="text-dark-red" />,
          Tandoor: <FaDrumstickBite className="text-dark-red" />,
        };

        return (
          <CategorySlider
            key={category}
            category={category}
            title={
              <span className="flex items-center gap-2">
                {categoryIcons[category] || (
                  <FaUtensils className="text-dark-red" />
                )}
                {category} Cuisine
              </span>
            }
            items={items}
          />
        );
      })}

      {/* Quick Actions */}
      <div className="mt-12 bg-white rounded-lg shadow-subtle p-6">
        <h2 className="text-2xl font-bold text-dark-red mb-4 text-center">
          Why Choose The Food City?
        </h2>

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
          <div className="text-center">
            <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-center mx-au
              <FaClock className="text-2xl text-white" />
            </div>
            <h3 className="font-semibold text-gray-800 mb-2">Fast Delivery</h3>
            <p className="text-gray-600 text-sm">
              Get your food delivered in 30-45 minutes, hot and fresh.
            </p>
          </div>

          <div className="text-center">
            <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-center mx-au
              <FaStar className="text-2xl text-white" />
            </div>
            <h3 className="font-semibold text-gray-800 mb-2">Quality Food</h3>
            <p className="text-gray-600 text-sm">
              Fresh ingredients and authentic recipes for the best taste.
            </p>
          </div>

          <div className="text-center">
            <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-center mx-au
              <FaDollarSign className="text-2xl text-white" />
            </div>
            <h3 className="font-semibold text-gray-800 mb-2">Great Prices</h3>
            <p className="text-gray-600 text-sm">
              Affordable prices with regular deals and discounts.
            </p>
          </div>
        </div>
      </div>
    </Container>
```

```
  );
}
```

20. src/components/admin/layout/AdminLayout.jsx

```
 import React, { useState } from "react";
import { Outlet, Link, useLocation } from "react-router-dom";
import { Container } from "../../shared/Layout.jsx";
import { useAuth } from "../../../context/AuthContext.jsx";
import {
  FaTachometerAlt,
  FaShoppingBag,
  FaUtensils,
  FaTruck,
  FaUsers,
  FaChartLine,
  FaUser,
  FaSignOutAlt,
  FaUserTie,
  FaBars,
  FaTimes,
} from "react-icons/fa";

export default function AdminLayout() {
  const { user, logout } = useAuth();
  const location = useLocation();
  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  const menuItems = [
    { path: "/admin", icon: FaTachometerAlt, label: "Dashboard", exact: true },
    { path: "/admin/orders", icon: FaShoppingBag, label: "Orders" },
    { path: "/admin/menu", icon: FaUtensils, label: "Menu Management" },
    { path: "/admin/delivery", icon: FaUserTie, label: "Delivery Staff" },
    { path: "/admin/customers", icon: FaUsers, label: "Customers" },
    { path: "/admin/reports", icon: FaChartLine, label: "Reports" },
  ];

  const isActiveRoute = (path, exact = false) => {
    if (exact) {
      return location.pathname === path;
    }
    return location.pathname.startsWith(path);
  };

  const closeMobileMenu = () => {
    setIsMobileMenuOpen(false);
  };

  return (
    <div className="min-h-screen bg-light-gray">
      {/* Top Navigation */}
      <div className="bg-white shadow-sm border-b">
        <Container>
          <div className="flex items-center justify-between h-16">
            <div className="flex items-center gap-3">
              {/* Mobile Menu Button */}
              <button
                onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}
                className="lg:hidden p-2 text-gray-600 hover:text-dark-red"
              >
                {isMobileMenuOpen ? (
                  <FaTimes size={20} />
                ) : (
                  <FaBars size={20} />
                )}
              </button>

              <div className="text-lg lg:text-xl font-bold text-dark-red">
                The Food City
              </div>
            <div className="hidden sm:block text-xs lg:text-sm text-gray-500 bg-red-100 px-2 py-1
              Admin Panel
            </div>
          </div>
          <div className="flex items-center gap-2 lg:gap-4">
            <div className="flex items-center gap-2">
              <div className="w-7 h-7 lg:w-8 lg:h-8 bg-dark-red rounded-full flex items-center jus
                <FaUser className="text-white text-xs lg:text-sm" />
              </div>
              <div className="hidden md:block">
                <p className="text-sm font-medium text-gray-800">
                  {user?.name}
                </p>
                <p className="text-xs text-gray-500 capitalize">
                  {user?.role}
                </p>
              </div>
            </div>
            <button
              onClick={logout}
              className="flex items-center gap-1 lg:gap-2 text-gray-600 hover:text-red-600 transit
              title="Logout"
            >
              <FaSignOutAlt className="text-sm lg:text-base" />
              <span className="hidden lg:inline text-sm">Logout</span>
            </button>
          </div>
        </div>
      </Container>
    </div>

    <div className="flex relative">
      {/* Mobile Overlay */}
      {isMobileMenuOpen && (
        <div
          className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"
          onClick={closeMobileMenu}
        />
      )}

      {/* Sidebar */}
      <div
        className={`
        fixed lg:static inset-y-0 left-0 z-50
        w-64 bg-white shadow-lg lg:shadow-sm
        transform transition-transform duration-300 ease-in-out
        ${isMobileMenuOpen
            ? "translate-x-0"
            : "-translate-x-full lg:translate-x-0"
          }
        min-h-screen lg:min-h-0
      `}
      >
        {/* Mobile Close Button */}
        <div className="lg:hidden flex justify-end p-4">
          <button
            onClick={closeMobileMenu}
            className="p-2 text-gray-600 hover:text-dark-red"
          >
            <FaTimes size={18} />
          </button>
        </div>

        <nav className="p-4 pt-0 lg:pt-4">
          <ul className="space-y-2">
            {menuItems.map((item) => {
              const Icon = item.icon;
              return (
                <li key={item.path}>
                  <Link
                    to={item.path}
                    onClick={closeMobileMenu}
                    className={`flex items-center gap-3 px-4 py-3 rounded-lg transition-colors ${i
                        ? "bg-dark-red text-white"
                    : "text-gray-700 hover:bg-red-50 hover:text-dark-red"
                  }`}
                >
                  <Icon className="text-lg flex-shrink-0" />
                  <span className="text-sm lg:text-base">{item.label}</span>
                </Link>
              </li>
            );
          })}
        </ul>
      </nav>
    </div>

    {/* Main Content */}
    <div className="flex-1 lg:ml-0">
      <main className="p-3 sm:p-4 lg:p-6">
        <Outlet />
      </main>
    </div>
  </div>
</div>
  );
}
```

21. src/components/shared/ConfirmDialog.jsx

```
 import React from "react";
import { FaTrash, FaExclamationTriangle, FaInfoCircle } from "react-icons/fa";

export default function ConfirmDialog({
  isOpen,
  onClose,
  onConfirm,
  title = "Confirm Action",
  message = "Are you sure you want to proceed?",
  confirmText = "Confirm",
  cancelText = "Cancel",
  type = "warning", // 'warning', 'danger', 'info'
}) {
  if (!isOpen) return null;

  const typeStyles = {
    warning: {
      icon: <FaExclamationTriangle className="text-white" />,
      confirmButton: "bg-warm-yellow hover:bg-yellow-600 text-black",
      iconBg: "bg-warm-yellow",
    },
    danger: {
      icon: <FaTrash className="text-white" />,
      confirmButton: "bg-red-900 hover:bg-red-800 text-white",
      iconBg: "bg-red-900",
    },
    info: {
      icon: <FaInfoCircle className="text-white" />,
      confirmButton: "bg-info-blue hover:bg-blue-600 text-white",
      iconBg: "bg-info-blue",
    },
  };

  const currentStyle = typeStyles[type];

  const handleConfirm = () => {
    onConfirm();
    onClose();
  };

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">
      <div className="bg-white rounded-lg shadow-lg max-w-md w-full">
        <div className="p-6">
```

```
          {/* Icon and Title */}
          <div className="flex items-center mb-4">
            <div
              className={`w-10 h-10 ${currentStyle.iconBg} rounded-full flex items-center justify-ce
            >
              {currentStyle.icon}
            </div>
            <h3 className="text-lg font-semibold text-gray-900">{title}</h3>
          </div>

          {/* Message */}
          <p className="text-gray-600 mb-6">{message}</p>

          {/* Buttons */}
          <div className="flex space-x-3">
            <button
              onClick={onClose}
              className="flex-1 bg-gray-200 text-gray-800 py-2 px-4 rounded-lg hover:bg-gray-300 tra
            >
              {cancelText}
            </button>
            <button
              onClick={handleConfirm}
              className={`flex-1 py-2 px-4 rounded-lg transition-colors ${currentStyle.confirmButton
            >
              {confirmText}
            </button>
          </div>
        </div>
      </div>
    </div>
  );
}

// Hook for easier usage
export function useConfirmDialog() {
  const [dialog, setDialog] = React.useState({
    isOpen: false,
    title: "",
    message: "",
    onConfirm: () => {},
    type: "warning",
  });

  const showConfirm = (options) => {
    return new Promise((resolve) => {
      setDialog({
        isOpen: true,
        title: options.title || "Confirm Action",
        message: options.message || "Are you sure?",
        type: options.type || "warning",
        onConfirm: () => {
          resolve(true);
          setDialog((prev) => ({ ...prev, isOpen: false }));
        },
      });
    });
  };

  const closeDialog = () => {
    setDialog((prev) => ({ ...prev, isOpen: false }));
  };

  const ConfirmDialogComponent = () => (
    <ConfirmDialog
      isOpen={dialog.isOpen}
      onClose={closeDialog}
      onConfirm={dialog.onConfirm}
      title={dialog.title}
      message={dialog.message}
      type={dialog.type}
    />
  );
  return { showConfirm, ConfirmDialogComponent };
}
```

22. src/components/shared/Toast.jsx

```
 import React, { useState, useEffect } from "react";

// Toast context for managing toasts globally
const ToastContext = React.createContext();

export function ToastProvider({ children }) {
  const [toasts, setToasts] = useState([]);

  const addToast = (message, type = "info", duration = 3000) => {
    const id = Date.now();
    const toast = { id, message, type, duration };

    setToasts((prev) => [...prev, toast]);

    // Auto remove toast after duration
    setTimeout(() => {
      removeToast(id);
    }, duration);
  };

  const removeToast = (id) => {
    setToasts((prev) => prev.filter((toast) => toast.id !== id));
  };

  const value = {
    addToast,
    removeToast,
  };

  return (
    <ToastContext.Provider value={value}>
      {children}
      <ToastContainer toasts={toasts} onRemove={removeToast} />
    </ToastContext.Provider>
  );
}

// Hook to use toast
export function useToast() {
  const context = React.useContext(ToastContext);
  if (!context) {
    throw new Error("useToast must be used within a ToastProvider");
  }
  return context;
}

// Toast container component
function ToastContainer({ toasts, onRemove }) {
  if (toasts.length === 0) return null;

  return (
    <div className="fixed top-4 right-4 z-50 space-y-2">
      {toasts.map((toast) => (
        <Toast
          key={toast.id}
          message={toast.message}
          type={toast.type}
          onClose={() => onRemove(toast.id)}
        />
      ))}
    </div>
  );
}
```

```
// Individual toast component
```

```
function Toast({ message, type, onClose }) {
  const [isVisible, setIsVisible] = useState(false);

  useEffect(() => {
    // Trigger animation
    setTimeout(() => setIsVisible(true), 10);
  }, []);

  const handleClose = () => {
    setIsVisible(false);
    setTimeout(onClose, 300); // Wait for animation to complete
  };

  const typeStyles = {
    success: "bg-success-green text-white",
    error: "bg-error-red text-white",
    warning: "bg-warm-yellow text-black",
    info: "bg-info-blue text-white",
  };

  const icons = {
    success: "✓",
    error: "✕",
    warning: "⚠",
    info: "ℹ",
  };

  return (
    <div
      className={`
        transform transition-all duration-300 ease-in-out
        ${
          isVisible ? "translate-x-0 opacity-100" : "translate-x-full opacity-0"
        }
        ${typeStyles[type]}
        px-4 py-3 rounded-lg shadow-lg max-w-sm min-w-[300px]
        flex items-center justify-between
      `}
    >
      <div className="flex items-center">
        <span className="mr-2 text-lg">{icons[type]}</span>
        <span className="text-sm font-medium">{message}</span>
      </div>
      <button
        onClick={handleClose}
        className="ml-4 text-lg hover:opacity-70 transition-opacity"
      >
        ×
      </button>
    </div>
  );
}

export default Toast;
```

23. src/components/shared/Layout.jsx

```
 import React from "react";

// Main layout wrapper
export function Layout({ children, className = "" }) {
  return (
    <div className={`min-h-screen bg-light-gray ${className}`}>{children}</div>
  );
}

// Container component
export function Container({ children, size = "default", className = "" }) {
  const sizeClasses = {
    sm: "max-w-2xl",
    default: "max-w-6xl",
    lg: "max-w-7xl",
    full: "max-w-full",
  };

  return (
    <div className={`container mx-auto px-4 ${sizeClasses[size]} ${className}`}>
      {children}
    </div>
  );
}

// Card component
export function Card({ children, className = "", padding = "default" }) {
  const paddingClasses = {
    none: "",
    sm: "p-4",
    default: "p-6",
    lg: "p-8",
  };

  return (
    <div
      className={`bg-white rounded-lg shadow-subtle ${paddingClasses[padding]} ${className}`}
    >
      {children}
    </div>
  );
}

// Section component
export function Section({ children, className = "", padding = "default" }) {
  const paddingClasses = {
    none: "",
    sm: "py-4",
    default: "py-8",
    lg: "py-12",
  };

  return (
    <section className={`${paddingClasses[padding]} ${className}`}>
      {children}
    </section>
  );
}

// Grid component
export function Grid({ children, cols = 1, gap = 4, className = "" }) {
  const colClasses = {
    1: "grid-cols-1",
    2: "grid-cols-1 sm:grid-cols-2",
    3: "grid-cols-1 sm:grid-cols-2 lg:grid-cols-3",
    4: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4",
    5: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-5",
    6: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-6",
  };

  const gapClasses = {
    2: "gap-2",
    4: "gap-4",
    6: "gap-6",
    8: "gap-8",
  };

  return (
    <div className={`grid ${colClasses[cols]} ${gapClasses[gap]} ${className}`}>
      {children}
    </div>
  );
}

// Flex component
export function Flex({
  children,
  direction = "row",
  align = "start",
  justify = "start",
  wrap = false,
  gap = 0,
  className = "",
}) {
  const directionClasses = {
    row: "flex-row",
    col: "flex-col",
    "row-reverse": "flex-row-reverse",
    "col-reverse": "flex-col-reverse",
  };

  const alignClasses = {
    start: "items-start",
    center: "items-center",
    end: "items-end",
    stretch: "items-stretch",
  };

  const justifyClasses = {
    start: "justify-start",
    center: "justify-center",
    end: "justify-end",
    between: "justify-between",
    around: "justify-around",
    evenly: "justify-evenly",
  };

  const gapClasses = {
    0: "",
    1: "gap-1",
    2: "gap-2",
    3: "gap-3",
    4: "gap-4",
    6: "gap-6",
    8: "gap-8",
  };

  return (
    <div
      className={`
      flex
      ${directionClasses[direction]}
      ${alignClasses[align]}
      ${justifyClasses[justify]}
      ${wrap ? "flex-wrap" : ""}
      ${gapClasses[gap]}
      ${className}
    `}
    >
      {children}
    </div>
  );
}

// Button component
export function Button({
  children,
  variant = "primary",
  size = "md",
  disabled = false,
  loading = false,
  fullWidth = false,
  onClick,
  type = "button",
  className = "",
  ...props
}) {
  const baseClasses =
    "font-medium rounded-lg transition-colors focus:outline-none focus:ring-2 focus:ring-offset-2 di
  const variantClasses = {
    primary: "bg-dark-red text-white hover:bg-hover-red focus:ring-dark-red",
    secondary:
      "bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500",
    success:
      "bg-success-green text-white hover:bg-green-600 focus:ring-success-green",
    danger: "bg-error-red text-white hover:bg-red-600 focus:ring-error-red",
    warning:
      "bg-warm-yellow text-black hover:bg-yellow-600 focus:ring-warm-yellow",
    outline:
      "border-2 border-dark-red text-dark-red hover:bg-dark-red hover:text-white focus:ring-dark-red
  };

  const sizeClasses = {
    sm: "px-3 py-1.5 text-sm",
    md: "px-4 py-2 text-sm",
    lg: "px-6 py-3 text-base",
  };

  return (
    <button
      type={type}
      onClick={onClick}
      disabled={disabled || loading}
      className={`
        ${baseClasses}
        ${variantClasses[variant]}
        ${sizeClasses[size]}
        ${fullWidth ? "w-full" : ""}
        ${className}
      `}
      {...props}
    >
      {loading ? (
        <div className="flex items-center justify-center">
          <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-current mr-2"></div>
          Loading...
        </div>
      ) : (
        children
      )}
    </button>
  );
}
```

24. src/components/shared/ErrorBoundary.jsx

```
 import React from "react";

class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null, errorInfo: null };
  }

  static getDerivedStateFromError(error) {
    // Update state so the next render will show the fallback UI
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // Log error details
    console.error(
      "ErrorBoundary caught an error:",
      error?.message || String(error),
      errorInfo?.componentStack || "No stack trace"
    );

    this.setState({
      error: error,
      errorInfo: errorInfo,
    });
  }

  render() {
    if (this.state.hasError) {
      // Custom fallback UI
      return (
        <div className="min-h-screen flex items-center justify-center bg-light-gray p-4">
          <div className="bg-white rounded-lg shadow-subtle p-8 max-w-md w-full text-center">
            <div className="mb-6">
              <div className="w-16 h-16 bg-error-red rounded-full flex items-center justify-center m
                <span className="text-white text-2xl">⚠️</span>
              </div>
              <h2 className="text-2xl font-bold text-error-red mb-2">
                Oops! Something went wrong
              </h2>
              <p className="text-gray-600 mb-4">
                We're sorry, but something unexpected happened. Please try
                refreshing the page.
              </p>
            </div>

            <div className="space-y-3">
              <button
                onClick={() => window.location.reload()}
                className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-red tra
              >
                Refresh Page
              </button>
              <button
                onClick={() => (window.location.href = "/")}
                className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-600 tran
              >
                Go to Home
              </button>
            </div>

            {/* Show error details in development */}
            {process.env.NODE_ENV === "development" && (
              <details className="mt-6 text-left">
                <summary className="cursor-pointer text-sm text-gray-500 hover:text-gray-700">
                  Error Details (Development Only)
                </summary>
                <div className="mt-2 p-3 bg-gray-100 rounded text-xs overflow-auto max-h-40">
                  <div className="mb-2">
                    <strong>Error:</strong>{" "}
                    {this.state.error && this.state.error.toString()}
                  </div>
                  <div>
                    <strong>Stack Trace:</strong>
                    <pre className="whitespace-pre-wrap">
                      {this.state.errorInfo?.componentStack ||
                        "No stack trace available"}
                    </pre>
                  </div>
                </div>
              </details>
            )}
          </div>
        </div>
      );
    }

    // No error, render children normally
    return this.props.children;
  }
}

export default ErrorBoundary;
```

25. src/components/shared/FormInput.jsx

```
 import React, { useState } from "react";

// Basic input component
export function FormInput({
  label,
  name,
  type = "text",
  value,
  onChange,
  error,
  placeholder,
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  const [showPassword, setShowPassword] = useState(false);

  const inputType = type === "password" && showPassword ? "text" : type;

  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <div className="relative">
        <input
          type={inputType}
          name={name}
          value={value}
          onChange={onChange}
          placeholder={placeholder}
          required={required}
          disabled={disabled}
          className={`
            w-full px-3 py-2 border rounded-lg transition-colors
            focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
            disabled:bg-gray-100 disabled:cursor-not-allowed
            ${
              error
                ? "border-error-red focus:ring-error-red"
                : "border-gray-300 hover:border-gray-400"
            }
          `}
          {...props}
        />

        {type === "password" && (
          <button
            type="button"
            onClick={() => setShowPassword(!showPassword)}
            className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-400 hover:text-
          >
            {showPassword ? "👁️" : "👁️‍🗨️"}
          </button>
        )}
      </div>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Textarea component
export function FormTextarea({
  label,
  name,
  value,
  onChange,
  error,
  placeholder,
  required = false,
  disabled = false,
  rows = 3,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <textarea
        name={name}
        value={value}
        onChange={onChange}
        placeholder={placeholder}
        required={required}
        disabled={disabled}
        rows={rows}
        className={`
          w-full px-3 py-2 border rounded-lg transition-colors resize-vertical
          focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
          disabled:bg-gray-100 disabled:cursor-not-allowed
          ${
            error
              ? "border-error-red focus:ring-error-red"
              : "border-gray-300 hover:border-gray-400"
          }
        `}
        {...props}
      />

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Select component
export function FormSelect({
  label,
  name,
  value,
  onChange,
  error,
  options = [],
  placeholder = "Select an option",
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <select
        name={name}
        value={value}
        onChange={onChange}
        required={required}
        disabled={disabled}
        className={`
          w-full px-3 py-2 border rounded-lg transition-colors
          focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
          disabled:bg-gray-100 disabled:cursor-not-allowed
          ${
            error
              ? "border-error-red focus:ring-error-red"
              : "border-gray-300 hover:border-gray-400"
          }
        `}
        {...props}
      >
        <option value="">{placeholder}</option>
        {options.map((option) => (
          <option key={option.value} value={option.value}>
            {option.label}
          </option>
        ))}
      </select>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Checkbox component
export function FormCheckbox({
  label,
  name,
  checked,
  onChange,
  error,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      <label className="flex items-center cursor-pointer">
        <input
          type="checkbox"
          name={name}
          checked={checked}
          onChange={onChange}
          disabled={disabled}
          className={`
            mr-2 h-4 w-4 text-dark-red border-gray-300 rounded
            focus:ring-dark-red focus:ring-2
            disabled:cursor-not-allowed
          `}
          {...props}
        />
        <span className="text-sm text-gray-700">{label}</span>
      </label>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Radio group component
export function FormRadioGroup({
  label,
  name,
  value,
  onChange,
  error,
  options = [],
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-2">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <div className="space-y-2">
        {options.map((option) => (
          <label
            key={option.value}
            className="flex items-center cursor-pointer"
          >
            <input
              type="radio"
              name={name}
              value={option.value}
              checked={value === option.value}
              onChange={onChange}
              disabled={disabled}
              className={`
                mr-2 h-4 w-4 text-dark-red border-gray-300
                focus:ring-dark-red focus:ring-2
                disabled:cursor-not-allowed
              `}
              {...props}
            />
            <span className="text-sm text-gray-700">{option.label}</span>
          </label>
        ))}
      </div>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}
```

26. src/components/shared/LoadingSpinner.jsx

```
 import React from "react";

export default function LoadingSpinner({
  size = "md",
  color = "dark-red",
  text = "",
}) {
  const sizeClasses = {
    sm: "h-4 w-4",
    md: "h-8 w-8",
    lg: "h-12 w-12",
    xl: "h-16 w-16",
  };

  const colorClasses = {
    "dark-red": "border-dark-red",
    white: "border-white",
    gray: "border-gray-500",
  };

  return (
    <div className="flex flex-col items-center justify-center">
      <div
        className={`animate-spin rounded-full border-b-2 ${sizeClasses[size]} ${colorClasses[color]}
      ></div>
      {text && <p className="mt-2 text-sm text-gray-600">{text}</p>}
    </div>
  );
}
```

```
// Full screen loading component
export function FullScreenLoader({ text = "Loading..." }) {
  return (
    <div className="fixed inset-0 bg-white bg-opacity-90 flex items-center justify-center z-50">
      <LoadingSpinner size="lg" text={text} />
    </div>
  );
}

// Inline loading component
export function InlineLoader({ text = "Loading..." }) {
  return (
    <div className="flex items-center justify-center py-8">
      <LoadingSpinner size="md" text={text} />
    </div>
  );
}
```

27. src/utils/validation.js

```
 // Form validation utilities

// Email validation
export const validateEmail = (email) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!email) return "Email is required";
  if (!emailRegex.test(email)) return "Please enter a valid email address";
  return "";
};

// Password validation
export const validatePassword = (password) => {
  if (!password) return "Password is required";
  if (password.length < 8) return "Password must be at least 8 characters long";
  if (!/(?=.*[a-z])/.test(password))
    return "Password must contain at least one lowercase letter";
  if (!/(?=.*[A-Z])/.test(password))
    return "Password must contain at least one uppercase letter";
  if (!/(?=.*\d)/.test(password))
    return "Password must contain at least one number";
  if (!/(?=.*[!@#$%^&*])/.test(password))
    return "Password must contain at least one special character";
  return "";
};

// Phone validation
export const validatePhone = (phone) => {
  const phoneRegex = /^[\+]?[1-9][\d]{0,15}$/;
  if (!phone) return "Phone number is required";
  if (!phoneRegex.test(phone.replace(/[\s\-\(\)]/g, "")))
    return "Please enter a valid phone number";
  return "";
};

// Name validation
export const validateName = (name) => {
  if (!name) return "Name is required";
  if (name.length < 2) return "Name must be at least 2 characters long";
  if (!/^[a-zA-Z\s]+$/.test(name))
    return "Name can only contain letters and spaces";
  return "";
};

// PIN code validation
export const validatePinCode = (pinCode) => {
  const pinRegex = /^[1-9][0-9]{5}$/;
  if (!pinCode) return "PIN code is required";
  if (!pinRegex.test(pinCode)) return "Please enter a valid 6-digit PIN code";
  return "";
```

```
};

// Address validation
export const validateAddress = (address) => {
  const errors = {};

  if (!address.street || address.street.trim().length < 5) {
    errors.street = "Street address must be at least 5 characters long";
  }

  if (!address.city || address.city.trim().length < 2) {
    errors.city = "City is required";
  }

  if (!address.state || address.state.trim().length < 2) {
    errors.state = "State is required";
  }

  const pinError = validatePinCode(address.pinCode);
  if (pinError) {
    errors.pinCode = pinError;
  }

  return errors;
};

// Price validation
export const validatePrice = (price) => {
  if (!price && price !== 0) return "Price is required";
  if (isNaN(price) || price < 0) return "Price must be a valid positive number";
  if (price > 10000) return "Price cannot exceed ₹10,000";
  return "";
};

// Quantity validation
export const validateQuantity = (quantity) => {
  if (!quantity && quantity !== 0) return "Quantity is required";
  if (isNaN(quantity) || quantity < 1) return "Quantity must be at least 1";
  if (quantity > 50) return "Quantity cannot exceed 50";
  return "";
};

// Promo code validation
export const validatePromoCode = (code) => {
  if (!code) return "Promo code is required";
  if (code.length < 3) return "Promo code must be at least 3 characters long";
  if (!/^[A-Z0-9]+$/.test(code))
    return "Promo code can only contain uppercase letters and numbers";
  return "";
};

// Generic required field validation
export const validateRequired = (value, fieldName) => {
  if (!value || (typeof value === "string" && value.trim() === "")) {
    return `${fieldName} is required`;
  }
  return "";
};

// Form validation helper
export const validateForm = (formData, validationRules) => {
  const errors = {};

  Object.keys(validationRules).forEach((field) => {
    const rules = validationRules[field];
    const value = formData[field];

    for (const rule of rules) {
      const error = rule(value);
      if (error) {
        errors[field] = error;
        break; // Stop at first error for this field
      }
    }
  });

  return {
    errors,
    isValid: Object.keys(errors).length === 0,
  };
};

// Common validation rule sets
export const validationRules = {
  login: {
    email: [validateEmail],
    password: [validateRequired],
  },
  register: {
    name: [validateName],
    email: [validateEmail],
    password: [validatePassword],
    phone: [validatePhone],
  },
  menuItem: {
    name: [(value) => validateRequired(value, "Item name")],
    description: [(value) => validateRequired(value, "Description")],
    price: [validatePrice],
    category: [(value) => validateRequired(value, "Category")],
  },
  address: {
    street: [(value) => validateRequired(value, "Street address")],
    city: [(value) => validateRequired(value, "City")],
    state: [(value) => validateRequired(value, "State")],
    pinCode: [validatePinCode],
  },
};
```

A Comprehensive Project Report On

THE FOOD CITY

(Next-Generation Online Food Ordering System)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY *(Computer Science & Engineering)*

SUBMITTED BY: [Your Name] [Roll Number]

UNDER THE GUIDANCE OF: [Guide Name] [Designation]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING [COLLEGE NAME] [CITY, STATE - PIN CODE] [YEAR]

DECLARATION

I, **[Your Name]**, hereby declare that the project report entitled **"THE FOOD CITY"** submitted by me to **[College Name]**, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science & Engineering**, is a record of original work carried out by me under the supervision of **[Guide Name]**.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Date: _______________
Place: _______________

(Signature of the Student)
[Your Name]
[Roll Number]

CERTIFICATE

This is to certify that the project report entitled **"THE FOOD CITY"** being submitted by **[Your Name]** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** is a bona fide record of the work carried out by him/her under my guidance and supervision.

The work embodied in this report has not been submitted to any other University or Institute for the award of any degree or diploma.

Date: _______________
Place: _______________

(Signature of Guide)
[Guide Name]
[Designation]

(Signature of HOD)
[HOD Name]
Department of Computer Science

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **[Guide Name]**, for providing me an opportunity to do the project work in **"The Food City"** and giving us all support and guidance which made me complete the project duly. I am extremely validating to [him/her] for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs.

I owe my deep gratitude to our **Principal [Principal Name]**, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

I would not forget to remember **[HOD Name]**, Head of Computer Science Department, for his encouragement and more over for his timely support and guidance till the completion of our project work.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Department of Computer Science which helped us in successfully completing our project work. Also, I would like to extend our sincere esteems to all staff in laboratory for their timely support.

[Your Name]

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

In the 21st century, technology has permeated every aspect of our lives, transforming how we communicate, work, and most importantly, how we eat. **The Food City** is an ambitious project designed to bridge the gap between traditional culinary experiences and modern digital convenience. It is a state-of-the-art web application that allows users to browse menus from their favorite restaurants, customize their orders, and have delicious meals delivered right to their doorsteps—all with a few clicks.

The project is built on the robust foundation of **React.js**, creating a Single Page Application (SPA) that ensures a seamless, app-like experience without the lag of traditional page reloads. With a focus on speed (powered by **Vite**) and aesthetics (styled with **Tailwind CSS**), The Food City represents the future of local food commerce.

1.2 Motivation

The motivation behind developing **The Food City** stems from the observation of the chaotic and often inefficient nature of traditional food ordering.

Imagine a Friday night scenario: A family wants to order dinner. They have to find pamphlets of different restaurants, call them one by one to check if they are open, ask for the menu which might have changed, shout over the phone to be heard in a busy kitchen, and finally wait anxiously without knowing if the order is actually being prepared.

We wanted to change this. We wanted to create a platform where:

- **Convenience is King:** No more phone calls. Just click and order.
- **Transparency is Absolute:** Users can see exactly what they are ordering, the price, and the status.
- **Empowerment:** Giving small restaurants a platform to compete with big chains by providing them with a digital storefront.

1.3 Problem Statement

The traditional food ordering system suffers from several critical flaws:

1. **Communication Errors:** Mishearing orders over the phone leads to wrong deliveries and customer dissatisfaction.
2. **Lack of Visuals:** Customers cannot see what the food looks like before ordering.
3. **Time-Consuming:** The process of calling, waiting on hold, and confirming orders takes too long.
4. **No Order History:** Customers cannot easily re-order their favorite meals.
5. ** inefficient Management:** Restaurants struggle to manage orders during peak hours using pen and paper.

1.4 Purpose & Objective

The main objective of "The Food City" is to develop a robust, user-friendly, and efficient online food ordering system.

Specific objectives include:

- To provide an intuitive interface for customers to browse food categories (Indian, Chinese, etc.).
- To implement a secure and easy-to-use cart and checkout system.
- To provide a real-time tracking mechanism for orders.
- To offer a comprehensive dashboard for restaurant administrators to manage menus and orders.
- In short, to make "ordering food" a delightful experience rather than a chore.

1.5 Scope

The scope of this project is vast, covering both the **B2C (Business to Consumer)** and **B2B (Business to Business)** aspects of food delivery.

- **Geographical Scope:** Initially targeted at [Your City Name], with scalability to expand to other regions.
- **Functional Scope:** Covers the entire lifecycle of an order from menu browsing -> cart -> checkout -> preparation -> delivery.
- **Technical Scope:** Web-based access on Desktops, Laptops, Tablets, and Smartphones.

CHAPTER 2: LITERATURE SURVEY & MARKET ANALYSIS

2.1 The Evolution of Food Delivery

Food delivery has evolved from the "Dabbawalas" of Mumbai, who used a complex manual coding system, to the telephone era of the 90s, where pizza chains promised "30 minutes or free." Today, we are in the era of **Hyper-local Logistics**, where algorithms determine the fastest route for a delivery partner.

2.2 Existing Systems

Current market leaders include:

- **Swiggy:** Known for its fast delivery and wide network.
- **Zomato:** Famous for its restaurant discovery and honest user reviews.
- **UberEats:** Leverages its global logistics network.

2.3 Limitations of Existing Systems (Why a new system?)

While giants exists, they have issues:

- **High Commissions:** They charge restaurants 20-30% commission, squeezing the profits of small eateries.
- **Complex UI:** Sometimes, the apps are too cluttered with ads and features, confusing the elderly or non-tech-savvy users.
- **Lack of Personalization:** Local favorites often get lost in the algorithm.

The Food City aims to solve this by focusing on a **Hyper-local, Community-First** approach. It is designed to be lightweight, incredibly fast, and very simple to use, ensuring that even a 60-year-old grandmother can order her favorite Idli Sambar without help.

2.4 Who Will Use "The Food City"? (Target Audience)

The beauty of food is that everyone eats! However, our core demographics are:

1. **The Busy Professional:**
    - *Persona:* Rahul, 28, Software Engineer.
    - *Scenario:* Works late, no time to cook. Needs a quick, nutritious meal delivered to his office desk. He loves The Food City's "Repeat Order" feature.
2. **The College Student:**
    - *Persona:* Priya, 20, Medical Student.
    - *Scenario:* Late-night study sessions. Needs affordable, tasty snacks at 2 AM. She uses the "Deals" section to find the best discounts.

3. **The Homemaker:**
    - *Persona:* Sunita, 45, Housewife.
    - *Scenario:* Unexpected guests arrive. She uses The Food City to order a bulk party pack from the local famous restaurant, saving her hours in the kitchen.
4. **The Restaurant Owner:**
    - *Persona:* Mr. Sharma, 50, Owner of 'Sharma Sweets'.
    - *Scenario:* Expert at making sweets but bad at technology. The Food City's Admin Panel is so simple that he can update his menu prices in seconds without needing a computer degree.

2.5 A Story of Change

Let's imagine a small town, "Adityapur". People there love the local "Gupta Ji Ka Burger". But Gupta Ji only has one phone line. Customers get frustrated because the line is always busy. Gupta Ji loses business.

Enter **The Food City**. We onboard Gupta Ji. Now, all the college students in Adityapur just open the app, click "Gupta Ji's Burger", customize it (Expected delivery: 20 mins), and pay. Gupta Ji has a tablet in his shop. It beeps: *"New Order! 2 Aloo Tikki Burgers"*. He accepts it. The students see *"Order Accepted"*. 10 mins later: *"Preparing"*. 20 mins later: *"Delivered"*.

Sales double. Customers are happy. Gupta Ji is happy. **This is the impact of The Food City.**

CHAPTER 3: SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

To develop and run the application effectively, the following hardware setup was used:

- **Processor:** Intel Core i5 10th Gen (for faster compilation).
- **RAM:** 8GB DDR4 (to handle VS Code and Chrome simultaneously).
- **Storage:** 256GB SSD (Solid State Drives significantly reduce project load times).
- **Display:** 15.6-inch FHD Display (for accurate UI design).

3.2 Software Requirements

- **Operating System:** Windows 10 / 11 (64-bit).
- **Browser:** Google Chrome (Latest Version) - chosen for its excellent DevTools.
- **Code Editor:** Visual Studio Code (VS Code) - The industry standard for web development.
- **Version Control:** Git & GitHub.

3.3 Technology Stack Description

Frontend: React.js

React is a JavaScript library for building user interfaces. We chose React because:

- **Component-Based:** We built reusable components like `FoodCard`, `Navbar`, and `Button`. This reduced code duplication by 40%.
- **Virtual DOM:** Ensures the app is blazing fast, even with hundreds of menu items.

Build Tool: Vite

Vite is the next generation frontend tooling. Unlike Webpack, which takes a minute to start the server, Vite starts instantly using ES modules. This improved our development speed drastically.

Styling: Tailwind CSS

Tailwind is a utility-first CSS framework. Instead of writing separate `.css` files and worrying about class name conflicts (like `btn-primary`), we use utility classes directly in the JSX (e.g., `className="bg-red-500 text-white p-4 rounded"`). This made the UI design process 3x faster.

State Management: Context API

For managing global data like the **Shopping Cart** and **User Login Status**, we used React's Context API. This avoids "prop drilling" (passing data through 10 layers of components).

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

The project follows a **Client-Side Rendering (CSR)** architecture.

1. **The Browser (Client):** Downloads the empty HTML structure.
2. **JavaScript Bundle:** React takes over and dynamically fills the content.
3. **Data Layer:** Currently, we use `localStorage` to simulate a database. In a production version, this would be an API call to a Node.js server.

4.2 Application Flow

User Flow: `Landing Page` -> `Login/Signup` -> `Home Page (Menu)` -> `Product Details` -> `Add to Cart` -> `Review Cart` -> `Checkout (Address/Payment)` -> `Order Confirmation` -> `Order History`.

Admin Flow: `Admin Login` -> `Dashboard (Stats)` -> `Manage Menu (Add/Edit Items)` -> `Manage Orders (Update Status)` -> `Logout`.

4.3 Database Design (Schema Simulation)

Although we use LocalStorage, we structured data relationally:

Users Table:

Field	Type	Description
id	string	Unique User ID
name	string	Full Name
email	string	Email (Primary Key)
password	string	Encrypted/Hashed
role	string	'customer' or 'admin'

Orders Table:

Field	Type	Description
orderId	string	Unique Order ID
userId	string	Who placed the order
items	array	List of food items
total	number	Total Bill Amount
status	string	'Pending', 'Preparing', 'Delivered'
timestamp	date	Time of order

CHAPTER 5: DETAILED MODULE DESCRIPTION & USER STORIES

5.1 The Customer Module

The customer module is the heart of the application. It is designed with a "Mobile-First" approach, recognizing that 80% of food orders are placed on smartphones.

5.1.1 Landing & Home Page

The first thing a user sees is a vibrant, appetizing landing page.

- **Features:** Search bar, Categories (South Indian, Chinese, Desserts), and "Top Rated" items.
- **Design:** We used high-quality images of food to stimulate appetite (Psychological Design).

5.1.2 The "Smart" Cart

The cart isn't just a list; it's a calculator.

- It automatically calculates taxes (GST).
- It checks for coupon codes (e.g., "WELCOME50").
- It allows users to increase/decrease quantity instantly.

5.1.3 Checkout & Payment

We implemented a dummy payment gateway interface. Users can select "Credit Card", "UPI", or "Cash on Delivery". The form validation ensures no field is left empty.

5.2 The Admin Module

This is where the business happens.

5.2.1 The Dashboard

Shows a graph (imagined) or key metrics cards:

- **Total Revenue Today:** e.g., ₹15,400
- **Pending Orders:** 3
- **Active Delivery Boys:** 5

5.2.2 Menu Management

A restaurant changes its menu often. The manager can:

- Click "Add Item".
- Upload a photo.
- Set Price and Description.
- Mark an item as "Out of Stock" if ingredients run out.

5.3 User Stories: A Day in The Food City

Morning (8:00 AM): Mr. Roy, user ID #402, wakes up. He realizes he has no milk or bread. He opens *The Food City*, selects the "Breakfast" category, orders a Club Sandwich and Coffee. *Backstage:* The Admin sees the order, accepts it, and the kitchen starts brewing coffee.

Afternoon (1:30 PM): A corporate office has a team lunch. They order 15 Biryanis. The bulk order logic kicks in. The Admin marks the order as "Preparing" and sets a longer estimated time (45 mins). The team tracks this on their screen.

Evening (9:00 PM): A couple wants a romantic dinner at home. They order Pasta and Lava Cake. They use a promo code found in the "Deals" section. The payment is smooth. The food arrives hot.

This seamless flow is what **The Food City** promises.

CHAPTER 6: IMPLEMENTATION DETAILS

6.1 Folder Structure

We maintained a clean, industry-standard structure:

- `src/components` : Small building blocks using the Atomic Design Principle (Atoms -> Molecules -> Organisms).
- `src/pages` : The main screens.
- `src/context` : The "brain" of the app, holding the state.
- `src/hooks` : Custom logic hooks.

6.2 Key Algorithms

6.2.1 Search Algorithm

We implemented a real-time filter. As the user types "Chick...", the app instantly filters the 100+ item array to show "Chicken Tikka", "Chicken Wings", etc., using Javascript's `.filter()` and `.includes()` methods.

6.2.2 Cart Logic

The `addToCart` function is comprehensive:

```
const addToCart = (product) => {
  // Check if item already exists
  const exist = cartItems.find((x) => x.id === product.id);
  if (exist) {
    // If yes, just increase quantity
    setCartItems(cartItems.map((x) =>
      x.id === product.id ? { ...exist, qty: exist.qty + 1 } : x
    ));
  } else {
    // If no, add new item
    setCartItems([...cartItems, { ...product, qty: 1 }]);
  }
};
```

CHAPTER 7: TESTING AND RESULTS

7.1 Testing Strategy

We followed the **"Agile Testing"** methodology. Every component was tested as soon as it was built.

7.2 Test Cases

Test Case ID	Test Scenario	Expected Result	Actual Result	Status
TC_01	User enters valid email/pass	Login Successful	Login Successful	PASS
TC_02	User enters wrong password	Show "Invalid Credentials"	Shown	PASS
TC_03	Click "Add to Cart"	Cart badge count +1	Badge updated	PASS
TC_04	Checkout with empty address	Show Error Alert	Error Alert Shown	PASS
TC_05	Admin adds new Item	Item appears in Menu	Item appeared	PASS

7.3 Compatibility Testing

We tested the application on:

- **Google Chrome:** Perfect rendering.
- **Firefox:** Perfect rendering.
- **Mobile (iPhone Safari):** Responsive design adapted perfectly. The grid layout shifted to a single-column layout.

CHAPTER 8: CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The Food City is not just a college project; it is a proof-of-concept for a scalable business model. It successfully demonstrates how modern web technologies like React and Tailwind can create powerful, user-centric applications.

We encountered challenges, such as managing the complex state of the cart and making the design responsive for all devices, but we overcame them using Context API and Flexbox/Grid systems. The final result is a polished, functional, and aesthetically pleasing application that meets all the initial objectives.

8.2 Future Scope

Technology never stops evolving, and neither will The Food City.

1. **AI Integration:** Using Machine Learning to recommend food based on past orders (e.g., "You usually order Biryani on Fridays, want to order again?").
2. **GPS Tracking:** Integrating Google Maps API to show the live location of the delivery boy.

3. **Voice Ordering:** "Alexa, order me a Pizza from The Food City."
4. **Multi-Language Support:** Adding Hindi/Regional languages to make it accessible to everyone in India.

BIBLIOGRAPHY

The following resources were consulted during the development of this project:

(Websites)

- **React Official Documentation:** https://react.dev/ - For understanding Hooks and Components.
- **Tailwind CSS Docs:** https://tailwindcss.com/ - For utility class references.
- **Stack Overflow:** For debugging specific errors and logic issues.
- **YouTube Channels:** 'CodeWithHarry' and 'JavaScript Mastery' for tutorials.

(Books)

- *Learning React* by Alex Banks and Eve Porcello.
- *Clean Code* by Robert C. Martin (for writing maintainable code).

(End of Report)

PROJECT REPORT

ON

ONLINE FOOD ORDERING SYSTEM (THE FOOD CITY)

SUBMITTED BY: NAME: KUNAL **PROJECT:** THE FOOD CITY - ONLINE FOOD ORDERING SYSTEM **DATE:** FEBRUARY 2026

CERTIFICATE

This is to certify that the project report entitled **"THE FOOD CITY"** submitted by **KUNAL** in partial fulfillment of the requirements for the completion of the Software Development Course is a record of bona fide work carried out under my supervision and guidance.

The project demonstrates a comprehensive understanding of Full Stack Web Development principles, specifically React.js ecosystem, Modern UI/UX Design, and State Management.

Signature of Guide Signature of HOD

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher/guide who gave me the golden opportunity to do this wonderful project on the topic **"THE FOOD CITY" (Online Food Ordering System)**, which also helped me in doing a lot of Research and I came to know about so many new things. I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

I am also grateful to the open-source community for providing excellent tools and libraries like React, Tailwind CSS, and various npm packages that made this project possible.

EXECUTIVE SUMMARY

"The Food City" is a state-of-the-art, web-based online food ordering application designed to revolutionize the way customers interact with restaurants. In an era where convenience is paramount, this application bridges the gap between culinary desires and digital accessibility.

The project is built using a modern technology stack, primarily featuring **React.js** for the frontend, **Tailwind CSS** for styling, and **Context API** for sophisticated state management. The application follows a Single Page Application (SPA) architecture, ensuring seamless transitions, rapid load times, and an app-like user experience on the web.

Key Features Include: * **Dynamic User Interface:** A visually stunning, responsive design that adapts to all device sizes (Mobile, Tablet, Desktop). * **Advanced Menu Management:** Capabilities for searching, filtering, and sorting food items based on various criteria (Cuisine, Price, Rating, Veg/Non-Veg). * **Robust Cart & Checkout:** A persistent shopping cart using LocalStorage with a comprehensive checkout flow including address management and order summary. * **Dual-Role Architecture:** Distinct interfaces for "Customers" (Ordering flow) and "Admins" (Management dashboard). * **Admin Dashboard:** A powerful analytics hub providing real-time insights into sales, orders, and customer activity. * **Security:** Implemented protected routes and role-based access control (RBAC) to ensure data integrity and authorized access.

The development process adhered to the Agile methodology, allowing for iterative improvements and rapid feedback integration. Extensive testing, including unit, integration, and user acceptance testing, has been conducted to ensure a bug-free and robust final product.

This report details every aspect of the project, from initial requirements gathering and system design to implementation details, testing strategies, and user manuals. It serves as comprehensive documentation for developers, stakeholders, and end-users alike.

TABLE OF CONTENTS

---

1. INTRODUCTION

1.1 Project Background

The food and beverage industry has seen a massive paradigm shift in the last decade. The traditional model of dining out or calling a restaurant for delivery is rapidly being supplemented, and in some cases replaced, by online ordering platforms. The convenience of browsing menus, customizing orders, and tracking deliveries from a smartphone has become an expectation rather than a luxury.

"The Food City" was conceived to address the need for a customizable, lightweight, and efficient food ordering solution for independent restaurants or food chains that wish to avoid the high commissions of aggregator platforms. By building a proprietary system, businesses can maintain direct relationships with their customers and retain control over their brand and data.

1.2 Problem Statement

Many small to medium-sized restaurants face significant challenges in the digital age: 1. **High Dependency on Aggregators:** Platforms like Swiggy and Zomato charge high commissions (up to 30%), significantly eating into profit margins. 2. **Lack of Customer Data:** Aggregators own the customer data. Restaurants often don't know who their loyal customers are. 3. **Limited Branding:** On third-party apps, a restaurant is just one of many cards in a list. There is limited scope for brand storytelling. 4. **Inefficient Phone Ordering:** Taking orders over the phone is prone to errors, time-consuming, and frustrating for customers during peak hours.

1.3 Objectives

The primary objectives of "The Food City" project are: 1. **To develop a user-friendly online ordering interface** that allows customers to browse menus and place orders intuitively. 2. **To implement a comprehensive administrative dashboard** for restaurant owners to manage menus, track orders, and view analytics. 3. **To ensure data persistence and state management** without a heavy backend initially, proving the viability of the frontend logic using LocalStorage and modern React patterns. 4. **To create a responsive design** that works flawlessly across desktops, tablets, and mobile phones, acknowledging the mobile-first nature of food ordering. 5. **To simulate real-world scenarios** such as authentication, role-based access control, and order lifecycle management.

1.4 Scope

In Scope: * **Customer Module:** Registration, Login, Menu Browsing, Search/Filter, Cart Management, Checkout Process, Order History, User Profile. * **Admin Module:** Admin Login, Dashboard (Analytics), Menu Management (Add/Edit/Delete/Toggle Availability), Order Management (Status Updates), Customer View. * **System Features:** LocalStorage persistence, Toast notifications, Loading states, Error handling, Protected routes.

Out of Scope (Future Versions): * **Live Payment Gateway Integration:** (Currently simulated). * **Real-time GPS Tracking:** (Currently simulated with status updates). * **Multi-vendor Support:** (Currently designed for a single entity).

1.5 Methodology

The project was executed using the **Agile functionality** with a focus on **Component-Driven Development (CDD)**.

1. **Requirement Gathering:** Analyzed competitors and interviewed potential users to list features.
2. **Design Phase:** created wireframes and defined the component hierarchy.
3. **Development:**
    - **Phase 1:** Core setup, routing, and basic layout.
    - **Phase 2:** Authentication and Context setup.
    - **Phase 3:** Customer modules (Menu, Cart).
    - **Phase 4:** Admin modules (Dashboard, Management).

4. **Testing:** Continuous unit testing of components and integration testing of flows.
5. **Deployment:** (Ready for deployment on Vercel/Netlify).

---

2. SYSTEM ANALYSIS

2.1 Existing System

In the absence of this system, the business relies on: * **Manual Phone Orders:** Staff writes down orders manually. * *Disadvantage:* High error rate, no visual menu for customers, waiting times on call. * **Third-Party Aggregators:** Using Swiggy/Zomato. * *Disadvantage:* Loss of revenue due to commissions, loss of brand identity. * **Walk-in Orders:** Physical presence required. * *Disadvantage:* Limited reach, customer inconvenience.

2.2 Proposed System

"The Food City" automates the entire flow: * **Digital Menu:** Always up-to-date, visually appealing with images. * **Automated Calculations:** Tax, delivery fees, and totals are calculated instantly, removing human error. * **Order Tracking:** Customers see the status of their order in real-time (Pending -> Preparing -> Delivery -> Delivered). * **Data Insights:** Owners get reports on "Top Selling Items" and "Peak Hours", allowing for better inventory planning.

2.3 Feasibility Study

A feasibility study is carried out to select the best system that meets performance requirements.

2.3.1 Technical Feasibility

The project uses standard web technologies (React, JavaScript, HTML5, CSS3) which are mature, well-documented, and supported by all modern browsers. The device requirements are minimal (any device with a browser), making it highly technically feasible.

2.3.2 Economic Feasibility

- **Development Cost:** Low (Open source libraries, free development tools like VS Code).
- **Maintenance Cost:** Low (Frontend hosting on platforms like Vercel is free for starter tiers).
- **ROI:** High, due to commission savings from third-party apps and increased operational efficiency.

2.3.3 Operational Feasibility

The User Interface is designed to be intuitive. * **For Customers:** Similar to popular apps they already use; zero learning curve. * **For Staff:** The Admin panel uses clear icons, color-coded statuses, and simple forms. Minimal training is required.

2.3.4 Legal Feasibility

The system handles user data (Names, Addresses, Phones). It is designed to be GDPR/Privacy-compliant by not storing sensitive payment data (simulated) and allowing users to manage their profiles.

2.4 Requirement Analysis

2.4.1 Functional Requirements

1. **Authentication:** Users must be able to Register and Login. System must distinguish between 'Admin' and 'Customer'.
2. **Menu Browsing:** Users should view items with images, prices, and descriptions.
3. **Search/Filter:** Users must be able to search for "Chicken" or filter by "Veg".
4. **Cart:** Users must be able to add items, change quantities, and remove items. Structure must persist on refresh.
5. **Order Placement:** Users must be able to input delivery details and confirm the order.
6. **Order Management:** Admins must be able to change order status.

2.4.2 Non-Functional Requirements

1. **Performance:** Page loads should be under 2 seconds.
2. **Reliability:** The system should not crash under normal load.
3. **Usability:** Mobile-responsive design is mandatory.
4. **Security:** Passwords must be validated (min length, complexity). API routes (simulated) must be protected.

---

3. SYSTEM DESIGN

3.1 System Architecture

The application follows a **Component-Based Architecture** within a **Single Page Application (SPA)** framework.

- **Presentation Layer:** React Components (Pages, UI Elements). Handles user interaction and HTML rendering.

- **Logic Layer:** React Hooks (Custom Hooks, useEffect) & Context API. Handles business logic, state management, and side effects.
- **Data Layer:** LocalStorage Wrapper (Utils). Handles data persistence and retrieval, acting as a mock database.

3.2 Data Flow Diagrams (DFD)

Context Level (Level 0) DFD

- **User** -> [Requests/Orders] -> **System**
- **System** -> [Order Status/Menu] -> **User**
- **Admin** -> [Updates Menu/Status] -> **System**
- **System** -> [Reports/Orders] -> **Admin**

Level 1 DFD (Order Process)

1. **Customer** inputs login details -> **Auth Process** validates -> Token Granted.
2. **Customer** selects Items -> **Cart Process** updates State -> Storage updated.
3. **Customer** confirms Order -> **Order Process** creates Record -> **Order Database** (LocalStorage).
4. **Admin** views Order -> **Order Process** retrieves Record -> **Dashboard Display**.

3.3 Database Design (Mock Schema)

Although implemented in LocalStorage, the data is structured relationally.

Table 1: Users | Field | Type | Description | | :— | :— | :— | | `id` | String (UUID) | Primary Key | | `name` | String | Full Name | | `email` | String | Unique Identifier | | `password` | String | Hashed (Simulated) | | `role` | Enum | 'admin' or 'customer' | | `addresses` | Array | List of address objects |

Table 2: MenuItems | Field | Type | Description | | :— | :— | :— | | `id` | String | Primary Key | | `name` | String | Item Name | | `price` | Number | Cost per unit | | `category` | String | e.g., Indian, Chinese | | `isAvailable`| Boolean | Stock status | | `tags` | Array | Search keywords | | `isVeg` | Boolean | Dietary classifier |

Table 3: Orders | Field | Type | Description | | :— | :— | :— | | `id` | String | Primary Key | | `userId` | String | Foreign Key (Users) | | `items` | Array | List of {itemId, qty, price} | | `total` | Number | Financial total | | `status` | Enum | pending, preparing, delivery, delivered | | `timestamp` | Date | Order creation time |

3.4 Algorithm Design

Cart Calculation Algorithm

```
FUNCTION calculateTotal(cartItems):
    subtotal = 0
    FOR EACH item IN cartItems:
        subtotal = subtotal + (item.price * item.quantity)

    deliveryFee = IF subtotal > 500 THEN 0 ELSE 40
    tax = subtotal * 0.05

    grandTotal = subtotal + deliveryFee + tax - discount
    RETURN grandTotal
END FUNCTION
```

Search Relevance Algorithm

The search feature uses a weighted scoring system (implicitly implemented via filtering): 1. Match in **Name** (High Priority). 2. Match in **Category** (Medium Priority). 3. Match in **Tags** (Low Priority). 4. Match in **Description**.

---

4. IMPLEMENTATION DETAILS

4.1 Technology Stack

- **Frontend Framework:** React 18.x
  - *Why?* Virtual DOM for performance, huge ecosystem, component reusability.
- **Build Tool:** Vite
  - *Why?* Extremely fast HMR (Hot Module Replacement), optimized production builds.
- **Styling:** Tailwind CSS
  - *Why?* Utility-first approach allows for rapid UI development without context switching between CSS and JS files. Consistent design system.
- **Routing:** React Router DOM v6
  - *Why?* Standard solution for declarative routing in React SPAs.
- **Icons:** React Icons (Fa, Md, Bi)
  - *Why?* Lightweight SVG icons that are easily customizable via props.
- **State Management:** React Context API + useReducer
  - *Why?* sufficient for this scale, avoids boilerplates of Redux.

4.2 Module Description

Authentication Module (`src/context/AuthContext.jsx`)

This module handles the user identity. It exposes `login`, `register`, and `logout` functions. It uses `localStorage` to persist the session so users stay logged in on refresh. * **Key Challenge:** Synchronizing state across multiple tabs. * **Solution:** Using `window.addEventListener('storage')` to detect session changes.

Cart Module (`src/context/CartContext.jsx`)

This is the heart of the e-commerce functionality. It maintains the `items` array. * **Features:** `addItem` (checks if item exists, increments qty if yes), `removeItem`, `updateQuantity`, `clearCart`. * **Persistence:** Automatically saves to `localStorage` whenever the cart state changes via `useEffect`.

Admin Dashboard (`src/pages/admin/Dashboard.jsx`)

Visualizes data using charts and stats cards. * **Implementation:** Calculates derived state from the raw `orders` array (e.g., `orders.reduce` for total revenue, `orders.filter` for today's orders).

4.3 Key Code Snippets Explanation

Protected Route Logic: This component acts as a gatekeeper. If a user tries to access `/admin` without the `admin` role, they are redirected.

```
// Simplified logic
if (!isAuthenticated) return <Navigate to="/login" />;
if (requiredRole && user.role !== requiredRole) return <AccessDenied />;
return children;
```

---

5. TESTING

5.1 Testing Strategy

We employed a "Black Box Testing" strategy where functionality was tested without looking at the internal code structure during the QA phase.

5.2 Test Cases

5.2.1 Authentication & User Profile

Test ID	Test Scenario	Steps	Expected Result	Status
TC_AUTH_01	Valid Login	1. Open Auth Modal2. Enter valid email/pass3. Click Login	Modal closes, Header shows User Name	PASS
TC_AUTH_02	Invalid Login	1. Enter wrong password2. Click Login	Error message "Invalid credentials" appears	PASS
TC_AUTH_03	Registration	1. Switch to Sign Up2. Enter details3. Click Create Account	Account created, auto-logged in	PASS
TC_AUTH_04	Password Strength	1. Enter "123"2. Submit	Validation error "Must be at least 8 chars…"	PASS

5.2.2 Cart Functionality

Test ID	Test Scenario	Steps	Expected Result	Status
TC_CART_01	Add Item	1. Click "Add" on Food Card	Item appears in cart, counter in header updates	PASS
TC_CART_02	Incr Quantity	1. Click "+" in Cart	Qty increases, Subtotal recalculates correcty	PASS
TC_CART_03	Remove Item	1. Click "Remove" icon	Item disappears from list	PASS

Test ID	Test Scenario	Steps	Expected Result	Status
TC_CART_04	Empty Cart	1. Click "Clear Cart"	All items removed, "Cart Empty" message shown	PASS

5.2.3 Order Processing

Test ID	Test Scenario	Steps	Expected Result	Status
TC_ORD_01	Checkout Flow	1. Go to Cart2. Click Checkout3. Fill details4. Pay	Order placed, redirect to Success Page	PASS
TC_ORD_02	Admin Update	1. Login as Admin2. Go to Orders3. Change status	Status updates in Admin and Customer View	PASS

---

6. USER MANUAL

6.1 For Customers

Getting Started

1. **Launch the App:** Open the URL in your browser.
2. **Sign Up:** Click the "Sign Up" button in the top right. Fill in your Name, Email, Phone, and Password.
3. **Browse Menu:** Click "Menu" in the navigation bar. You can scroll through categories or use the search bar.

Placing an Order

1. **Select Items:** Click the "Add" button on any item you wish to eat.
2. **View Cart:** Click the Cart icon (top right) or "Cart" in the bottom nav (mobile).
3. **Checkout:** Review your items. Add any special instructions (e.g., "No onions"). Click "Proceed to Checkout".

4. **Payment:** Confirm your delivery address. Select Payment method (Cash or UPI). Click "Place Order".
5. **Track:** Go to "My Orders" in the profile menu to see the live status.

6.2 For Administrators

Accessing the Dashboard

1. **Login:** Use the dedicated Admin credentials (`admin@example.com` / `Admin123!`).
2. **Dashboard:** The landing page shows you today's snapshot: Revenue, Total Orders, and Pending Orders.

Managing the Menu

1. **Navigate:** Click "Menu Management" in the sidebar.
2. **Add Item:** Click the "+ Add New Item" button. Upload an image, set the price, and description.
3. **Toggle Availability:** If an ingredient is out of stock, click the "Toggle" switch on the item card to mark it as "Unavailable". The item will instantly disappear for customers.
4. **Delete:** Use the Trash icon to permanently remove an item.

Processing Orders

1. **Navigate:** Click "Orders" in the sidebar.
2. **View:** Filters are available for "Pending", "Preparing", etc.
3. **Update Status:**
    - When kitchen starts: Click "Mark as Preparing".
    - When food is ready: Click "Out for Delivery".
    - When driver returns: Click "Mark as Delivered".

---

7. REQUIREMENTS TRACEABILITY MATRIX (RTM)

The RTM maps the user requirements to the specific design modules and test cases to ensure full coverage.

Req ID	Requirement Description	Module Implemented	Design Component	Test Case ID
R-01	User shall create an account	Auth Module	`AuthModal.jsx`, `AuthContext.jsx`	TC_AUTH_03

Req ID	Requirement Description	Module Implemented	Design Component	Test Case ID
R-02	User shall login securely	Auth Module	`AuthModal.jsx`, `validation.js`	TC_AUTH_01
R-03	User shall view menu items	Customer Module	`MenuPage.jsx`, `FoodCard.jsx`	TC_MENU_01
R-04	User shall filter by Veg/NonVeg	Customer Module	`FilterPanel.jsx`	TC_MENU_02
R-05	System shall calculate totals	Cart Module	`CartContext.jsx`	TC_CART_02
R-06	Admin shall view sales stats	Admin Module	`Dashboard.jsx`	TC_ADM_01
R-07	Admin shall update order status	Admin Module	`OrderManagement.jsx`	TC_ORD_02

---

8. CONCLUSION & FUTURE SCOPE

Conclusion "The Food City" successfully achieves its primary goals of creating a responsive, functional, and user-friendly food ordering system. By leveraging the power of React and LocalStorage, we have demonstrated that complex business logic can be handled efficiently on the client side for small to medium-scale applications. The project highlights the importance of clean code architecture, state management, and user-centric design.

Future Scope To make this a commercial-grade product, the following enhancements are planned: 1. **Backend Integration:** Moving from LocalStorage to a MongoDB/Node.js backend for scalable data storage. 2. **Payment Gateway:** Integrating Razorpay or Stripe for real money transactions. 3. **PWA Support:** Converting the web app into a Progressive Web App (PWA) for installability on mobile phones. 4. **Driver App:** A separate interface for delivery partners to accept orders and navigate using GPS.

---

9. BIBLIOGRAPHY

1. **React Documentation:** https://react.dev/
2. **Tailwind CSS Documentation:** https://tailwindcss.com/docs

3. **MDN Web Docs:** For JavaScript and LocalStorage references.
4. **React Icons:** https://react-icons.github.io/react-icons/
5. **React Router:** https://reactrouter.com/

---

10. APPENDIX A: SOURCE CODE

1. src/main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

2. src/App.jsx

```
import React from "react";
import { AuthProvider, useAuth } from "./context/AuthContext.jsx";
import { CartProvider } from "./context/CartContext.jsx";
import { ToastProvider } from "./components/shared/Toast.jsx";
import AppRouter from "./router/AppRouter.jsx";
import ErrorBoundary from "./components/shared/ErrorBoundary.jsx";
import ScrollToTop from "./components/shared/ScrollToTop.jsx";
import { FullScreenLoader } from "./components/shared/LoadingSpinner.jsx";

// App content component that waits for auth initialization
function AppContent() {
  const { isLoading } = useAuth();

  if (isLoading) {
    return <FullScreenLoader text="Initializing application..." />;
  }

  return (
    <CartProvider>
      <AppRouter />
      <ScrollToTop />
    </CartProvider>
  );
}

function App() {
  return (
    <ErrorBoundary>
      <ToastProvider>
        <AuthProvider>
          <AppContent />
        </AuthProvider>
      </ToastProvider>
    </ErrorBoundary>
  );
}

export default App;
```

3. src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Custom utilities */
@layer utilities {
  .scrollbar-hide {
    -ms-overflow-style: none;
    scrollbar-width: none;
  }

  .scrollbar-hide::-webkit-scrollbar {
    display: none;
  }

  .line-clamp-1 {
    display: -webkit-box;
    -webkit-line-clamp: 1;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }

  .line-clamp-2 {
    display: -webkit-box;
    -webkit-line-clamp: 2;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }
  .line-clamp-3 {
    display: -webkit-box;
    -webkit-line-clamp: 3;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }
}
```

4. src/context/AuthContext.jsx

```
import React, { createContext, useContext, useReducer, useEffect } from "react";
import { mockAPI } from "../services/mockApi.js";
import { storageManager } from "../utils/localStorage.js";

// Auth context
const AuthContext = createContext();

// Auth actions
const AUTH_ACTIONS = {
  LOGIN_START: "LOGIN_START",
  LOGIN_SUCCESS: "LOGIN_SUCCESS",
  LOGIN_FAILURE: "LOGIN_FAILURE",
  REGISTER_START: "REGISTER_START",
  REGISTER_SUCCESS: "REGISTER_SUCCESS",
  REGISTER_FAILURE: "REGISTER_FAILURE",
  LOGOUT: "LOGOUT",
  CLEAR_ERROR: "CLEAR_ERROR",
  SET_USER: "SET_USER",
  INIT_COMPLETE: "INIT_COMPLETE",
};

// Initial state
const initialState = {
  user: null,
  isAuthenticated: false,
  isLoading: true, // Start with loading true
  error: null,
};

// Auth reducer
function authReducer(state, action) {
  switch (action.type) {
    case AUTH_ACTIONS.LOGIN_START:
    case AUTH_ACTIONS.REGISTER_START:
      return {
```

```
      ...state,
      isLoading: true,
      error: null,
    };

  case AUTH_ACTIONS.LOGIN_SUCCESS:
  case AUTH_ACTIONS.REGISTER_SUCCESS:
    return {
      ...state,
      user: action.payload,
      isAuthenticated: true,
      isLoading: false,
      error: null,
    };

  case AUTH_ACTIONS.LOGIN_FAILURE:
  case AUTH_ACTIONS.REGISTER_FAILURE:
    return {
      ...state,
      user: null,
      isAuthenticated: false,
      isLoading: false,
      error: action.payload,
    };

  case AUTH_ACTIONS.LOGOUT:
    return {
      ...state,
      user: null,
      isAuthenticated: false,
      isLoading: false,
      error: null,
    };

  case AUTH_ACTIONS.SET_USER:
    return {
      ...state,
      user: action.payload,
      isAuthenticated: !!action.payload,
      isLoading: false,
    };

  case AUTH_ACTIONS.INIT_COMPLETE:
    return {
      ...state,
      isLoading: false,
      };

    case AUTH_ACTIONS.CLEAR_ERROR:
      return {
        ...state,
        error: null,
      };

    default:
      return state;
  }
}

// Auth provider component
export function AuthProvider({ children }) {
  const [state, dispatch] = useReducer(authReducer, initialState);

  // Check for existing session on mount
  useEffect(() => {
    const initializeAuth = () => {
      try {
        const savedUser = storageManager.getUserSession();
        if (savedUser) {
          dispatch({ type: AUTH_ACTIONS.SET_USER, payload: savedUser });
        } else {
          dispatch({ type: AUTH_ACTIONS.INIT_COMPLETE });
        }
      } catch (error) {
        console.error("Error initializing auth:", error);
        dispatch({ type: AUTH_ACTIONS.INIT_COMPLETE });
      }
    };

    // Add a small delay to ensure localStorage is ready
    setTimeout(initializeAuth, 100);
  }, []);

  // Login function
  const login = async (credentials) => {
    dispatch({ type: AUTH_ACTIONS.LOGIN_START });

    try {
      const response = await mockAPI.login(credentials);

      if (response.success) {
        storageManager.setUserSession(response.user);
      dispatch({ type: AUTH_ACTIONS.LOGIN_SUCCESS, payload: response.user });
      return { success: true, message: response.message };
    } else {
      dispatch({
        type: AUTH_ACTIONS.LOGIN_FAILURE,
        payload: response.message,
      });
      return { success: false, message: response.message };
    }
  } catch (error) {
    const errorMessage = "Login failed. Please try again.";
    dispatch({ type: AUTH_ACTIONS.LOGIN_FAILURE, payload: errorMessage });
    return { success: false, message: errorMessage };
  }
};

// Register function
const register = async (userData) => {
  dispatch({ type: AUTH_ACTIONS.REGISTER_START });

  try {
    const response = await mockAPI.register(userData);

    if (response.success) {
      storageManager.setUserSession(response.user);
      dispatch({
        type: AUTH_ACTIONS.REGISTER_SUCCESS,
        payload: response.user,
      });
      return { success: true, message: response.message };
    } else {
      dispatch({
        type: AUTH_ACTIONS.REGISTER_FAILURE,
        payload: response.message,
      });
      return { success: false, message: response.message };
    }
  } catch (error) {
    const errorMessage = "Registration failed. Please try again.";
    dispatch({ type: AUTH_ACTIONS.REGISTER_FAILURE, payload: errorMessage });
    return { success: false, message: errorMessage };
  }
};

// Logout function
const logout = async () => {
  try {
    await mockAPI.logout();
    dispatch({ type: AUTH_ACTIONS.LOGOUT });
    return { success: true };
  } catch (error) {
    console.error("Logout error:", error?.message || String(error));
    // Force logout even if API call fails
    dispatch({ type: AUTH_ACTIONS.LOGOUT });
    return { success: true };
  }
};

// Clear error function
const clearError = () => {
  dispatch({ type: AUTH_ACTIONS.CLEAR_ERROR });
};

// Update user function
const updateUser = (userData) => {
  const updatedUser = { ...state.user, ...userData };
  storageManager.setUserSession(updatedUser);
  dispatch({ type: AUTH_ACTIONS.SET_USER, payload: updatedUser });
};

// Check if user is admin
const isAdmin = () => {
  return state.user?.role === "admin";
};

// Check if user is customer
const isCustomer = () => {
  return state.user?.role === "customer";
};

const value = {
  ...state,
  login,
  register,
  logout,
  clearError,
  updateUser,
  isAdmin,
  isCustomer,
};

return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
```

```
}

// Custom hook to use auth context
export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error("useAuth must be used within an AuthProvider");
  }
  return context;
}

// HOC for protected routes
export function withAuth(Component, requiredRole = null) {
  return function AuthenticatedComponent(props) {
    const { isAuthenticated, user, isLoading } = useAuth();

    if (isLoading) {
      return (
        <div className="min-h-screen flex items-center justify-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-dark-red"></
        </div>
      );
    }

    if (!isAuthenticated) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4">
            <h2 className="text-2xl font-bold text-dark-red text-center mb-4">
              Access Denied
            </h2>
            <p className="text-gray-600 text-center mb-6">
              Please log in to access this page.
            </p>
            <button
              onClick={() => (window.location.href = "/login")}
              className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-r
            >
              Go to Login
            </button>
          </div>
        </div>
      );
    }

    if (requiredRole && user?.role !== requiredRole) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4">
            <h2 className="text-2xl font-bold text-error-red text-center mb-4">
              Unauthorized
            </h2>
            <p className="text-gray-600 text-center mb-6">
              You don't have permission to access this page.
            </p>
            <button
              onClick={() => window.history.back()}
              className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-6
            >
              Go Back
            </button>
          </div>
        </div>
      );
    }

    return <Component {...props} />;
  };
}
```

5. src/context/CartContext.jsx

```
import React, { createContext, useContext, useReducer, useEffect } from "react";
import { storageManager } from "../utils/localStorage.js";
import { vibrationUtils } from "../utils/vibration.js";

// Cart context
const CartContext = createContext();

// Cart actions
const CART_ACTIONS = {
  ADD_ITEM: "ADD_ITEM",
  REMOVE_ITEM: "REMOVE_ITEM",
  UPDATE_QUANTITY: "UPDATE_QUANTITY",
  CLEAR_CART: "CLEAR_CART",
  APPLY_PROMO: "APPLY_PROMO",
  REMOVE_PROMO: "REMOVE_PROMO",
  SET_DELIVERY_ADDRESS: "SET_DELIVERY_ADDRESS",
  SET_PAYMENT_METHOD: "SET_PAYMENT_METHOD",
  LOAD_CART: "LOAD_CART",
};
```

```javascript
// Initial state
const initialState = {
  items: [],
  subtotal: 0,
  discount: 0,
  deliveryFee: 30,
  total: 0,
  promoCode: null,
  deliveryAddress: null,
  paymentMethod: "cod",
  itemCount: 0,
};

// Cart reducer
function cartReducer(state, action) {
  switch (action.type) {
    case CART_ACTIONS.ADD_ITEM: {
      const existingItemIndex = state.items.findIndex(
        (item) => item.id === action.payload.id
      );

      let newItems;
      if (existingItemIndex >= 0) {
        // Update existing item quantity
        newItems = state.items.map((item, index) =>
          index === existingItemIndex
            ? { ...item, quantity: item.quantity + 1 }
            : item
        );
      } else {
        // Add new item
        newItems = [...state.items, { ...action.payload, quantity: 1 }];
      }

      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.REMOVE_ITEM: {
      const newItems = state.items.filter((item) => item.id !== action.payload);
      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.UPDATE_QUANTITY: {
      const { itemId, quantity } = action.payload;

      if (quantity <= 0) {
        const newItems = state.items.filter((item) => item.id !== itemId);
        return calculateTotals({ ...state, items: newItems });
      }

      const newItems = state.items.map((item) =>
        item.id === itemId ? { ...item, quantity } : item
      );
      return calculateTotals({ ...state, items: newItems });
    }

    case CART_ACTIONS.CLEAR_CART:
      return {
        ...initialState,
        deliveryAddress: state.deliveryAddress,
        paymentMethod: state.paymentMethod,
      };

    case CART_ACTIONS.APPLY_PROMO:
      return calculateTotals({
        ...state,
        promoCode: action.payload.promoCode,
        discount: action.payload.discount,
      });

    case CART_ACTIONS.REMOVE_PROMO:
      return calculateTotals({
        ...state,
        promoCode: null,
        discount: 0,
      });

    case CART_ACTIONS.SET_DELIVERY_ADDRESS:
      return { ...state, deliveryAddress: action.payload };

    case CART_ACTIONS.SET_PAYMENT_METHOD:
      return { ...state, paymentMethod: action.payload };

    case CART_ACTIONS.LOAD_CART:
      return calculateTotals(action.payload);

    default:
      return state;
  }
}

// Helper function to calculate totals
```

```
function calculateTotals(state) {
  // Ensure items is an array
  const items = Array.isArray(state.items) ? state.items : [];

  const subtotal = items.reduce((sum, item) => {
    const price = Number(item.price) || 0;
    const quantity = Number(item.quantity) || 0;
    return sum + price * quantity;
  }, 0);

  const itemCount = items.reduce((sum, item) => {
    const quantity = Number(item.quantity) || 0;
    return sum + quantity;
  }, 0);

  const discount = Number(state.discount) || 0;
  const deliveryFee = Number(state.deliveryFee) || 0;
  const total = subtotal - discount + deliveryFee;

  return {
    ...state,
    items,
    subtotal: Number(subtotal.toFixed(2)),
    total: Number(Math.max(0, total).toFixed(2)),
    itemCount,
    discount,
    deliveryFee,
  };
}

// Cart provider component
export function CartProvider({ children }) {
  const [state, dispatch] = useReducer(cartReducer, initialState);

  // Load cart from localStorage on mount
  useEffect(() => {
    const savedCart = storageManager.getCartData();
    if (savedCart && savedCart.items && savedCart.items.length > 0) {
      dispatch({ type: CART_ACTIONS.LOAD_CART, payload: savedCart });
    }
  }, []);

  // Save cart to localStorage whenever state changes
  useEffect(() => {
    storageManager.setCartData(state);
  }, [state]);
```

```
// Add item to cart
const addItem = (item) => {
  dispatch({ type: CART_ACTIONS.ADD_ITEM, payload: item });

  // Trigger vibration on mobile for haptic feedback
  vibrationUtils.addToCart();
};

// Remove item from cart
const removeItem = (itemId) => {
  dispatch({ type: CART_ACTIONS.REMOVE_ITEM, payload: itemId });

  // Light vibration for remove action
  vibrationUtils.light();
};

// Update item quantity
const updateQuantity = (itemId, quantity) => {
  dispatch({
    type: CART_ACTIONS.UPDATE_QUANTITY,
    payload: { itemId, quantity },
  });
};

// Clear entire cart
const clearCart = () => {
  dispatch({ type: CART_ACTIONS.CLEAR_CART });
};

// Apply promo code
const applyPromo = (promoCode, discount) => {
  dispatch({
    type: CART_ACTIONS.APPLY_PROMO,
    payload: { promoCode, discount },
  });

  // Success vibration for promo code applied
  vibrationUtils.success();
};

// Remove promo code
const removePromo = () => {
  dispatch({ type: CART_ACTIONS.REMOVE_PROMO });
};
```

```
// Set delivery address
const setDeliveryAddress = (address) => {
  dispatch({ type: CART_ACTIONS.SET_DELIVERY_ADDRESS, payload: address });
};

// Set payment method
const setPaymentMethod = (method) => {
  dispatch({ type: CART_ACTIONS.SET_PAYMENT_METHOD, payload: method });
};

// Get item quantity in cart
const getItemQuantity = (itemId) => {
  const item = state.items.find((item) => item.id === itemId);
  return item ? item.quantity : 0;
};

// Check if item is in cart
const isItemInCart = (itemId) => {
  return state.items.some((item) => item.id === itemId);
};

// Get cart summary for order
const getOrderSummary = () => {
  return {
    items: state.items.map((item) => ({
      menuItemId: item.id,
      name: item.name,
      price: item.price,
      quantity: item.quantity,
      specialInstructions: item.specialInstructions || "",
    })),
    subtotal: state.subtotal,
    discount: state.discount,
    deliveryFee: state.deliveryFee,
    total: state.total,
    promoCode: state.promoCode?.code || "",
    deliveryAddress: state.deliveryAddress,
    paymentMethod: state.paymentMethod,
  };
};

const value = {
  ...state,
  addItem,
  removeItem,
  updateQuantity,
    clearCart,
    applyPromo,
    removePromo,
    setDeliveryAddress,
    setPaymentMethod,
    getItemQuantity,
    isItemInCart,
    getOrderSummary,
  };

  return <CartContext.Provider value={value}>{children}</CartContext.Provider>;
}

// Custom hook to use cart context
export function useCart() {
  const context = useContext(CartContext);
  if (!context) {
    throw new Error("useCart must be used within a CartProvider");
  }
  return context;
}
```

6. src/components/auth/AuthModal.jsx

```
import React, { useState, useEffect, useRef } from "react";
import { useAuth } from "../../context/AuthContext.jsx";

export default function AuthModal({ isOpen, onClose, initialMode = "login" }) {
  const [mode, setMode] = useState(initialMode);
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
    phone: "",
  });
  const [showPassword, setShowPassword] = useState(false);
  const clearErrorRef = useRef();

  const { login, register, isLoading, error, clearError } = useAuth();

  // Store clearError in ref to avoid dependency issues
  clearErrorRef.current = clearError;

  // Sync mode with initialMode prop when it changes
  useEffect(() => {
    setMode(initialMode);
}, [initialMode]);

// Clear form data and errors when modal opens
useEffect(() => {
  if (isOpen) {
    setFormData({
      name: "",
      email: "",
      password: "",
      phone: "",
    });
    clearErrorRef.current();
  }
}, [isOpen]); // Removed clearError from dependencies

const handleInputChange = (e) => {
  const { name, value } = e.target;
  setFormData((prev) => ({
    ...prev,
    [name]: value,
  }));

  // Clear error when user starts typing
  if (error) {
    clearErrorRef.current();
  }
};

const handleSubmit = async (e) => {
  e.preventDefault();

  let result;
  if (mode === "login") {
    result = await login({
      email: formData.email,
      password: formData.password,
    });
  } else {
    result = await register({
      name: formData.name,
      email: formData.email,
      password: formData.password,
      phone: formData.phone,
    });
  }
```

```
    if (result.success) {
      onClose();
      // Reset form
      setFormData({
        name: "",
        email: "",
        password: "",
        phone: "",
      });
    }
  };

  const switchMode = () => {
    setMode(mode === "login" ? "register" : "login");
    clearErrorRef.current();
    setFormData({
      name: "",
      email: "",
      password: "",
      phone: "",
    });
  };

  const fillDemoCredentials = (role) => {
    if (role === "customer") {
      setFormData({
        ...formData,
        email: "mohit.kumar@example.com",
        password: "Password123!",
      });
    } else if (role === "admin") {
      setFormData({
        ...formData,
        email: "admin@example.com",
        password: "Admin123!",
      });
    }
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-
      <div className="bg-white rounded-lg shadow-lg max-w-md w-full max-h-[90vh] overflow-y-
        <div className="p-6">
          {/* Header */}
```

```
<div className="flex justify-between items-center mb-6">
  <h2 className="text-2xl font-bold text-dark-red">
    {mode === "login" ? "Welcome Back" : "Create Account"}
  </h2>
  <button
    onClick={onClose}
    className="text-gray-400 hover:text-gray-600 text-2xl"
  >
    ×
  </button>
</div>

{/* Demo Credentials */}
<div className="mb-6 p-4 bg-light-gray rounded-lg">
  <p className="text-sm text-gray-600 mb-2">Demo Credentials:</p>
  <div className="flex gap-2">
    <button
      type="button"
      onClick={() => fillDemoCredentials("customer")}
      className="text-xs bg-info-blue text-white px-2 py-1 rounded hover:bg-blue-6
    >
      Customer
    </button>
    <button
      type="button"
      onClick={() => fillDemoCredentials("admin")}
      className="text-xs bg-spicy-orange text-white px-2 py-1 rounded hover:bg-ora
    >
      Admin
    </button>
  </div>
</div>

{/* Error Message */}
{error && (
  <div className="mb-4 p-3 bg-red-100 border border-red-400 text-red-700 rounded">
    {error}
  </div>
)}

{/* Form */}
<form onSubmit={handleSubmit} className="space-y-4">
  {mode === "register" && (
    <div>
      <label className="block text-sm font-medium text-gray-700 mb-1">
        Full Name
    </label>
    <input
      type="text"
      name="name"
      value={formData.name}
      onChange={handleInputChange}
      required
      className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outlin
      placeholder="Enter your full name"
    />
  </div>
)}

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">
    Email Address
  </label>
  <input
    type="email"
    name="email"
    value={formData.email}
    onChange={handleInputChange}
    required
    className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-
    placeholder="Enter your email"
  />
</div>

{mode === "register" && (
  <div>
    <label className="block text-sm font-medium text-gray-700 mb-1">
      Phone Number
    </label>
    <input
      type="tel"
      name="phone"
      value={formData.phone}
      onChange={handleInputChange}
      required
      className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outlin
      placeholder="Enter your phone number"
    />
  </div>
)}

<div>
```

```
    <label className="block text-sm font-medium text-gray-700 mb-1">
      Password
    </label>
    <div className="relative">
      <input
        type={showPassword ? "text" : "password"}
        name="password"
        value={formData.password}
        onChange={handleInputChange}
        required
        className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outlin
        placeholder="Enter your password"
      />
      <button
        type="button"
        onClick={() => setShowPassword(!showPassword)}
        className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-4
      >
        {showPassword ? " " : " "}
      </button>
    </div>
  </div>

  <button
    type="submit"
    disabled={isLoading}
    className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-r
  >
    {isLoading ? (
      <div className="flex items-center justify-center">
        <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-white
        {mode === "login" ? "Signing In..." : "Creating Account..."}
      </div>
    ) : mode === "login" ? (
      "Sign In"
    ) : (
      "Create Account"
    )}
  </button>
</form>

{/* Switch Mode */}
<div className="mt-6 text-center">
  <p className="text-gray-600">
    {mode === "login"
      ? "Don't have an account? "
```

```
                  : "Already have an account? "}
                <button
                  onClick={switchMode}
                  className="text-dark-red hover:text-hover-red font-medium"
                >
                  {mode === "login" ? "Sign Up" : "Sign In"}
                </button>
              </p>
            </div>
          </div>
        </div>
      </div>
    );
  }
```

7. src/utils/localStorage.js

```
// Local storage utilities for The Food City

const STORAGE_KEYS = {
  USER_SESSION: "foodcity_user_session",
  CART_DATA: "foodcity_cart",
  MENU_ITEMS: "foodcity_menu_items",
  ORDERS: "foodcity_orders",
  PROMO_CODES: "foodcity_promo_codes",
  DELIVERY_BOYS: "foodcity_delivery_boys",
  REVIEWS: "foodcity_reviews",
  USERS: "foodcity_users",
  ANALYTICS: "foodcity_analytics",
};

class LocalStorageManager {
  // Generic methods
  setItem(key, value) {
    try {
      localStorage.setItem(key, JSON.stringify(value));
      return true;
    } catch (error) {
      console.error(
        "Error saving to localStorage:",
        error?.message || String(error)
      );
      return false;
    }
  }
```

```
getItem(key) {
  try {
    const item = localStorage.getItem(key);
    return item ? JSON.parse(item) : null;
  } catch (error) {
    console.error(
      "Error reading from localStorage:",
      error?.message || String(error)
    );
    return null;
  }
}

removeItem(key) {
  try {
    localStorage.removeItem(key);
    return true;
  } catch (error) {
    console.error(
      "Error removing from localStorage:",
      error?.message || String(error)
    );
    return false;
  }
}

// User session management
setUserSession(user) {
  return this.setItem(STORAGE_KEYS.USER_SESSION, user);
}

getUserSession() {
  return this.getItem(STORAGE_KEYS.USER_SESSION);
}

clearUserSession() {
  return this.removeItem(STORAGE_KEYS.USER_SESSION);
}

// Cart management
setCartData(cartData) {
  return this.setItem(STORAGE_KEYS.CART_DATA, cartData);
}

getCartData() {
  return this.getItem(STORAGE_KEYS.CART_DATA) || { items: [], total: 0 };
```

```
}

clearCartData() {
  return this.removeItem(STORAGE_KEYS.CART_DATA);
}

// Menu items management
setMenuItems(menuItems) {
  return this.setItem(STORAGE_KEYS.MENU_ITEMS, menuItems);
}

getMenuItems() {
  return this.getItem(STORAGE_KEYS.MENU_ITEMS) || [];
}

// Orders management
setOrders(orders) {
  return this.setItem(STORAGE_KEYS.ORDERS, orders);
}

getOrders() {
  return this.getItem(STORAGE_KEYS.ORDERS) || [];
}

addOrder(order) {
  const orders = this.getOrders();
  orders.push(order);
  return this.setOrders(orders);
}

updateOrder(orderId, updates) {
  const orders = this.getOrders();
  const orderIndex = orders.findIndex((order) => order.id === orderId);
  if (orderIndex !== -1) {
    orders[orderIndex] = { ...orders[orderIndex], ...updates };
    return this.setOrders(orders);
  }
  return false;
}

// Promo codes management
setPromoCodes(promoCodes) {
  return this.setItem(STORAGE_KEYS.PROMO_CODES, promoCodes);
}

getPromoCodes() {
  return this.getItem(STORAGE_KEYS.PROMO_CODES) || [];
}

// Delivery boys management
setDeliveryBoys(deliveryBoys) {
  return this.setItem(STORAGE_KEYS.DELIVERY_BOYS, deliveryBoys);
}

getDeliveryBoys() {
  return this.getItem(STORAGE_KEYS.DELIVERY_BOYS) || [];
}

// Reviews management
setReviews(reviews) {
  return this.setItem(STORAGE_KEYS.REVIEWS, reviews);
}

getReviews() {
  return this.getItem(STORAGE_KEYS.REVIEWS) || [];
}

addReview(review) {
  const reviews = this.getReviews();
  reviews.push(review);
  return this.setReviews(reviews);
}

// Users management
setUsers(users) {
  return this.setItem(STORAGE_KEYS.USERS, users);
}

getUsers() {
  return this.getItem(STORAGE_KEYS.USERS) || [];
}

// Analytics management
setAnalytics(analytics) {
  return this.setItem(STORAGE_KEYS.ANALYTICS, analytics);
}

getAnalytics() {
  return this.getItem(STORAGE_KEYS.ANALYTICS) || {};
}

// Initialize with default data
```

```
initializeDefaultData(defaultData) {
  // Always update menu items to get latest additions
  this.setMenuItems(defaultData.menuItems);

  // Check if users contain old data (John Doe) or missing new data (Mohit Kumar) and for
  const currentUsers = this.getUsers();
  const hasOldData = currentUsers.some(
    (u) => u.name === "John Doe" || u.name === "Jane Smith"
  );
  const missingNewData = !currentUsers.some((u) => u.name === "Mohit Kumar");

  if (!currentUsers.length || hasOldData || missingNewData) {
    this.setUsers(defaultData.users);
    // Also reset orders if we reset users to avoid ID mismatches
    this.setOrders(defaultData.orders);
  }

  // Check for expired promo codes and force update
  const currentPromos = this.getPromoCodes();
  const hasExpiredPromos = currentPromos.some(
    (p) => new Date(p.expiryDate) < new Date()
  );

  if (!currentPromos.length || hasExpiredPromos) {
    this.setPromoCodes(defaultData.promoCodes);
  }
  if (!this.getDeliveryBoys().length) {
    this.setDeliveryBoys(defaultData.deliveryBoys);
  }

  // Check analytics for old data
  const analytics = this.getAnalytics();
  const hasOldAnalytics =
    analytics.topCustomers &&
    analytics.topCustomers.some((c) => c.name === "John Doe");

  if (!Object.keys(analytics).length || hasOldAnalytics) {
    this.setAnalytics(defaultData.analytics);
  }

  if (!this.getReviews().length) {
    this.setReviews(defaultData.reviews);
  }
}

// Clear all data (for testing/reset)
```

```
  clearAllData() {
    Object.values(STORAGE_KEYS).forEach((key) => {
      this.removeItem(key);
    });
  }
}

export const storageManager = new LocalStorageManager();
export { STORAGE_KEYS };
```

8. src/data/mockData.js

```
// Mock data for The Food City platform

export const mockUsers = [
  {
    id: "1",
    name: "Mohit Kumar",
    email: "mohit.kumar@example.com",
    password: "Password123!",
    phone: "+919876543210",
    role: "customer",
    addresses: [
      {
        id: "1",
        userId: "1",
        label: "Home",
        street: "Flat 402, Sunshine Apartments, Andheri West",
        city: "Mumbai",
        state: "Maharashtra",
        pinCode: "400001",
        landmark: "Near Central Mall",
        isDefault: true,
      },
    ],
    createdAt: new Date("2024-01-15"),
    isActive: true,
  },
  {
    id: "2",
    name: "Fatima Shaikh",
    email: "fatima.shaikh@example.com",
    password: "Password123!",
    phone: "+919876543211",
    role: "customer",
    addresses: [
      {
        id: "2",
        userId: "2",
        label: "Home",
        street: "Block C, Lajpat Nagar",
        city: "Delhi",
        state: "Delhi",
        pinCode: "110001",
        landmark: "Opposite Metro Station",
        isDefault: true,
      },
    ],
    createdAt: new Date("2024-02-10"),
    isActive: true,
  },
  {
    id: "admin1",
    name: "Admin User",
    email: "admin@example.com",
    password: "Admin123!",
    phone: "+919876543212",
    role: "admin",
    addresses: [],
    createdAt: new Date("2024-01-01"),
    isActive: true,
  },
];


export const mockMenuItems = [
  // Indian Category
  {
    id: "1",
    name: "Butter Chicken",
    description: "Creamy tomato-based curry with tender chicken pieces",
    price: 250,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1603894584373-5ac82b2ae398?w=400",
    rating: 4.5,
    reviewCount: 128,
    isAvailable: true,
    preparationTime: 25,
    tags: ["spicy", "creamy", "popular"],
  },
  {
    id: "2",
    name: "Paneer Tikka",
    description: "Grilled cottage cheese marinated in aromatic spices",
    price: 200,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=400",
    rating: 4.3,
    reviewCount: 95,
    isAvailable: true,
    preparationTime: 20,
    tags: ["grilled", "spicy", "vegetarian"],
  },
  {
    id: "3",
    name: "Dal Makhani",
    description: "Rich and creamy black lentils cooked overnight",
    price: 180,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1546833999-b9f581a1996d?w=400",
    rating: 4.4,
    reviewCount: 87,
    isAvailable: true,
    preparationTime: 15,
    tags: ["creamy", "comfort food", "vegetarian"],
  },
  {
    id: "13",
    name: "Chicken Biryani",
    description: "Aromatic basmati rice cooked with spiced chicken and saffron",
    price: 320,
    category: "Indian",
    type: "Non-Veg",
    image: "https://www.licious.in/blog/wp-content/uploads/2022/06/chicken-hyderabadi-biryar
    rating: 4.8,
    reviewCount: 245,
    isAvailable: true,
    preparationTime: 45,
    tags: ["aromatic", "rice", "festive"],
  },
  {
    id: "14",
    name: "Palak Paneer",
    description: "Cottage cheese cubes in creamy spinach gravy",
    price: 210,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1601050690597-df0568f70950?w=400",
    rating: 4.5,
    reviewCount: 134,
    isAvailable: true,
    preparationTime: 20,
    tags: ["creamy", "healthy", "vegetarian"],
  },
  {
    id: "15",
    name: "Mutton Rogan Josh",
    description: "Tender mutton cooked in aromatic Kashmiri spices",
    price: 380,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1585937421612-70a008356fbe?w=400",
    rating: 4.6,
    reviewCount: 98,
    isAvailable: true,
    preparationTime: 50,
    tags: ["aromatic", "tender", "kashmiri"],
  },
  {
    id: "16",
    name: "Chole Bhature",
    description: "Spicy chickpea curry served with fluffy fried bread",
    price: 160,
    category: "Indian",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1606491956689-2ea866880c84?w=400",
    rating: 4.3,
    reviewCount: 167,
    isAvailable: true,
    preparationTime: 25,
    tags: ["spicy", "traditional", "filling"],
  },
  {
    id: "17",
    name: "Fish Curry",
    description: "Fresh fish cooked in coconut-based curry",
    price: 290,
    category: "Indian",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1631452180519-c014fe946bc7?w=400",
    rating: 4.4,
    reviewCount: 76,
    isAvailable: true,
    preparationTime: 30,
    tags: ["coconut", "fresh", "coastal"],
  },

  // Chinese Category
  {
    id: "4",
    name: "Chicken Fried Rice",
    description: "Wok-tossed rice with chicken and vegetables",
    price: 180,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1512058564366-18510be2db19?w=400",
    rating: 4.2,
    reviewCount: 156,
    isAvailable: true,
    preparationTime: 18,
    tags: ["fried", "savory", "filling"],
  },
  {
    id: "5",
    name: "Veg Manchurian",
    description: "Deep-fried vegetable balls in tangy sauce",
    price: 150,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1626645738196-c2a7c87a8f58?w=400",
    rating: 4.1,
    reviewCount: 73,
    isAvailable: true,
    preparationTime: 22,
    tags: ["tangy", "crispy", "vegetarian"],
  },
  {
    id: "6",
    name: "Hakka Noodles",
    description: "Stir-fried noodles with vegetables and sauces",
    price: 160,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1585032226651-759b368d7246?w=400",
    rating: 4.0,
    reviewCount: 92,
    isAvailable: true,
    preparationTime: 16,
    tags: ["stir-fried", "savory", "noodles"],
  },
  {
    id: "18",
    name: "Sweet and Sour Chicken",
    description: "Crispy chicken in tangy sweet and sour sauce",
    price: 220,
    category: "Chinese",
    type: "Non-Veg",
    image: "https://images.unsplash.com/photo-1559847844-5315695dadae?w=400",
    rating: 4.3,
    reviewCount: 89,
    isAvailable: true,
    preparationTime: 20,
    tags: ["crispy", "tangy", "sweet"],
  },
  {
    id: "19",
    name: "Chilli Garlic Fried Rice",
    description: "Spicy fried rice with garlic and vegetables",
    price: 170,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1512058564366-18510be2db19?w=400",
    rating: 4.1,
    reviewCount: 112,
    isAvailable: true,
    preparationTime: 15,
    tags: ["spicy", "garlic", "rice"],
  },
  {
    id: "20",
    name: "Honey Chilli Potato",
    description: "Crispy potato fingers tossed in honey chilli sauce",
    price: 140,
    category: "Chinese",
    type: "Veg",
    image: "https://images.unsplash.com/photo-1598103442097-8b74394b95c6?w=400",
    rating: 4.2,
    reviewCount: 156,
    isAvailable: true,
    preparationTime: 18,
    tags: ["crispy", "honey", "appetizer"],
  },
{
  id: "21",
  name: "Chicken Chowmein",
  description: "Stir-fried noodles with chicken and mixed vegetables",
  price: 190,
  category: "Chinese",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1585032226651-759b368d7246?w=400",
  rating: 4.4,
  reviewCount: 145,
  isAvailable: true,
  preparationTime: 22,
  tags: ["stir-fried", "noodles", "protein"],
},
{
  id: "22",
  name: "Dragon Chicken",
  description: "Spicy Indo-Chinese chicken with bell peppers",
  price: 250,
  category: "Chinese",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1559847844-5315695dadae?w=400",
  rating: 4.5,
  reviewCount: 103,
  isAvailable: true,
  preparationTime: 25,
  tags: ["spicy", "indo-chinese", "bell peppers"],
},

// South Category
{
  id: "7",
  name: "Masala Dosa",
  description: "Crispy rice crepe filled with spiced potato curry",
  price: 120,
  category: "South",
  type: "Veg",
  image: "https://vismaifood.com/storage/app/uploads/public/45a/29b/a17/thumb__700_0_0_0_a
  rating: 4.6,
  reviewCount: 203,
  isAvailable: true,
  preparationTime: 20,
  tags: ["crispy", "traditional", "vegetarian"],
},
{
  id: "8",
  name: "Chicken Chettinad",
  description: "Spicy South Indian chicken curry with aromatic spices",
  price: 220,
  category: "South",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1565557623262-b51c2513a641?w=400",
  rating: 4.4,
  reviewCount: 67,
  isAvailable: true,
  preparationTime: 30,
  tags: ["spicy", "traditional", "aromatic"],
},
{
  id: "9",
  name: "Sambar Rice",
  description: "Steamed rice served with lentil-based vegetable stew",
  price: 140,
  category: "South",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1596797038530-2c107229654b?w=400",
  rating: 4.2,
  reviewCount: 45,
  isAvailable: true,
  preparationTime: 15,
  tags: ["healthy", "comfort food", "vegetarian"],
},
{
  id: "23",
  name: "Idli Sambar",
  description:
    "Steamed rice cakes served with lentil curry and coconut chutney",
  price: 100,
  category: "South",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1589301760014-d929f3979dbc?w=400",
  rating: 4.5,
  reviewCount: 189,
  isAvailable: true,
  preparationTime: 12,
  tags: ["healthy", "steamed", "traditional"],
},
{
  id: "24",
  name: "Medu Vada",
  description: "Crispy lentil donuts served with sambar and chutney",
  price: 80,
  category: "South",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1589301760014-d929f3979dbc?w=400",
  rating: 4.3,
  reviewCount: 234,
  isAvailable: true,
  preparationTime: 15,
  tags: ["crispy", "lentil", "snack"],
},
{
  id: "25",
  name: "Rava Upma",
  description: "Savory semolina porridge with vegetables and spices",
  price: 90,
  category: "South",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1606491956689-2ea866880c84?w=400",
  rating: 4.0,
  reviewCount: 67,
  isAvailable: true,
  preparationTime: 10,
  tags: ["savory", "healthy", "breakfast"],
},
{
  id: "26",
  name: "Chicken 65",
  description: "Spicy deep-fried chicken with curry leaves and chilies",
  price: 260,
  category: "South",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1565557623262-b51c2513a641?w=400",
  rating: 4.6,
  reviewCount: 178,
  isAvailable: true,
  preparationTime: 20,
  tags: ["spicy", "fried", "appetizer"],
},
{
  id: "27",
  name: "Coconut Rice",
  description: "Fragrant rice cooked with coconut and South Indian spices",
  price: 130,
  category: "South",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1596797038530-2c107229654b?w=400",
  rating: 4.2,
reviewCount: 92,
isAvailable: true,
preparationTime: 18,
tags: ["coconut", "fragrant", "rice"],
},

// Tandoor Category
{
id: "10",
name: "Chicken Tikka",
description: "Marinated chicken pieces grilled in tandoor oven",
price: 240,
category: "Tandoor",
type: "Non-Veg",
image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
rating: 4.7,
reviewCount: 189,
isAvailable: true,
preparationTime: 25,
tags: ["grilled", "smoky", "protein-rich"],
},
{
id: "11",
name: "Naan",
description: "Soft leavened bread baked in tandoor",
price: 40,
category: "Tandoor",
type: "Veg",
image: "https://images.unsplash.com/photo-1586190848861-99aa4a171e90?w=400",
rating: 4.3,
reviewCount: 234,
isAvailable: true,
preparationTime: 10,
tags: ["bread", "soft", "traditional"],
},
{
id: "12",
name: "Seekh Kebab",
description: "Spiced minced meat skewers grilled to perfection",
price: 280,
category: "Tandoor",
type: "Non-Veg",
image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
rating: 4.5,
reviewCount: 112,
isAvailable: true,
  preparationTime: 22,
  tags: ["grilled", "spiced", "kebab"],
},
{
  id: "28",
  name: "Tandoori Chicken",
  description:
    "Whole chicken marinated in yogurt and spices, roasted in tandoor",
  price: 420,
  category: "Tandoor",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
  rating: 4.8,
  reviewCount: 201,
  isAvailable: true,
  preparationTime: 35,
  tags: ["grilled", "yogurt", "traditional"],
},
{
  id: "29",
  name: "Paneer Tikka",
  description: "Marinated cottage cheese cubes grilled with vegetables",
  price: 220,
  category: "Tandoor",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=400",
  rating: 4.4,
  reviewCount: 145,
  isAvailable: true,
  preparationTime: 18,
  tags: ["grilled", "vegetarian", "cottage cheese"],
},
{
  id: "30",
  name: "Garlic Naan",
  description: "Soft bread topped with garlic and herbs, baked in tandoor",
  price: 60,
  category: "Tandoor",
  type: "Veg",
  image: "https://images.unsplash.com/photo-1586190848861-99aa4a171e90?w=400",
  rating: 4.6,
  reviewCount: 298,
  isAvailable: true,
  preparationTime: 12,
  tags: ["bread", "garlic", "herbs"],
},
{
  id: "31",
  name: "Tandoori Fish",
  description: "Fresh fish marinated in tandoori spices and grilled",
  price: 320,
  category: "Tandoor",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1631452180519-c014fe946bc7?w=400",
  rating: 4.5,
  reviewCount: 87,
  isAvailable: true,
  preparationTime: 25,
  tags: ["grilled", "fish", "spiced"],
},
{
  id: "32",
  name: "Reshmi Kebab",
  description: "Creamy chicken kebabs made with cashews and cream",
  price: 300,
  category: "Tandoor",
  type: "Non-Veg",
  image: "https://images.unsplash.com/photo-1599487488170-d11ec9c172f0?w=400",
  rating: 4.7,
  reviewCount: 156,
  isAvailable: true,
  preparationTime: 28,
  tags: ["creamy", "cashews", "kebab"],
},
];

export const mockPromoCodes = [
{
  id: "1",
  code: "FIRST50",
  description: "50% off on your first order",
  discountType: "percentage",
  discountValue: 50,
  minOrderValue: 200,
  maxDiscount: 150,
  expiryDate: new Date("2030-12-31"),
  usageLimit: 1000,
  usedCount: 245,
  isActive: true,
},
{
  id: "2",
  code: "WELCOME20",
  description: "20% off for new customers",
  discountType: "percentage",
  discountValue: 20,
  minOrderValue: 150,
  maxDiscount: 100,
  expiryDate: new Date("2030-11-30"),
  usageLimit: 500,
  usedCount: 123,
  isActive: true,
},
{
  id: "3",
  code: "FLAT100",
  description: "Flat  100 off on orders above  500",
  discountType: "fixed",
  discountValue: 100,
  minOrderValue: 500,
  maxDiscount: 100,
  expiryDate: new Date("2030-10-15"),
  usageLimit: 200,
  usedCount: 67,
  isActive: true,
},
];

export const mockDeliveryBoys = [
{
  id: "1",
  name: "Amit Sharma",
  phone: "+919876543210",
  isAvailable: true,
  currentOrders: [],
  rating: 4.8,
  totalDeliveries: 1247,
},
{
  id: "2",
  name: "Rajesh Kumar",
  phone: "+919876543211",
  isAvailable: true,
  currentOrders: [],
  rating: 4.6,
  totalDeliveries: 892,
},
{
  id: "3",
  name: "Suresh Patel",
  phone: "+919876543212",
  isAvailable: false,
  currentOrders: ["order_1"],
  rating: 4.7,
  totalDeliveries: 1056,
},
];

export const mockOrders = [
{
  id: "order_1",
  customerId: "1",
  items: [
    {
      menuItemId: "1",
      name: "Butter Chicken",
      price: 250,
      quantity: 1,
      specialInstructions: "Medium spicy",
    },
    {
      menuItemId: "11",
      name: "Naan",
      price: 40,
      quantity: 2,
      specialInstructions: "",
    },
  ],
  subtotal: 330,
  discount: 66,
  deliveryFee: 30,
  total: 294,
  status: "delivered",
  deliveryAddress: {
    id: "1",
    userId: "1",
    label: "Home",
    street: "123 Main St",
    city: "Mumbai",
    state: "Maharashtra",
    pinCode: "400001",
    landmark: "Near Central Mall",
    isDefault: true,
  },
  paymentMethod: "razorpay",
  promoCode: "WELCOME20",
  deliveryBoyId: "3",
  estimatedDeliveryTime: new Date("2024-07-18T13:30:00"),
  createdAt: new Date("2024-07-18T12:15:00"),
  updatedAt: new Date("2024-07-18T13:25:00"),
},
];

export const mockReviews = [
{
  id: "1",
  orderId: "order_1",
  customerId: "1",
  menuItemId: "1",
  rating: 5,
  comment:
    "Absolutely delicious! The butter chicken was perfectly creamy and flavorful.",
  createdAt: new Date("2024-07-18T14:00:00"),
  isModerated: false,
},
];

// Analytics mock data
export const mockAnalytics = {
totalViews: 15420,
totalRevenue: 125000,
totalOrders: 1247,
averageRating: 4.3,
todaysSales: 8500,
todaysOrders: 45,
cancelledOrders: 23,
topCustomers: [
  { name: "Mohit Kumar", orders: 15, spent: 3200 },
  { name: "Fatima Shaikh", orders: 12, spent: 2800 },
  { name: "Sadat Alam", orders: 10, spent: 2400 },
],
topItems: [
  { name: "Butter Chicken", orders: 89, revenue: 22250 },
  { name: "Masala Dosa", orders: 76, revenue: 9120 },
  { name: "Chicken Tikka", orders: 65, revenue: 15600 },
],
revenueByTime: [
  { time: "9 AM", revenue: 1200 },
  { time: "12 PM", revenue: 3500 },
  { time: "3 PM", revenue: 2100 },
  { time: "6 PM", revenue: 4200 },
  { time: "9 PM", revenue: 3800 },
],
};
```

9. src/pages/admin/Dashboard.jsx

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import { Card } from "../../components/shared/Layout.jsx";
import {
  FaShoppingBag,
  FaRupeeSign,
  FaCalendarAlt,
  FaChartLine,
  FaArrowUp,
  FaArrowDown,
  FaHistory,
  FaClock,
  FaUsers,
  FaUtensils,
  FaHourglassHalf,
  FaCheckCircle,
  FaStar,
  FaTrophy,
  FaEye,
  FaTruck,
} from "react-icons/fa";

export default function Dashboard() {
  const [selectedDate, setSelectedDate] = useState(
    new Date().toISOString().split("T")[0]
  );
  const [stats, setStats] = useState({
    todayOrders: 0,
    todayRevenue: 0,
    allTimeOrders: 0,
    allTimeRevenue: 0,
    selectedDayOrders: 0,
    selectedDayRevenue: 0,
    totalCustomers: 0,
    menuItems: 0,
    pendingOrders: 0,
    deliveredOrders: 0,
    averageRating: 0,
    totalDeliveryStaffs: 0,
  });
  const [recentOrders, setRecentOrders] = useState([]);
  const [topSellingItems, setTopSellingItems] = useState([]);
  const [loading, setLoading] = useState(true);

  const loadDashboardData = React.useCallback(() => {
    setLoading(true);

    // Mock data - replace with actual API calls
    setTimeout(() => {
      const today = new Date().toISOString().split("T")[0];
      const isToday = selectedDate === today;

      setStats({
        todayOrders: 43,
        todayRevenue: 12850,
        allTimeOrders: 2847,
        allTimeRevenue: 485600,
        selectedDayOrders: isToday ? 43 : Math.floor(Math.random() * 60) + 10,
        selectedDayRevenue: isToday
          ? 12850
          : Math.floor(Math.random() * 20000) + 5000,
        totalCustomers: 856,
        menuItems: 124,
        pendingOrders: 12,
        deliveredOrders: 2835,
        averageRating: 4.6,
        totalDeliveryStaffs: 18,
      });

      // Mock recent orders
      setRecentOrders([
        {
          id: "ORD001",
          customerName: "Vikram Malhotra",
          items: 3,
          total: 850,
          status: "preparing",
          time: "10 mins ago",
        },
        {
          id: "ORD002",
          customerName: "Anjali Gupta",
          items: 2,
          total: 650,
          status: "delivered",
          time: "25 mins ago",
        },
        {
          id: "ORD003",
          customerName: "Rohan Das",
          items: 5,
          total: 1200,
          status: "delivery",
          time: "32 mins ago",
        },
        {
          id: "ORD004",
          customerName: "Meera Iyer",
          items: 1,
          total: 320,
          status: "pending",
          time: "45 mins ago",
        },
        {
          id: "ORD005",
          customerName: "Arjun Singh",
          items: 4,
          total: 980,
          status: "delivered",
          time: "1 hour ago",
        },
      ]);

      // Mock top selling items
      setTopSellingItems([
        { name: "Butter Chicken", orders: 45, revenue: 11250 },
        { name: "Pizza Margherita", orders: 38, revenue: 9500 },
        { name: "Chicken Biryani", orders: 32, revenue: 8000 },
        { name: "Paneer Tikka", orders: 28, revenue: 5600 },
        { name: "Masala Dosa", orders: 25, revenue: 3750 },
      ]);

      setLoading(false);
    }, 500);
  }, [selectedDate]);

  useEffect(() => {
    loadDashboardData();
  }, [loadDashboardData]);

  const formatCurrency = (amount) => {
```

```
  return new Intl.NumberFormat("en-IN", {
    style: "currency",
    currency: "INR",
    maximumFractionDigits: 0,
  }).format(amount);
};

const formatDate = (dateString) => {
  const date = new Date(dateString);
  const today = new Date();
  const yesterday = new Date(today);
  yesterday.setDate(yesterday.getDate() - 1);

  if (dateString === today.toISOString().split("T")[0]) {
    return "Today";
  } else if (dateString === yesterday.toISOString().split("T")[0]) {
    return "Yesterday";
  } else {
    return date.toLocaleDateString("en-IN", {
      weekday: "short",
      day: "numeric",
      month: "short",
      year: "numeric",
    });
  }
};

const getOrderStatusColor = (status) => {
  switch (status) {
    case "pending":
      return "bg-gray-100 text-gray-800";
    case "preparing":
      return "bg-yellow-100 text-yellow-800";
    case "delivery":
      return "bg-blue-100 text-blue-800";
    case "delivered":
      return "bg-green-100 text-green-800";
    default:
      return "bg-gray-100 text-gray-800";
  }
};

const StatCard = ({ title, value, icon: IconComponent, color, subtext }) => (
  <Card className="p-6 hover:shadow-md transition-all duration-200 border border-gray-100'
    <div className="flex items-start justify-between">
      <div className="flex-1">
```

```
        <div className="flex items-center gap-3 mb-3">
          <div className={`p-2.5 rounded-lg ${color} flex-shrink-0`}>
            {IconComponent && (
              <IconComponent className="text-lg text-white" />
            )}
          </div>
          <p className="text-sm font-medium text-gray-600 leading-tight">
            {title}
          </p>
        </div>
        <div className="space-y-1">
          <p className="text-2xl font-bold text-gray-900 leading-none">
            {value}
          </p>
          {subtext && (
            <p className="text-xs text-gray-500 leading-relaxed">{subtext}</p>
          )}
        </div>
      </div>
    </div>
  </Card>
);

return (
  <div className="space-y-6">
    {/* Header */}
    <div>
      <h1 className="text-2xl sm:text-3xl font-bold text-gray-900">
        Dashboard
      </h1>
      <p className="text-sm sm:text-base text-gray-600 mt-1">
        Monitor your restaurant's orders and revenue
      </p>
    </div>

    {/* Date Selector */}
    <Card className="p-4 sm:p-6">
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4"
        <div className="flex items-center gap-3">
          <FaCalendarAlt className="text-red-600 text-xl" />
          <div>
            <h3 className="text-base sm:text-lg font-semibold text-gray-900">
              Select Date
            </h3>
            <p className="text-xs sm:text-sm text-gray-600">
              View orders and revenue for any specific day
          </p>
        </div>
      </div>
      <div className="flex flex-col sm:flex-row sm:items-center gap-2 sm:gap-3">
        <label
          htmlFor="date-picker"
          className="text-sm font-medium text-gray-700"
        >
          Choose Date:
        </label>
        <input
          id="date-picker"
          type="date"
          value={selectedDate}
          onChange={(e) => setSelectedDate(e.target.value)}
          max={new Date().toISOString().split("T")[0]}
          className="px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring
        />
        <span className="text-sm font-medium text-red-600">
          {formatDate(selectedDate)}
        </span>
      </div>
    </div>
  </Card>

  {/* Today's Stats */}
  <div>
    <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center
      <FaClock className="text-red-600" />
      Today's Performance
    </h2>
    <div className="grid grid-cols-2 gap-3 md:gap-6">
      <StatCard
        title="Today's Orders"
        value={loading ? "..." : stats.todayOrders}
        icon={FaShoppingBag}
        color="bg-blue-500"
        subtext={`Orders received today (${new Date().toLocaleDateString(
          "en-IN"
        )})`}
      />
      <StatCard
        title="Today's Revenue"
        value={loading ? "..." : formatCurrency(stats.todayRevenue)}
        icon={FaRupeeSign}
        color="bg-green-500"
      subtext="Total earnings today"
    />
  </div>
</div>

{/* Selected Day Stats */}
{selectedDate !== new Date().toISOString().split("T")[0] && (
  <div>
    <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-cent
      <FaCalendarAlt className="text-red-600" />
      {formatDate(selectedDate)} Performance
    </h2>
    <div className="grid grid-cols-2 gap-3 md:gap-6">
      <StatCard
        title={`${formatDate(selectedDate)} Orders`}
        value={loading ? "..." : stats.selectedDayOrders}
        icon={FaShoppingBag}
        color="bg-purple-500"
        subtext={`Orders on ${new Date(selectedDate).toLocaleDateString(
          "en-IN"
        )}`}
      />
      <StatCard
        title={`${formatDate(selectedDate)} Revenue`}
        value={loading ? "..." : formatCurrency(stats.selectedDayRevenue)}
        icon={FaRupeeSign}
        color="bg-orange-500"
        subtext="Total earnings for selected day"
      />
    </div>
  </div>
)}

{/* All Time Stats */}
<div>
  <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-center
    <FaHistory className="text-red-600" />
    All Time Performance
  </h2>
  <div className="grid grid-cols-2 gap-3 md:gap-6">
    <StatCard
      title="Total Orders"
      value={
        loading ? "..." : stats.allTimeOrders.toLocaleString("en-IN")
      }
      icon={FaChartLine}
      color="bg-red-600"
      subtext="All orders since restaurant started"
    />
    <StatCard
      title="Total Revenue"
      value={loading ? "..." : formatCurrency(stats.allTimeRevenue)}
      icon={FaRupeeSign}
      color="bg-emerald-600"
      subtext="Total earnings since inception"
    />
  </div>
</div>

{/* Additional Business Metrics */}
<div>
  <h2 className="text-lg sm:text-xl font-semibold text-gray-900 mb-4 flex items-cente
    <FaChartLine className="text-red-600" />
    Business Metrics
  </h2>
  <div className="grid grid-cols-2 md:grid-cols-3 gap-3 md:gap-6">
    <StatCard
      title="Total Customers"
      value={
        loading ? "..." : stats.totalCustomers.toLocaleString("en-IN")
      }
      icon={FaUsers}
      color="bg-blue-600"
      subtext="Registered customers"
    />
    <StatCard
      title="Total Delivery Staffs"
      value={loading ? "..." : stats.totalDeliveryStaffs}
      icon={FaTruck}
      color="bg-indigo-600"
      subtext="Active delivery personnel"
    />
    <StatCard
      title="Menu Items"
      value={loading ? "..." : stats.menuItems}
      icon={FaUtensils}
      color="bg-orange-500"
      subtext="Available dishes"
    />
    <StatCard
      title="Pending Orders"
      value={loading ? "..." : stats.pendingOrders}
```

```
          icon={FaHourglassHalf}
          color="bg-yellow-500"
          subtext="Awaiting preparation"
        />
        <StatCard
          title="Delivered Orders"
          value={
            loading ? "..." : stats.deliveredOrders.toLocaleString("en-IN")
          }
          icon={FaCheckCircle}
          color="bg-green-600"
          subtext="Successfully completed"
        />
        <StatCard
          title="Average Rating"
          value={loading ? "..." : `${stats.averageRating} `}
          icon={FaStar}
          color="bg-amber-500"
          subtext="Customer satisfaction"
          trend={5}
        />
      </div>
    </div>

    {/* Recent Orders and Top Selling Items */}
    <div className="grid grid-cols-1 xl:grid-cols-2 gap-4 sm:gap-6">
      {/* Recent Orders */}
      <Card className="p-4 sm:p-6">
        <div className="flex items-center justify-between mb-4">
          <h3 className="text-base sm:text-lg font-semibold text-gray-900 flex items-cente
            <FaShoppingBag className="text-red-600" />
            Recent Orders
          </h3>
          <Link
            to="/admin/orders"
            className="text-red-600 hover:text-red-700 text-xs sm:text-sm font-medium fle
          >
            <FaEye />
            <span className="hidden sm:inline">View All</span>
          </Link>
        </div>
        <div className="space-y-3">
          {recentOrders.slice(0, 5).map((order) => (
            <div
              key={order.id}
              className="flex items-center justify-between p-3 bg-gray-50 rounded-lg"
        >
          <div className="flex-1 min-w-0">
            <p className="font-medium text-gray-900 text-sm sm:text-base truncate">
              {order.customerName}
            </p>
            <p className="text-xs sm:text-sm text-gray-600">
              {order.items} items • {order.total}
            </p>
          </div>
          <div className="text-right flex-shrink-0 ml-2">
            <span
              className={`px-2 py-1 rounded-full text-xs font-medium ${getOrderStatus(
                order.status
              )}`}
            >
              {order.status}
            </span>
            <p className="text-xs text-gray-500 mt-1">{order.time}</p>
          </div>
        </div>
      ))}
    </div>
  </Card>

  {/* Top Selling Items */}
  <Card className="p-4 sm:p-6">
    <div className="flex items-center justify-between mb-4">
      <h3 className="text-base sm:text-lg font-semibold text-gray-900 flex items-cente
        <FaTrophy className="text-red-600" />
        Top Selling Items
      </h3>
      <Link
        to="/admin/menu"
        className="text-red-600 hover:text-red-700 text-xs sm:text-sm font-medium fle
      >
        <FaEye />
        <span className="hidden sm:inline">View Menu</span>
      </Link>
    </div>
    <div className="space-y-3">
      {topSellingItems.map((item, index) => (
        <div
          key={item.name}
          className="flex items-center justify-between p-3 bg-gray-50 rounded-lg"
        >
          <div className="flex items-center gap-3 flex-1 min-w-0">
```

```
                    <div className="w-7 h-7 sm:w-8 sm:h-8 bg-red-100 rounded-full flex items-c
                      <span className="text-xs sm:text-sm font-bold text-red-600">
                        #{index + 1}
                      </span>
                    </div>
                    <div className="min-w-0 flex-1">
                      <p className="font-medium text-gray-900 text-sm sm:text-base">
                        <span className="truncate">{item.name}</span>
                      </p>
                      <p className="text-xs sm:text-sm text-gray-600">
                        {item.orders} orders
                      </p>
                    </div>
                  </div>
                  <div className="text-right flex-shrink-0 ml-2">
                    <p className="font-medium text-gray-900 text-sm sm:text-base">
                      {item.revenue.toLocaleString("en-IN")}
                    </p>
                    <p className="text-xs text-gray-500">Revenue</p>
                  </div>
                </div>
              ))}
            </div>
          </Card>
        </div>
      </div>
  );
}
```

10. src/pages/customer/MenuPage.jsx

```
import React, { useState, useEffect, useMemo } from "react";
import { useSearchParams } from "react-router-dom";
import { Container, Grid } from "../../components/shared/Layout.jsx";
import { mockAPI } from "../../services/mockApi.js";
import SearchBar from "../../components/customer/SearchBar.jsx";
import FilterPanel from "../../components/customer/FilterPanel.jsx";
import SortOptions from "../../components/customer/SortOptions.jsx";
import FoodCard from "../../components/customer/FoodCard.jsx";
import { InlineLoader } from "../../components/shared/LoadingSpinner.jsx";

const CATEGORIES = ["All", "Indian", "Chinese", "South", "Tandoor"];

export default function MenuPage() {
  const [searchParams] = useSearchParams();
  const [menuItems, setMenuItems] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);

// Search and filter state
const [searchQuery, setSearchQuery] = useState("");
const [filters, setFilters] = useState({});
const [sortBy, setSortBy] = useState("relevance");
const [activeCategory, setActiveCategory] = useState("");

useEffect(() => {
  loadMenuItems();
}, []);

useEffect(() => {
  // Set category from URL parameter
  const categoryParam = searchParams.get("category");
  const searchParam = searchParams.get("search");

  if (categoryParam && CATEGORIES.includes(categoryParam)) {
    setActiveCategory(categoryParam);
  }

  if (searchParam) {
    setSearchQuery(searchParam);
  }
}, [searchParams]);

const loadMenuItems = async () => {
  try {
    setLoading(true);
    const response = await mockAPI.getMenuItems();
    if (response.success) {
      setMenuItems(response.data);
    } else {
      setError("Failed to load menu items");
    }
  } catch (err) {
    setError("Something went wrong while loading menu items");
    console.error("Error loading menu items:", err?.message || String(err));
  } finally {
    setLoading(false);
  }
};

// Generate search suggestions
const searchSuggestions = useMemo(() => {
  const suggestions = new Set();

  menuItems.forEach((item) => {
    // Add item names
    suggestions.add(item.name);
    // Add categories
    suggestions.add(item.category);
    // Add tags
    if (item.tags) {
      item.tags.forEach((tag) => suggestions.add(tag));
    }
  });

  return Array.from(suggestions);
}, [menuItems]);

// Filter and sort menu items
const filteredAndSortedItems = useMemo(() => {
  let filtered = [...menuItems];

  // Apply category filter from tabs
  if (activeCategory && activeCategory !== "All") {
    filtered = filtered.filter((item) => item.category === activeCategory);
  }

  // Apply search query
  if (searchQuery.trim()) {
    const query = searchQuery.toLowerCase();
    filtered = filtered.filter(
      (item) =>
        item.name.toLowerCase().includes(query) ||
        item.description.toLowerCase().includes(query) ||
        item.category.toLowerCase().includes(query) ||
        (item.tags &&
          item.tags.some((tag) => tag.toLowerCase().includes(query)))
    );
  }

  // Apply filters
  if (filters.category) {
    filtered = filtered.filter((item) => item.category === filters.category);
  }

  if (filters.type) {
    filtered = filtered.filter((item) => item.type === filters.type);
  }
  if (filters.minPrice !== undefined && filters.maxPrice !== undefined) {
    filtered = filtered.filter(
      (item) =>
        item.price >= filters.minPrice && item.price <= filters.maxPrice
    );
  }

  // Apply sorting
  switch (sortBy) {
    case "rating":
      filtered.sort((a, b) => b.rating - a.rating);
      break;
    case "price-low":
      filtered.sort((a, b) => a.price - b.price);
      break;
    case "price-high":
      filtered.sort((a, b) => b.price - a.price);
      break;
    case "name":
      filtered.sort((a, b) => a.name.localeCompare(b.name));
      break;
    case "delivery-time":
      filtered.sort((a, b) => a.preparationTime - b.preparationTime);
      break;
    default:
      // Relevance - keep original order or sort by rating
      filtered.sort((a, b) => b.rating - a.rating);
  }

  return filtered;
}, [menuItems, searchQuery, filters, sortBy, activeCategory]);

const handleSearch = (query) => {
  setSearchQuery(query);
};

const handleFiltersChange = (newFilters) => {
  setFilters(newFilters);
};

const handleClearFilters = () => {
  setFilters({});
};

const handleSortChange = (newSortBy) => {
  setSortBy(newSortBy);
};

const handleCategoryChange = (category) => {
  setActiveCategory(category);
  // Clear category filter when using tabs
  if (filters.category) {
    const newFilters = { ...filters };
    delete newFilters.category;
    setFilters(newFilters);
  }
};

if (loading) {
  return (
    <Container className="py-8">
      <InlineLoader text="Loading menu..." />
    </Container>
  );
}

if (error) {
  return (
    <Container className="py-8">
      <div className="text-center">
        <div className="bg-error-red text-white p-4 rounded-lg max-w-md mx-auto">
          <h3 className="font-semibold mb-2">Oops! Something went wrong</h3>
          <p className="text-sm">{error}</p>
          <button
            onClick={loadMenuItems}
            className="mt-3 bg-white text-error-red px-4 py-2 rounded font-medium hover:bg
          >
            Try Again
          </button>
        </div>
      </div>
    </Container>
  );
}

return (
  <Container className="py-4 md:py-8">
    {/* Header */}
    <div className="mb-6">
      <h1 className="text-3xl font-bold text-dark-red mb-2">Our Menu</h1>
      <p className="text-gray-600">
        Discover delicious dishes from our kitchen
      </p>
    </div>

    {/* Search Bar */}
    <div className="mb-6 flex justify-center">
      <SearchBar
        onSearch={handleSearch}
        suggestions={searchSuggestions}
        placeholder="Search for dishes, cuisines, or ingredients..."
      />
    </div>

    {/* Category Tabs */}
    <div className="mb-6">
      <div className="flex space-x-2 overflow-x-auto scrollbar-hide pb-2">
        {CATEGORIES.map((category) => (
          <button
            key={category}
            onClick={() =>
              handleCategoryChange(category === "All" ? "" : category)
            }
            className={`
              flex-shrink-0 px-4 py-2 rounded-full font-medium transition-colors
              ${
                activeCategory === category ||
                (category === "All" && !activeCategory)
                  ? "bg-dark-red text-white"
                  : "bg-gray-100 text-gray-700 hover:bg-gray-200"
              }
            `}
          >
            {category}
          </button>
        ))}
      </div>
    </div>

    {/* Controls */}
    <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4 mb-
      <div className="flex items-center space-x-4">
        <FilterPanel
          filters={filters}
          onFiltersChange={handleFiltersChange}
          onClear={handleClearFilters}
        />
```

```
      <SortOptions sortBy={sortBy} onSortChange={handleSortChange} />
    </div>

    <div className="text-sm text-gray-600">
      {filteredAndSortedItems.length} item
      {filteredAndSortedItems.length !== 1 ? "s" : ""} found
    </div>
  </div>

  {/* Results */}
  {filteredAndSortedItems.length === 0 ? (
    <div className="text-center py-12">
      <div className="w-24 h-24 bg-gray-100 rounded-full flex items-center justify-cente
        <svg
          className="w-12 h-12 text-gray-400"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z"
          />
        </svg>
      </div>
      <h3 className="text-xl font-semibold text-gray-800 mb-2">
        No items found
      </h3>
      <p className="text-gray-600 mb-4">
        Try adjusting your search or filters to find what you're looking
        for.
      </p>
      <button
        onClick={() => {
          setSearchQuery("");
          setFilters({});
          setActiveCategory("");
        }}
        className="bg-dark-red text-white px-6 py-2 rounded-lg hover:bg-hover-red transi
      >
        Clear All Filters
      </button>
    </div>
  ) : (
        <Grid cols={4} gap={6}>
          {filteredAndSortedItems.map((item) => (
            <FoodCard key={item.id} item={item} />
          ))}
        </Grid>
      )}
    </Container>
  );
}
```

11. src/pages/admin/OrderManagement.jsx

```
import React, { useState, useEffect } from "react";
import { Container, Card } from "../../components/shared/Layout.jsx";
import ConfirmDialog from "../../components/shared/ConfirmDialog.jsx";
import { useToast } from "../../components/shared/Toast.jsx";
import {
  FaSearch,
  FaFilter,
  FaClock,
  FaCheck,
  FaTruck,
  FaEye,
  FaPhoneAlt,
  FaMapMarkerAlt,
} from "react-icons/fa";

export default function OrderManagement() {
  const [orders, setOrders] = useState([]);
  const [filteredOrders, setFilteredOrders] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [statusFilter, setStatusFilter] = useState("all");
  const [loading, setLoading] = useState(true);

  // Professional confirmation dialog state
  const [confirmDialog, setConfirmDialog] = useState({
    isOpen: false,
    title: "",
    message: "",
    onConfirm: null,
    type: "warning",
  });

  // Toast notifications
  const { addToast } = useToast();
```

```
// Mock orders data
useEffect(() => {
  const mockOrders = [
    {
      id: "ORD001",
      customerName: "Rahul Sharma",
      customerPhone: "+91-9876543210",
      items: [
        { name: "Butter Chicken", quantity: 2, price: 250 },
        { name: "Naan", quantity: 3, price: 60 },
      ],
      total: 680,
      status: "pending",
      orderTime: "2025-01-22T14:30:00",
      address: "123 MG Road, Bangalore",
      paymentMethod: "Online",
    },
    {
      id: "ORD002",
      customerName: "Priya Patel",
      customerPhone: "+91-9876543211",
      items: [
        { name: "Pizza Margherita", quantity: 1, price: 299 },
        { name: "Garlic Bread", quantity: 2, price: 120 },
      ],
      total: 539,
      status: "preparing",
      orderTime: "2025-01-22T14:15:00",
      address: "456 Brigade Road, Bangalore",
      paymentMethod: "Cash on Delivery",
    },
    {
      id: "ORD003",
      customerName: "Amit Kumar",
      customerPhone: "+91-9876543212",
      items: [
        { name: "Chicken Biryani", quantity: 1, price: 320 },
        { name: "Raita", quantity: 1, price: 80 },
      ],
      total: 400,
      status: "delivery",
      orderTime: "2025-01-22T13:45:00",
      address: "789 Koramangala, Bangalore",
      paymentMethod: "UPI",
    },
    {
      id: "ORD004",
      customerName: "Sneha Reddy",
      customerPhone: "+91-9876543213",
      items: [
        { name: "Paneer Tikka", quantity: 1, price: 220 },
        { name: "Dal Tadka", quantity: 1, price: 180 },
        { name: "Rice", quantity: 2, price: 120 },
      ],
      total: 520,
      status: "delivered",
      orderTime: "2025-01-22T12:30:00",
      address: "321 Whitefield, Bangalore",
      paymentMethod: "Credit Card",
    },
  ];

  setTimeout(() => {
    setOrders(mockOrders);
    setFilteredOrders(mockOrders);
    setLoading(false);
  }, 500);
}, []);

// Filter orders based on search and status
useEffect(() => {
  let filtered = orders;

  if (searchTerm) {
    filtered = filtered.filter(
      (order) =>
        order.id.toLowerCase().includes(searchTerm.toLowerCase()) ||
        order.customerName.toLowerCase().includes(searchTerm.toLowerCase()) ||
        order.customerPhone.includes(searchTerm)
    );
  }

  if (statusFilter !== "all") {
    filtered = filtered.filter((order) => order.status === statusFilter);
  }

  setFilteredOrders(filtered);
}, [searchTerm, statusFilter, orders]);

const getStatusColor = (status) => {
  switch (status) {
    case "pending":
      return "bg-yellow-100 text-yellow-800";
    case "preparing":
      return "bg-orange-100 text-orange-800";
    case "delivery":
      return "bg-red-100 text-red-800";
    case "delivered":
      return "bg-green-100 text-green-800";
    default:
      return "bg-gray-100 text-gray-800";
  }
};

const getStatusIcon = (status) => {
  switch (status) {
    case "pending":
      return <FaClock className="mr-1" />;
    case "preparing":
      return <FaClock className="mr-1" />;
    case "delivery":
      return <FaTruck className="mr-1" />;
    case "delivered":
      return <FaCheck className="mr-1" />;
    default:
      return <FaClock className="mr-1" />;
  }
};

const updateOrderStatus = (orderId, newStatus) => {
  setOrders((prevOrders) =>
    prevOrders.map((order) =>
      order.id === orderId ? { ...order, status: newStatus } : order
    )
  );
};

const handleStatusUpdate = (orderId, newStatus, customerName) => {
  let dialogConfig = {};

  switch (newStatus) {
    case "preparing":
      dialogConfig = {
        title: "Mark Order as Preparing",
        message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Pre
        type: "info",
      };
      break;
    case "delivery":
      dialogConfig = {
        title: "Send Order for Delivery",
        message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Out
        type: "warning",
      };
      break;
    case "delivered":
      dialogConfig = {
        title: "Mark Order as Delivered",
        message: `Are you sure you want to mark order ${orderId} (${customerName}) as "Del
        type: "danger",
      };
      break;
    default:
      dialogConfig = {
        title: "Update Order Status",
        message: `Are you sure you want to update the status of order ${orderId}?`,
        type: "warning",
      };
  }

  setConfirmDialog({
    isOpen: true,
    ...dialogConfig,
    onConfirm: () => {
      updateOrderStatus(orderId, newStatus);

      // Show success toast
      const successMessages = {
        preparing: "Order marked as preparing successfully!",
        delivery: "Order sent for delivery successfully!",
        delivered: "Order marked as delivered successfully!",
      };

      addToast(
        successMessages[newStatus] || "Order status updated successfully!",
        "success",
        4000
      );
    },
  });
};

const formatTime = (timeString) => {
  return new Date(timeString).toLocaleString("en-IN", {
    hour: "2-digit",
    minute: "2-digit",
    day: "2-digit",
    month: "short",
  });
};

return (
  <Container className="py-6">
    <div className="space-y-6">
      {/* Header */}
      <div>
        <h1 className="text-2xl sm:text-3xl font-bold text-gray-900 mb-2">
          Order Management
        </h1>
        <p className="text-gray-600">Manage and track all customer orders</p>
      </div>

      {/* Search and Filters */}
      <Card className="p-4 sm:p-6">
        <div className="flex flex-col sm:flex-row gap-4">
          {/* Search */}
          <div className="flex-1 relative">
            <FaSearch className="absolute left-3 top-1/2 transform -translate-y-1/2 text-g
            <input
              type="text"
              placeholder="Search by Order ID, Customer Name, or Phone..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
              className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded-lg focus:ou
            />
          </div>

          {/* Status Filter */}
          <div className="relative">
            <FaFilter className="absolute left-3 top-1/2 transform -translate-y-1/2 text-g
            <select
              value={statusFilter}
              onChange={(e) => setStatusFilter(e.target.value)}
              className="pl-10 pr-8 py-2 border border-gray-300 rounded-lg focus:outline-n
            >
              <option value="all">All Orders</option>
              <option value="pending">Pending</option>
              <option value="preparing">Preparing</option>
              <option value="delivery">Out for Delivery</option>
              <option value="delivered">Delivered</option>
    </select>
  </div>
</div>
</Card>

{/* Orders List */}
<div className="space-y-4">
{loading ? (
  <Card className="p-6 text-center">
    <p className="text-gray-600">Loading orders...</p>
  </Card>
) : filteredOrders.length === 0 ? (
  <Card className="p-6 text-center">
    <p className="text-gray-600">No orders found</p>
  </Card>
) : (
  filteredOrders.map((order) => (
    <Card key={order.id} className="p-4 sm:p-6">
      <div className="space-y-4">
        {/* Order Header */}
        <div className="flex flex-col sm:flex-row sm:items-center justify-between
          <div className="flex items-center gap-3">
            <h3 className="text-lg font-semibold text-gray-900">
              {order.id}
            </h3>
            <span
              className={`px-3 py-1 rounded-full text-xs font-medium flex items-ce
                order.status
              )}`}
            >
              {getStatusIcon(order.status)}
              {order.status.charAt(0).toUpperCase() +
                order.status.slice(1)}
            </span>
          </div>
          <div className="text-sm text-gray-500">
            {formatTime(order.orderTime)}
          </div>
        </div>

        {/* Customer Info */}
        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
          <div>
            <h4 className="font-medium text-gray-900 mb-2">
              Customer Details
            </h4>
      <div className="space-y-1 text-sm">
        <p className="flex items-center gap-2">
          <FaEye className="text-gray-400" />
          {order.customerName}
        </p>
        <p className="flex items-center gap-2">
          <FaPhoneAlt className="text-gray-400" />
          {order.customerPhone}
        </p>
        <p className="flex items-start gap-2">
          <FaMapMarkerAlt className="text-gray-400 mt-0.5" />
          <span>{order.address}</span>
        </p>
      </div>
    </div>

    <div>
      <h4 className="font-medium text-gray-900 mb-2">
        Order Items
      </h4>
      <div className="space-y-1 text-sm">
        {order.items.map((item, index) => (
          <div
            key={index}
            className="flex justify-between items-center"
          >
            <span>
              {item.name} x{item.quantity}
            </span>
            <span> {item.price * item.quantity}</span>
          </div>
        ))}
        <div className="border-t pt-1 mt-2 font-medium">
          <div className="flex justify-between">
            <span>Total Amount</span>
            <span> {order.total}</span>
          </div>
          <div className="flex justify-between text-xs text-gray-600">
            <span>Payment</span>
            <span>{order.paymentMethod}</span>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
{/* Action Buttons */}
{order.status !== "delivered" && (
  <div className="flex flex-wrap gap-2 pt-3 border-t">
    {order.status !== "preparing" && (
      <button
        onClick={() =>
          handleStatusUpdate(
            order.id,
            "preparing",
            order.customerName
          )
        }
        className="px-4 py-2 bg-orange-600 text-white rounded-lg hover:bg-
      >
        <FaClock />
        Mark as Preparing
      </button>
    )}
    {order.status !== "delivery" && (
      <button
        onClick={() =>
          handleStatusUpdate(
            order.id,
            "delivery",
            order.customerName
          )
        }
        className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-rec
      >
        <FaTruck />
        Out for Delivery
      </button>
    )}
    <button
      onClick={() =>
        handleStatusUpdate(
          order.id,
          "delivered",
          order.customerName
        )
      }
      className="px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-gre
    >
      <FaCheck />
      Mark as Delivered
    </button>
```

```
                    </div>
                  )}
                </div>
              </Card>
            ))
          )}
        </div>
      </div>

      {/* Professional Confirmation Dialog */}
      <ConfirmDialog
        isOpen={confirmDialog.isOpen}
        onClose={() => setConfirmDialog((prev) => ({ ...prev, isOpen: false }))}
        onConfirm={confirmDialog.onConfirm}
        title={confirmDialog.title}
        message={confirmDialog.message}
        type={confirmDialog.type}
        confirmText="Yes, Update Status"
        cancelText="Cancel"
      />
    </Container>
  );
}
```

12. src/pages/admin/MenuManagement.jsx

```
import React, { useState, useEffect } from "react";
import { Container, Card } from "../../components/shared/Layout.jsx";
import ConfirmDialog from "../../components/shared/ConfirmDialog.jsx";
import { useToast } from "../../components/shared/Toast.jsx";
import {
  FaPlus,
  FaEdit,
  FaTrash,
  FaSearch,
  FaFilter,
  FaToggleOn,
  FaToggleOff,
  FaRupeeSign,
  FaClock,
  FaUtensils,
} from "react-icons/fa";

export default function MenuManagement() {
  const [menuItems, setMenuItems] = useState([]);
  const [categories, setCategories] = useState([]);
const [searchTerm, setSearchTerm] = useState("");
const [selectedCategory, setSelectedCategory] = useState("all");
const [loading, setLoading] = useState(true);
const [showAddForm, setShowAddForm] = useState(false);
const [editingItem, setEditingItem] = useState(null);
const [confirmDialog, setConfirmDialog] = useState({
  isOpen: false,
  title: "",
  message: "",
  onConfirm: null,
  type: "warning",
});

const { addToast } = useToast();

// Mock data for menu items
useEffect(() => {
  const mockCategories = [
    { id: 1, name: "Trending Now", items: 8 },
    { id: 2, name: "Indian Cuisine", items: 15 },
    { id: 3, name: "Chinese Cuisine", items: 10 },
    { id: 4, name: "South Cuisine", items: 6 },
    { id: 5, name: "Tandoor Cuisine", items: 4 },
  ];

  const mockMenuItems = [
    {
      id: 1,
      name: "Butter Chicken",
      category: "Indian Cuisine",
      price: 250,
      description: "Creamy tomato-based curry with tender chicken pieces",
      prepTime: 25,
      isAvailable: true,
      isVeg: false,
      tags: ["comfort food", "creamy", "popular"],
      image:
        "https://images.unsplash.com/photo-1588166524941-3bf61a9c41db?w=150&h=150&fit=crop
    },
    {
      id: 2,
      name: "Paneer Tikka",
      category: "Tandoor Cuisine",
      price: 180,
      description: "Grilled cottage cheese cubes with aromatic spices",
      prepTime: 15,
    isAvailable: true,
    isVeg: true,
    tags: ["vegetarian", "healthy", "grilled"],
    image:
      "https://images.unsplash.com/photo-1567188040759-fb8a883dc6d8?w=150&h=150&fit=cro
  },
  {
    id: 3,
    name: "Chicken Manchurian",
    category: "Chinese Cuisine",
    price: 220,
    description: "Spicy chicken balls in tangy sauce",
    prepTime: 20,
    isAvailable: true,
    isVeg: false,
    tags: ["spicy", "tangy", "chinese"],
    image:
      "https://images.unsplash.com/photo-1571934811356-5cc061b6821f?w=150&h=150&fit=cro
  },
  {
    id: 4,
    name: "Masala Dosa",
    category: "South Cuisine",
    price: 120,
    description: "Crispy rice crepe with spiced potato filling",
    prepTime: 15,
    isAvailable: false,
    isVeg: true,
    tags: ["healthy", "vegetarian", "south indian"],
    image:
      "https://vismaifood.com/storage/app/uploads/public/45a/29b/a17/thumb__700_0_0_0_a
  },
  {
    id: 5,
    name: "Chicken Biryani",
    category: "Trending Now",
    price: 320,
    description: "Aromatic basmati rice with spiced chicken",
    prepTime: 35,
    isAvailable: true,
    isVeg: false,
    tags: ["trending", "aromatic", "biryani"],
    image:
      "https://www.licious.in/blog/wp-content/uploads/2022/06/chicken-hyderabadi-biryan
  },
];
```

```
  setTimeout(() => {
    setCategories(mockCategories);
    setMenuItems(mockMenuItems);
    setLoading(false);
  }, 500);
}, []);

// Filter menu items
const filteredItems = menuItems.filter((item) => {
  const matchesSearch =
    item.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
    item.description.toLowerCase().includes(searchTerm.toLowerCase());
  const matchesCategory =
    selectedCategory === "all" || item.category === selectedCategory;
  return matchesSearch && matchesCategory;
});

// Toggle item availability
const toggleAvailability = (itemId) => {
  setMenuItems((prev) =>
    prev.map((item) =>
      item.id === itemId ? { ...item, isAvailable: !item.isAvailable } : item
    )
  );

  const item = menuItems.find((item) => item.id === itemId);
  const newStatus = !item.isAvailable;
  addToast(
    `${item.name} ${newStatus ? "enabled" : "disabled"} successfully!`,
    "success"
  );
};

// Delete item with confirmation
const handleDeleteItem = (item) => {
  setConfirmDialog({
    isOpen: true,
    title: "Delete Menu Item",
    message: `Are you sure you want to delete "${item.name}"?\n\nThis action cannot be und
    type: "danger",
    onConfirm: () => {
      setMenuItems((prev) =>
        prev.filter((menuItem) => menuItem.id !== item.id)
      );
      addToast(`${item.name} deleted successfully!`, "success");
    },
  });
};

// Handle save item (add or edit)
const handleSaveItem = (itemData) => {
  if (editingItem) {
    // Edit existing item
    setMenuItems((prev) =>
      prev.map((item) =>
        item.id === editingItem.id ? { ...item, ...itemData } : item
      )
    );
    addToast(`${itemData.name} updated successfully!`, "success");
  } else {
    // Add new item
    const newItem = {
      ...itemData,
      id: Math.max(...menuItems.map((item) => item.id), 0) + 1,
    };
    setMenuItems((prev) => [...prev, newItem]);
    addToast(`${itemData.name} added successfully!`, "success");
  }

  // Close form
  setShowAddForm(false);
  setEditingItem(null);
};

return (
  <Container className="py-6">
    <div className="space-y-6">
      {/* Header */}
      <div className="flex flex-col sm:flex-row justify-between items-start sm:items-cente
        <div>
          <h1 className="text-2xl sm:text-3xl font-bold text-gray-900">
            Menu Management
          </h1>
          <p className="text-gray-600 mt-1">
            Manage your restaurant menu items and categories
          </p>
        </div>
        <button
          onClick={() => setShowAddForm(true)}
          className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 flex iten
        >
          <FaPlus />
          Add New Item
        </button>
      </div>

      {/* Statistics Cards */}
      <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-blue-100 rounded-lg">
              <FaUtensils className="text-blue-600" />
            </div>
            <div>
              <p className="text-sm text-gray-600">Total Items</p>
              <p className="text-xl font-bold text-gray-900">
                {menuItems.length}
              </p>
            </div>
          </div>
        </Card>

        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-green-100 rounded-lg">
              <FaToggleOn className="text-green-600" />
            </div>
            <div>
              <p className="text-sm text-gray-600">Available</p>
              <p className="text-xl font-bold text-gray-900">
                {menuItems.filter((item) => item.isAvailable).length}
              </p>
            </div>
          </div>
        </Card>

        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-red-100 rounded-lg">
              <FaToggleOff className="text-red-600" />
            </div>
            <div>
              <p className="text-sm text-gray-600">Unavailable</p>
              <p className="text-xl font-bold text-gray-900">
                {menuItems.filter((item) => !item.isAvailable).length}
              </p>
            </div>
```

```
        </div>
      </Card>

      <Card className="p-4">
        <div className="flex items-center gap-3">
          <div className="p-2 bg-orange-100 rounded-lg">
            <FaFilter className="text-orange-600" />
          </div>
          <div>
            <p className="text-sm text-gray-600">Categories</p>
            <p className="text-xl font-bold text-gray-900">
              {categories.length}
            </p>
          </div>
        </div>
      </Card>
    </div>

    {/* Search and Filters */}
    <Card className="p-4 sm:p-6">
      <div className="flex flex-col sm:flex-row gap-4">
        {/* Search */}
        <div className="flex-1 relative">
          <FaSearch className="absolute left-3 top-1/2 transform -translate-y-1/2 text-g
          <input
            type="text"
            placeholder="Search menu items..."
            value={searchTerm}
            onChange={(e) => setSearchTerm(e.target.value)}
            className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded-lg focus:ou
          />
        </div>

        {/* Category Filter */}
        <div className="relative">
          <FaFilter className="absolute left-3 top-1/2 transform -translate-y-1/2 text-g
          <select
            value={selectedCategory}
            onChange={(e) => setSelectedCategory(e.target.value)}
            className="pl-10 pr-8 py-2 border border-gray-300 rounded-lg focus:outline-n
          >
            <option value="all">All Categories</option>
            {categories.map((category) => (
              <option key={category.id} value={category.name}>
                {category.name}
              </option>
            ))}
          </select>
        </div>
      </div>
    </Card>

    {/* Menu Items List */}
    <div className="space-y-4">
      {loading ? (
        <Card className="p-6 text-center">
          <p className="text-gray-600">Loading menu items...</p>
        </Card>
      ) : filteredItems.length === 0 ? (
        <Card className="p-6 text-center">
          <p className="text-gray-600">No menu items found</p>
        </Card>
      ) : (
        filteredItems.map((item) => (
          <Card key={item.id} className="p-4 sm:p-6">
            <div className="flex flex-col sm:flex-row gap-4">
              {/* Item Image */}
              <div className="w-full sm:w-24 h-48 sm:h-24 rounded-lg overflow-hidden bg-
                <img
                  src={item.image}
                  alt={item.name}
                  className="w-full h-full object-cover"
                  referrerPolicy="no-referrer"
                  onError={(e) => {
                    e.target.onerror = null;
                    e.target.src = "https://images.unsplash.com/photo-1546833999-b9f581a
                  }}
                />
              </div>

              {/* Item Details */}
              <div className="flex-1 space-y-2">
                <div className="flex flex-col sm:flex-row sm:items-start justify-betwee
                  <div>
                    <div className="flex items-center gap-2">
                      <h3 className="text-lg font-semibold text-gray-900">
                        {item.name}
                      </h3>
                      <span
                        className={`px-2 py-1 rounded-full text-xs font-medium ${item.i
                            ? "bg-green-100 text-green-800"
                            : "bg-red-100 text-red-800"
          }`}
        >
          {item.isVeg ? "Veg" : "Non-Veg"}
        </span>
        <span
          className={`px-2 py-1 rounded-full text-xs font-medium ${item.is
              ? "bg-green-100 text-green-800"
              : "bg-red-100 text-red-800"
            }`}
        >
          {item.isAvailable ? "Available" : "Unavailable"}
        </span>
      </div>
      <p className="text-sm text-gray-600 mt-1">
        {item.description}
      </p>
      <div className="flex items-center gap-4 mt-2 text-sm text-gray-500">
        <span className="flex items-center gap-1">
          <FaRupeeSign /> {item.price}
        </span>
        <span className="flex items-center gap-1">
          <FaClock />
          {item.prepTime} mins
        </span>
        <span className="px-2 py-1 bg-gray-100 rounded">
          {item.category}
        </span>
        {item.spiceLevel !== "None" && (
          <span className="px-2 py-1 bg-orange-100 text-orange-800 rounded
            {item.spiceLevel}
          </span>
        )}
      </div>
    </div>

    {/* Action Buttons */}
    <div className="flex items-center gap-2">
      <button
        onClick={() => toggleAvailability(item.id)}
        className={`p-2 rounded-lg ${item.isAvailable
            ? "bg-green-100 text-green-600 hover:bg-green-200"
            : "bg-red-100 text-red-600 hover:bg-red-200"
          }`}
        title={
          item.isAvailable ? "Disable Item" : "Enable Item"
        }
```

```
                    >
                      {item.isAvailable ? <FaToggleOn /> : <FaToggleOff />}
                    </button>
                    <button
                      onClick={() => setEditingItem(item)}
                      className="p-2 bg-blue-100 text-blue-600 rounded-lg hover:bg-blue-
                      title="Edit Item"
                    >
                      <FaEdit />
                    </button>
                    <button
                      onClick={() => handleDeleteItem(item)}
                      className="p-2 bg-red-100 text-red-600 rounded-lg hover:bg-red-20
                      title="Delete Item"
                    >
                      <FaTrash />
                    </button>
                  </div>
                </div>
              </div>
            </div>
          </Card>
        ))
      )}
    </div>
  </div>

  {/* Add/Edit Item Modal */}
  {(showAddForm || editingItem) && (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-cente
      <div className="bg-white rounded-lg max-w-2xl w-full max-h-[90vh] overflow-y-auto'
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-xl font-bold text-gray-900">
              {editingItem ? "Edit Menu Item" : "Add New Menu Item"}
            </h2>
            <button
              onClick={() => {
                setShowAddForm(false);
                setEditingItem(null);
              }}
              className="text-gray-400 hover:text-gray-600"
            >
              <svg
                className="w-6 h-6"
                fill="none"
```

```
                    stroke="currentColor"
                    viewBox="0 0 24 24"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M6 18L18 6M6 6l12 12"
                    />
                  </svg>
                </button>
              </div>

              <ItemForm
                item={editingItem}
                categories={categories}
                onSave={handleSaveItem}
                onCancel={() => {
                  setShowAddForm(false);
                  setEditingItem(null);
                }}
              />
            </div>
          </div>
        </div>
      )}

      {/* Confirmation Dialog */}
      <ConfirmDialog
        isOpen={confirmDialog.isOpen}
        onClose={() => setConfirmDialog((prev) => ({ ...prev, isOpen: false }))}
        onConfirm={confirmDialog.onConfirm}
        title={confirmDialog.title}
        message={confirmDialog.message}
        type={confirmDialog.type}
        confirmText="Yes, Delete"
        cancelText="Cancel"
      />
    </Container>
  );
}

// ItemForm component for adding/editing menu items
function ItemForm({ item, categories, onSave, onCancel }) {
  const [formData, setFormData] = useState({
    name: item?.name || "",
  category: item?.category || "Trending Now",
  price: item?.price || "",
  description: item?.description || "",
  prepTime: item?.prepTime || "",
  isVeg: item?.isVeg || false,
  isAvailable: item?.isAvailable !== undefined ? item.isAvailable : true,
  image: item?.image || "",
  tags: item?.tags || [],
});

const [errors, setErrors] = useState({});
const [imagePreview, setImagePreview] = useState(item?.image || "");
const [uploadingImage, setUploadingImage] = useState(false);
const [newTag, setNewTag] = useState("");

// Handle form field changes
const handleChange = (e) => {
  const { name, value, type, checked } = e.target;
  let newValue = type === "checkbox" ? checked : value;

  // Limit description to 60 characters
  if (name === "description" && newValue.length > 60) {
    newValue = newValue.substring(0, 60);
  }

  setFormData((prev) => ({
    ...prev,
    [name]: newValue,
  }));

  // Clear error when user starts typing
  if (errors[name]) {
    setErrors((prev) => ({ ...prev, [name]: "" }));
  }
};

// Handle tag operations
const addTag = () => {
  if (
    newTag.trim() &&
    formData.tags.length < 3 &&
    !formData.tags.includes(newTag.trim())
  ) {
    setFormData((prev) => ({
      ...prev,
      tags: [...prev.tags, newTag.trim()],
    }));
    setNewTag("");
  }
};

const removeTag = (tagToRemove) => {
  setFormData((prev) => ({
    ...prev,
    tags: prev.tags.filter((tag) => tag !== tagToRemove),
  }));
};

const handleTagInput = (e) => {
  const value = e.target.value;
  if (value.length <= 12) {
    setNewTag(value);
  }
};

// Handle image upload
const handleImageUpload = (e) => {
  const file = e.target.files[0];
  if (!file) return;

  // Validate file type
  if (!file.type.startsWith("image/")) {
    setErrors((prev) => ({
      ...prev,
      image: "Please select a valid image file",
    }));
    return;
  }

  // Validate file size (max 2MB)
  if (file.size > 2 * 1024 * 1024) {
    setErrors((prev) => ({
      ...prev,
      image: "Image size must be less than 2MB",
    }));
    return;
  }

  setUploadingImage(true);

  // Create preview URL
  const reader = new FileReader();
  reader.onload = (e) => {
    const imageUrl = e.target.result;
    setImagePreview(imageUrl);
    setFormData((prev) => ({ ...prev, image: imageUrl }));
    setUploadingImage(false);

    // Clear any previous errors
    if (errors.image) {
      setErrors((prev) => ({ ...prev, image: "" }));
    }
  };
  reader.readAsDataURL(file);
};

const validateForm = () => {
  const newErrors = {};

  if (!formData.name.trim()) newErrors.name = "Name is required";
  if (!formData.category.trim()) newErrors.category = "Category is required";
  if (!formData.price || formData.price <= 0)
    newErrors.price = "Valid price is required";
  if (!formData.description.trim())
    newErrors.description = "Description is required";
  if (!formData.prepTime || formData.prepTime <= 0)
    newErrors.prepTime = "Valid prep time is required";

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

const handleSubmit = (e) => {
  e.preventDefault();
  if (validateForm()) {
    onSave({
      ...formData,
      price: parseFloat(formData.price),
      prepTime: parseInt(formData.prepTime),
    });
  }
};

return (
  <form onSubmit={handleSubmit} className="space-y-6">
    <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
      {/* Name */}
      <div>
        <label className="block text-sm font-medium text-gray-700 mb-2">
          Item Name *
        </label>
        <input
          type="text"
          name="name"
          value={formData.name}
          onChange={handleChange}
          className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 f
            }`}
          placeholder="Enter item name"
        />
        {errors.name && (
          <p className="text-red-500 text-sm mt-1">{errors.name}</p>
        )}
      </div>

      {/* Category */}
      <div>
        <label className="block text-sm font-medium text-gray-700 mb-2">
          Category *
        </label>
        <select
          name="category"
          value={formData.category}
          onChange={handleChange}
          className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 f
            }`}
        >
          {categories.map((cat) => (
            <option key={cat.id} value={cat.name}>
              {cat.name}
            </option>
          ))}
        </select>
        {errors.category && (
          <p className="text-red-500 text-sm mt-1">{errors.category}</p>
        )}
      </div>

      {/* Price */}
      <div>
        <label className="block text-sm font-medium text-gray-700 mb-2">
          Price ( ) *
        </label>
        <input
        type="number"
        name="price"
        value={formData.price}
        onChange={handleChange}
        min="0"
        step="0.01"
        className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 f
          }`}
        placeholder="Enter price"
      />
      {errors.price && (
        <p className="text-red-500 text-sm mt-1">{errors.price}</p>
      )}
</div>

{/* Prep Time */}
<div>
  <label className="block text-sm font-medium text-gray-700 mb-2">
    Prep Time (minutes) *
  </label>
  <input
    type="number"
    name="prepTime"
    value={formData.prepTime}
    onChange={handleChange}
    min="1"
    className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 f
      }`}
    placeholder="Enter prep time"
  />
  {errors.prepTime && (
    <p className="text-red-500 text-sm mt-1">{errors.prepTime}</p>
  )}
</div>

{/* Tags Management */}
<div className="md:col-span-2">
  <label className="block text-sm font-medium text-gray-700 mb-2">
    Tags (Max 3 tags, 12 chars each)
  </label>
  <div className="space-y-3">
    {/* Add Tag Input */}
    <div className="flex gap-2">
      <input
        type="text"
        value={newTag}
        onChange={handleTagInput}
        onKeyPress={(e) =>
          e.key === "Enter" && (e.preventDefault(), addTag())
        }
        maxLength={12}
        className="flex-1 px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 f
        placeholder="Enter tag (max 12 chars)"
        disabled={formData.tags.length >= 3}
      />
      <button
        type="button"
        onClick={addTag}
        disabled={
          !newTag.trim() ||
          formData.tags.length >= 3 ||
          formData.tags.includes(newTag.trim())
        }
        className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 disab
      >
        Add
      </button>
    </div>

    {/* Character Counter */}
    <div className="text-sm text-gray-400">
      {newTag.length}/12 characters • {formData.tags.length}/3 tags
    </div>

    {/* Display Tags */}
    {formData.tags.length > 0 && (
      <div className="flex flex-wrap gap-2">
        {formData.tags.map((tag, index) => (
          <span
            key={index}
            className="inline-flex items-center gap-1 px-3 py-1 bg-blue-100 text-blu
          >
            {tag}
            <button
              type="button"
              onClick={() => removeTag(tag)}
              className="ml-1 text-blue-600 hover:text-blue-800"
            >
              ×
            </button>
          </span>
        ))}
```

```
      </div>
    )}
  </div>
</div>

{/* Image Upload */}
<div className="md:col-span-2">
  <label className="block text-sm font-medium text-gray-700 mb-2">
    Product Image
  </label>
  <div className="space-y-4">
    {/* Upload Button */}
    <div className="flex items-center gap-4">
      <label className="cursor-pointer bg-gray-100 hover:bg-gray-200 px-4 py-2 rounc
        <input
          type="file"
          accept="image/*"
          onChange={handleImageUpload}
          className="hidden"
        />
        <div className="flex items-center gap-2 text-gray-600">
          <svg
            className="w-5 h-5"
            fill="none"
            stroke="currentColor"
            viewBox="0 0 24 24"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M7 16a4 4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M15 13l-3
            />
          </svg>
          {uploadingImage ? "Uploading..." : "Choose Image"}
        </div>
      </label>
      <span className="text-sm text-gray-500">
        Max 2MB • JPG, PNG, GIF
      </span>
    </div>

    {/* Image Preview */}
    {imagePreview && (
      <div className="relative inline-block">
        <img
```

```
            src={imagePreview}
            alt="Preview"
            className="w-32 h-32 object-cover rounded-lg border"
          />
          <button
            type="button"
            onClick={() => {
              setImagePreview("");
              setFormData((prev) => ({ ...prev, image: "" }));
            }}
            className="absolute -top-2 -right-2 bg-red-500 text-white rounded-full w-
          >
            ×
          </button>
        </div>
      )}

      {errors.image && (
        <p className="text-red-500 text-sm">{errors.image}</p>
      )}
    </div>
  </div>
</div>

{/* Description */}
<div>
  <label className="block text-sm font-medium text-gray-700 mb-2">
    Description * (Max 60 characters)
  </label>
  <div className="relative">
    <textarea
      name="description"
      value={formData.description}
      onChange={handleChange}
      rows={3}
      maxLength={60}
      className={`w-full px-3 py-2 border rounded-lg focus:ring-2 focus:ring-red-500 f
        }`}
      placeholder="Enter item description (max 60 characters)"
    />
    <div className="absolute bottom-2 right-2 text-sm text-gray-400">
      {formData.description.length}/60
    </div>
  </div>
  {errors.description && (
    <p className="text-red-500 text-sm mt-1">{errors.description}</p>
```

```
          )}
        </div>

        {/* Checkboxes */}
        <div className="flex flex-wrap gap-6">
          <label className="flex items-center gap-2">
            <input
              type="checkbox"
              name="isVeg"
              checked={formData.isVeg}
              onChange={handleChange}
              className="rounded text-red-600 focus:ring-red-500"
            />
            <span className="text-sm font-medium text-gray-700">Vegetarian</span>
          </label>

          <label className="flex items-center gap-2">
            <input
              type="checkbox"
              name="isAvailable"
              checked={formData.isAvailable}
              onChange={handleChange}
              className="rounded text-red-600 focus:ring-red-500"
            />
            <span className="text-sm font-medium text-gray-700">Available</span>
          </label>
        </div>

        {/* Form Actions */}
        <div className="flex gap-4 pt-4 border-t">
          <button
            type="submit"
            className="flex-1 px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 font
          >
            {item ? "Update Item" : "Add Item"}
          </button>
          <button
            type="button"
            onClick={onCancel}
            className="flex-1 px-4 py-2 bg-gray-200 text-gray-800 rounded-lg hover:bg-gray-30
          >
            Cancel
          </button>
        </div>
      </form>
    );
```

```
}
```

13. src/components/auth/ProtectedRoute.jsx

```
import React from "react";
import { useAuth } from "../../context/AuthContext.jsx";

export default function ProtectedRoute({
  children,
  requiredRole = null,
  fallback = null,
}) {
  const { isAuthenticated, user, isLoading } = useAuth();

  // Show loading spinner while checking authentication
  if (isLoading) {
    return (
      <div className="min-h-screen flex items-center justify-center bg-light-gray">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-dark-red mx-
          <p className="text-gray-600">Loading...</p>
        </div>
      </div>
    );
  }

  // Check if user is authenticated
  if (!isAuthenticated) {
    return (
      fallback || (
        <div className="min-h-screen flex items-center justify-center bg-light-gray">
          <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4 text-ce
            <div className="mb-6">
              <div className="w-16 h-16 bg-error-red rounded-full flex items-center justify-
                <span className="text-white text-2xl"> </span>
              </div>
              <h2 className="text-2xl font-bold text-dark-red mb-2">
                Access Denied
              </h2>
              <p className="text-gray-600">
                Please log in to access this page.
              </p>
            </div>
            <button
              onClick={() => (window.location.href = "/")}
              className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover-
          >
            Go to Home
          </button>
        </div>
      </div>
    )
  );
}

// Check role-based access
if (requiredRole && user?.role !== requiredRole) {
  return (
    fallback || (
      <div className="min-h-screen flex items-center justify-center bg-light-gray">
        <div className="bg-white p-8 rounded-lg shadow-subtle max-w-md w-full mx-4 text-ce
          <div className="mb-6">
            <div className="w-16 h-16 bg-warn-yellow rounded-full flex items-center justif
              <span className="text-white text-2xl"> </span>
            </div>
            <h2 className="text-2xl font-bold text-error-red mb-2">
              Unauthorized Access
            </h2>
            <p className="text-gray-600">
              You don't have permission to access this page.
            </p>
          </div>
          <div className="space-y-2">
            <button
              onClick={() => window.history.back()}
              className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-
            >
              Go Back
            </button>
            <button
              onClick={() => (window.location.href = "/")}
              className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover
            >
              Go to Home
            </button>
          </div>
        </div>
      </div>
    )
  );
}
```

```
  // User is authenticated and authorized
  return children;
}
```

14. src/components/customer/FoodCard.jsx

```
import React from "react";
import { useCart } from "../../context/CartContext.jsx";
import { Button } from "../shared/Layout.jsx";

export default function FoodCard({ item }) {
  const { addItem, getItemQuantity, updateQuantity } = useCart();
  const quantity = getItemQuantity(item.id);

  const handleAddToCart = () => {
    addItem(item);
  };

  const handleIncrement = () => {
    updateQuantity(item.id, quantity + 1);
  };

  const handleDecrement = () => {
    if (quantity > 1) {
      updateQuantity(item.id, quantity - 1);
    } else {
      updateQuantity(item.id, 0);
    }
  };

  return (
    <div className="bg-white rounded-lg shadow-subtle overflow-hidden hover:shadow-lg transi
      {/* Image */}
      <div className="relative h-48 sm:h-52 md:h-48 lg:h-52 overflow-hidden flex-shrink-0">
        <img
          src={item.image}
          alt={item.name}
          className="w-full h-full object-cover hover:scale-105 transition-transform duratic
          referrerPolicy="no-referrer"
          onError={(e) => {
            e.target.onerror = null;
            e.target.src = "https://images.unsplash.com/photo-1546833999-b9f581a1996d?w=400'
          }}
        />

        {/* Type Badge */}
  <div className="absolute top-2 left-2">
    <span
      className={`
      px-2 py-1 rounded-full text-xs font-medium
      ${item.type === "Veg"
          ? "bg-success-green text-white"
          : "bg-error-red text-white"
        }
    `}
    >
      {item.type}
    </span>
  </div>

  {/* Rating Badge */}
  <div className="absolute top-2 right-2 bg-black bg-opacity-70 text-white px-2 py-1 r
    <span className="text-golden-yellow mr-1"> </span>
    {item.rating}
  </div>
</div>

{/* Content */}
<div className="p-4 flex-1 flex flex-col">
  <div className="mb-3 flex-1">
    <h3 className="text-lg font-semibold text-gray-800 mb-2 line-clamp-1">
      {item.name}
    </h3>
    <p className="text-gray-600 text-sm line-clamp-2 min-h-[2.5rem]">
      {item.description}
    </p>
  </div>

  {/* Tags */}
  {item.tags && item.tags.length > 0 && (
    <div className="flex flex-wrap gap-1 mb-3 min-h-[1.5rem]">
      {item.tags.slice(0, 3).map((tag, index) => (
        <span
          key={index}
          className="bg-light-gray text-gray-600 px-2 py-1 rounded-full text-xs"
        >
          {tag}
        </span>
      ))}
    </div>
  )}
```

```
{/* Price and Actions */}
<div className="flex items-center justify-between mt-auto">
  <div className="flex-1">
    <span className="text-xl font-bold text-dark-red">
      {item.price}
    </span>
    {item.preparationTime && (
      <p className="text-xs text-gray-500 mt-1">
        {item.preparationTime} mins
      </p>
    )}
  </div>

  {/* Add to Cart / Quantity Controls */}
  <div className="flex items-center ml-2">
    {quantity === 0 ? (
      <Button
        onClick={handleAddToCart}
        variant="primary"
        size="sm"
        className="px-3 py-1 text-sm min-w-[60px]"
      >
        Add
      </Button>
    ) : (
      <div className="flex items-center space-x-2">
        <button
          onClick={handleDecrement}
          className="w-7 h-7 rounded-full bg-gray-200 hover:bg-gray-300 flex items-c
        >
          -
        </button>
        <span className="font-medium text-dark-red min-w-[20px] text-center text-sm'
          {quantity}
        </span>
        <button
          onClick={handleIncrement}
          className="w-7 h-7 rounded-full bg-dark-red hover:bg-hover-red flex items-
        >
          +
        </button>
      </div>
    )}
  </div>
</div>
```

```
        {/* Reviews */}
        {item.reviewCount > 0 && (
          <div className="mt-3 pt-2 border-t border-gray-100">
            <p className="text-xs text-gray-500">
              {item.reviewCount} review{item.reviewCount !== 1 ? "s" : ""}
            </p>
          </div>
        )}
      </div>
    </div>
  );
}
```

15. src/components/customer/CartItem.jsx

```
import React, { useState } from "react";
import { useCart } from "../../context/CartContext.jsx";

export default function CartItem({ item }) {
  const { updateQuantity, removeItem } = useCart();
  const [specialInstructions, setSpecialInstructions] = useState(
    item.specialInstructions || ""
  );
  const [showInstructions, setShowInstructions] = useState(false);

  const handleQuantityChange = (newQuantity) => {
    if (newQuantity <= 0) {
      removeItem(item.id);
    } else {
      updateQuantity(item.id, newQuantity);
    }
  };

  const handleInstructionsChange = (instructions) => {
    setSpecialInstructions(instructions);
    // Update the item with special instructions
    // This would typically be handled by the cart context
  };

  return (
    <div className="bg-white rounded-lg shadow-sm border border-gray-200 p-3 md:p-4 mb-3 md:
      <div className="flex items-start space-x-3 md:space-x-4">
        {/* Item Image */}
        <div className="flex-shrink-0">
          <img
            src={item.image}
    alt={item.name}
    className="w-16 h-16 md:w-20 md:h-20 object-cover rounded-lg"
  />
</div>

{/* Item Details */}
<div className="flex-1 min-w-0">
  <div className="flex items-start justify-between">
    <div className="flex-1 min-w-0">
      <h3 className="text-base md:text-lg font-semibold text-gray-800 mb-1 truncate'
        {item.name}
      </h3>

      {/* Type Badge */}
      <div className="flex items-center space-x-1 md:space-x-2 mb-2">
        <span
          className={`
          inline-flex items-center px-1.5 md:px-2 py-1 rounded-full text-xs font-mec
          ${
            item.type === "Veg"
              ? "bg-success-green text-white"
              : "bg-error-red text-white"
          }
        `}
        >
          <span
            className={`
            w-1.5 h-1.5 md:w-2 md:h-2 rounded-full mr-1
            ${item.type === "Veg" ? "bg-white" : "bg-white"}
          `}
          ></span>
          {item.type}
        </span>

        {item.category && (
          <span className="text-xs text-gray-500 bg-gray-100 px-1.5 md:px-2 py-1 rou
            {item.category}
          </span>
        )}
      </div>

      {/* Price */}
      <div className="flex items-center space-x-2 mb-2 md:mb-3">
        <span className="text-lg md:text-xl font-bold text-dark-red">
           {item.price}
        </span>
      <span className="text-xs md:text-sm text-gray-500">each</span>
    </div>
  </div>

  {/* Remove Button */}
  <button
    onClick={() => removeItem(item.id)}
    className="text-gray-400 hover:text-error-red transition-colors p-1 flex-shrin
    title="Remove item"
  >
    <svg
      className="w-4 h-4 md:w-5 md:h-5"
      fill="none"
      stroke="currentColor"
      viewBox="0 0 24 24"
    >
      <path
        strokeLinecap="round"
        strokeLinejoin="round"
        strokeWidth={2}
        d="M19 7l-.867 12.142A2 2 0 0116.138 21H7.862a2 2 0 01-1.995-1.858L5 7m5 4
      />
    </svg>
  </button>
</div>

{/* Quantity Controls */}
<div className="flex flex-col sm:flex-row sm:items-center justify-between gap-3 sm
  <div className="flex items-center space-x-2 md:space-x-3">
    <span className="text-xs md:text-sm font-medium text-gray-700">
      Quantity:
    </span>
    <div className="flex items-center space-x-2">
      <button
        onClick={() => handleQuantityChange(item.quantity - 1)}
        className="w-7 h-7 md:w-8 md:h-8 rounded-full bg-gray-200 hover:bg-gray-30
        disabled={item.quantity <= 1}
      >
        -
      </button>
      <span className="font-medium text-dark-red min-w-[30px] text-center text-sm
        {item.quantity}
      </span>
      <button
        onClick={() => handleQuantityChange(item.quantity + 1)}
        className="w-7 h-7 md:w-8 md:h-8 rounded-full bg-dark-red hover:bg-hover-r
```

```
          >
            +
          </button>
        </div>
      </div>

      {/* Subtotal */}
      <div className="text-left sm:text-right">
        <div className="text-base md:text-lg font-bold text-dark-red">
          {(item.price * item.quantity).toFixed(2)}
        </div>
        <div className="text-xs text-gray-500">
          {item.quantity} × {item.price}
        </div>
      </div>
    </div>

    {/* Special Instructions */}
    <div className="mt-3 pt-3 border-t border-gray-100">
      <button
        onClick={() => setShowInstructions(!showInstructions)}
        className="text-xs md:text-sm text-dark-red hover:text-hover-red transition-co
      >
        <svg
          className="w-3 h-3 md:w-4 md:h-4 mr-1"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2h11a2 2 0 002-2v-5m-1.414-9.414a2 2
          />
        </svg>
        {showInstructions ? "Hide" : "Add"} special instructions
      </button>

      {showInstructions && (
        <div className="mt-2">
          <textarea
            value={specialInstructions}
            onChange={(e) => handleInstructionsChange(e.target.value)}
            placeholder="Any special requests for this item? (e.g., less spicy, extra
            className="w-full px-2 md:px-3 py-2 border border-gray-300 rounded-lg focu
```

```
                rows={2}
              />
            </div>
          )}

          {specialInstructions && !showInstructions && (
            <div className="mt-2 text-xs md:text-sm text-gray-600 bg-gray-50 p-2 rounded">
              <strong>Note:</strong> {specialInstructions}
            </div>
          )}
        </div>
      </div>
    </div>
  </div>
  );
}
```

16. src/components/customer/layout/Header.jsx

```
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "../../../context/AuthContext.jsx";
import { useCart } from "../../../context/CartContext.jsx";

export default function Header({ onOpenAuth }) {
  const [isProfileDropdownOpen, setIsProfileDropdownOpen] = useState(false);
  const { isAuthenticated, user, logout } = useAuth();
  const { itemCount } = useCart();
  const navigate = useNavigate();

  const handleLogout = async () => {
    await logout();
    setIsProfileDropdownOpen(false);
    navigate("/");
  };

  return (
    <header className="bg-white shadow-sm sticky top-0 z-40">
      <div className="container mx-auto px-4">
        <div className="flex items-center justify-between h-16">
          {/* Logo */}
          <Link to="/" className="flex items-center space-x-2">
            <img
              src="https://cornflowerblue-lion-884998.hostingersite.com/wp-content/uploads/
              alt="The Food City"
              className="h-10 w-10 object-contain"
  />
  <span className="text-xl font-bold text-dark-red hidden sm:block">
    The Food City
  </span>
</Link>

{/* Desktop Navigation */}
<nav className="hidden md:flex items-center space-x-6">
  <Link
    to="/"
    className="text-gray-700 hover:text-dark-red transition-colors"
  >
    Home
  </Link>
  <Link
    to="/menu"
    className="text-gray-700 hover:text-dark-red transition-colors"
  >
    Menu
  </Link>
  <Link
    to="/deals"
    className="text-gray-700 hover:text-dark-red transition-colors"
  >
    Deals
  </Link>
  <Link
    to="/about"
    className="text-gray-700 hover:text-dark-red transition-colors"
  >
    About
  </Link>
  <Link
    to="/contact"
    className="text-gray-700 hover:text-dark-red transition-colors"
  >
    Contact
  </Link>
</nav>

{/* Right side actions */}
<div className="flex items-center space-x-4">
  {/* Cart Icon - Hidden on mobile since it's in bottom nav */}
  <Link
    to="/cart"
    className="relative p-2 text-gray-700 hover:text-dark-red transition-colors hi
```

```
>
  <svg
    className="w-6 h-6"
    fill="none"
    stroke="currentColor"
    viewBox="0 0 24 24"
  >
    <path
      strokeLinecap="round"
      strokeLinejoin="round"
      strokeWidth={2}
      d="M3 3h2l.4 2M7 13h10l4-8H5.4m0 0L7 13m0 0l-2.5 5M7 13l2.5 5m0 0h8"
    />
  </svg>
  {itemCount > 0 && (
    <span className="absolute -top-1 -right-1 bg-error-red text-white text-xs ro
      {itemCount > 99 ? "99+" : itemCount}
    </span>
  )}
</Link>

{/* User Menu */}
{isAuthenticated ? (
  <div className="relative">
    <button
      onClick={() =>
        setIsProfileDropdownOpen(!isProfileDropdownOpen)
      }
      className="flex items-center space-x-2 p-2 rounded-lg hover:bg-gray-100 tr
    >
      <div className="w-8 h-8 bg-dark-red rounded-full flex items-center justify
        <span className="text-white text-sm font-medium">
          {user?.name?.charAt(0).toUpperCase()}
        </span>
      </div>
      <span className="hidden sm:block text-gray-700">
        {user?.name}
      </span>
      <svg
        className="w-4 h-4 text-gray-500"
        fill="none"
        stroke="currentColor"
        viewBox="0 0 24 24"
      >
        <path
          strokeLinecap="round"
```

```
                strokeLinejoin="round"
                strokeWidth={2}
                d="M19 9l-7 7-7-7"
              />
            </svg>
          </button>

          {/* Dropdown Menu */}
          {isProfileDropdownOpen && (
            <div className="absolute right-0 mt-2 w-48 bg-white rounded-lg shadow-lg 
              <Link
                to="/profile"
                onClick={() => setIsProfileDropdownOpen(false)}
                className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
              >
                Profile
              </Link>
              <Link
                to="/orders"
                onClick={() => setIsProfileDropdownOpen(false)}
                className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
              >
                My Orders
              </Link>
              {user?.role === "admin" && (
                <Link
                  to="/admin"
                  onClick={() => setIsProfileDropdownOpen(false)}
                  className="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100"
                >
                  Admin Panel
                </Link>
              )}
              <hr className="my-1" />
              <button
                onClick={handleLogout}
                className="block w-full text-left px-4 py-2 text-sm text-gray-700 hove
              >
                Logout
              </button>
            </div>
          )}
        </div>
      ) : (
        <div className="flex items-center space-x-2">
          <button
```

```
                  onClick={() => onOpenAuth("login")}
                  className="text-gray-700 hover:text-dark-red transition-colors"
                >
                  Login
                </button>
                <button
                  onClick={() => onOpenAuth("register")}
                  className="bg-dark-red text-white px-4 py-2 rounded-lg hover:bg-hover-red
                >
                  Sign Up
                </button>
              </div>
            )}
          </div>
        </div>
      </div>
    </header>
  );
}
```

17. src/components/customer/layout/Footer.jsx

```
import React from "react";
import { Link } from "react-router-dom";

export default function Footer() {
  return (
    <footer className="bg-gray-800 text-white">
      <div className="container mx-auto px-4 py-8">
        <div className="grid grid-cols-1 md:grid-cols-4 gap-8">
          {/* Company Info */}
          <div>
            <div className="flex items-center space-x-2 mb-4">
              <img
                src="https://cornflowerblue-lion-884998.hostingersite.com/wp-content/uploads
                alt="The Food City"
                className="h-8 w-8 object-contain"
              />
              <span className="text-lg font-bold">The Food City</span>
            </div>
            <p className="text-gray-300 text-sm mb-4">
              Delicious food delivered to your doorstep. Experience the best
              flavors from around the world.
            </p>
            <div className="flex space-x-4">
              <a
      href="#"
      className="text-gray-300 hover:text-white transition-colors"
    >
      <svg
        className="w-5 h-5"
        fill="currentColor"
        viewBox="0 0 24 24"
      >
        <path d="M24 4.557c-.883.392-1.832.656-2.828.775 1.017-.609 1.798-1.574 2.
      </svg>
    </a>
    <a
      href="#"
      className="text-gray-300 hover:text-white transition-colors"
    >
      <svg
        className="w-5 h-5"
        fill="currentColor"
        viewBox="0 0 24 24"
      >
        <path d="M22.46 6c-.77.35-1.6.58-2.46.69.88-.53 1.56-1.37 1.88-2.38-.83.5-
      </svg>
    </a>
    <a
      href="#"
      className="text-gray-300 hover:text-white transition-colors"
    >
      <svg
        className="w-5 h-5"
        fill="currentColor"
        viewBox="0 0 24 24"
      >
        <path d="M12.017 0C5.396 0 .029 5.367.029 11.987c0 5.079 3.158 9.417 7.618
      </svg>
    </a>
  </div>
</div>

{/* Quick Links */}
<div>
  <h3 className="text-lg font-semibold mb-4">Quick Links</h3>
  <ul className="space-y-2">
    <li>
      <Link
        to="/"
        className="text-gray-300 hover:text-white transition-colors text-sm"
```

```
            >
              Home
            </Link>
          </li>
          <li>
            <Link
              to="/menu"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Menu
            </Link>
          </li>
          <li>
            <Link
              to="/deals"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Deals & Offers
            </Link>
          </li>
          <li>
            <Link
              to="/about"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              About Us
            </Link>
          </li>
          <li>
            <Link
              to="/contact"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Contact Us
            </Link>
          </li>
        </ul>
      </div>

      {/* Legal */}
      <div>
        <h3 className="text-lg font-semibold mb-4">Legal</h3>
        <ul className="space-y-2">
          <li>
            <Link
              to="/terms"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Terms & Conditions
            </Link>
          </li>
          <li>
            <Link
              to="/privacy"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Privacy Policy
            </Link>
          </li>
          <li>
            <Link
              to="/refund"
              className="text-gray-300 hover:text-white transition-colors text-sm"
            >
              Refund Policy
            </Link>
          </li>
        </ul>
      </div>

      {/* Contact Info */}
      <div>
        <h3 className="text-lg font-semibold mb-4">Contact Info</h3>
        <div className="space-y-2 text-sm text-gray-300">
          <div className="flex items-center space-x-2">
            <svg
              className="w-4 h-4"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M17.657 16.657L13.414 20.9a1.998 1.998 0 01-2.827 0l-4.244-4.243a8 8
              />
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M15 11a3 3 0 11-6 0 3 3 0 016 0z"
          />
        </svg>
        <span>123 Food Street, Mumbai, India</span>
      </div>
      <div className="flex items-center space-x-2">
        <svg
          className="w-4 h-4"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M3 5a2 2 0 012-2h3.28a1 1 0 01.948.684l1.498 4.493a1 1 0 01-.502 1.2
          />
        </svg>
        <span>+91 98765 43210</span>
      </div>
      <div className="flex items-center space-x-2">
        <svg
          className="w-4 h-4"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            d="M3 8l7.89 5.26a2 2 0 002.22 0L21 8M5 19h14a2 2 0 002-2V7a2 2 0 00-2-
          />
        </svg>
        <span>info@thefoodcity.com</span>
      </div>
    </div>
  </div>
</div>

{/* Bottom Bar */}
<div className="border-t border-gray-700 mt-8 pt-8 flex flex-col md:flex-row justify
  <p className="text-gray-300 text-sm">
    © 2025 The Food City. All rights reserved.
  </p>
  <p className="text-gray-300 text-sm mt-2 md:mt-0">
```

```
            Designed and developed by{" "}
            <a
              href="https://enegixwebsolutions.com/"
              target="_blank"
              rel="noopener noreferrer"
              className="text-white hover:text-dark-red transition-colors font-medium"
            >
              Enegix Web Solutions
            </a>
          </p>
        </div>
      </div>
    </footer>
  );
}
```

18. src/components/customer/layout/CustomerLayout.jsx

```
import React, { useState } from "react";
import { Outlet } from "react-router-dom";
import Header from "./Header.jsx";
import BottomNavigation from "./BottomNavigation.jsx";
import Footer from "./Footer.jsx";
import AuthModal from "../../auth/AuthModal.jsx";
import { ToastProvider } from "../../shared/Toast.jsx";

export default function CustomerLayout() {
  const [showAuthModal, setShowAuthModal] = useState(false);
  const [authMode, setAuthMode] = useState("login");

  const openAuthModal = (mode = "login") => {
    setAuthMode(mode);
    setShowAuthModal(true);
  };

  return (
    <ToastProvider>
      <div className="min-h-screen bg-light-gray flex flex-col">
        {/* Header */}
        <Header onOpenAuth={openAuthModal} />

        {/* Main Content */}
        <main className="flex-1 pb-20 md:pb-8">
          <Outlet />
        </main>
        {/* Footer - Hidden on mobile, shown on desktop */}
        <div className="hidden md:block">
          <Footer />
        </div>

        {/* Bottom Navigation - Mobile only */}
        <div className="md:hidden">
          <BottomNavigation onOpenAuth={openAuthModal} />
        </div>

        {/* Auth Modal */}
        <AuthModal
          isOpen={showAuthModal}
          onClose={() => setShowAuthModal(false)}
          initialMode={authMode}
        />
      </div>
    </ToastProvider>
  );
}
```

19. src/pages/customer/HomePage.jsx

```
import React, { useState, useEffect } from "react";
import { useSearchParams } from "react-router-dom";
import { Container } from "../../components/shared/Layout.jsx";
import { mockAPI } from "../../services/mockApi.js";
import HeroSection from "../../components/customer/HeroSection.jsx";
import SimpleSearchBox from "../../components/customer/SimpleSearchBox.jsx";
import SpecialOffersSlider from "../../components/customer/SpecialOffersSlider.jsx";
import CategorySlider from "../../components/customer/CategorySlider.jsx";
import { InlineLoader } from "../../components/shared/LoadingSpinner.jsx";
import {
  FaFire,
  FaBowlFood,
  FaHotjar,
  FaDrumstickBite,
  FaUtensils,
} from "react-icons/fa6";
import { FaPepperHot, FaClock, FaStar, FaDollarSign } from "react-icons/fa";

export default function HomePage() {
  const [menuItems, setMenuItems] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchParams] = useSearchParams();
```

```
useEffect(() => {
  loadMenuItems();
}, [searchParams]);

const loadMenuItems = async () => {
  try {
    setLoading(true);
    const response = await mockAPI.getMenuItems();
    if (response.success) {
      setMenuItems(response.data);
    } else {
      setError("Failed to load menu items");
    }
  } catch (err) {
    setError("Something went wrong while loading menu items");
    console.error("Error loading menu items:", err?.message || String(err));
  } finally {
    setLoading(false);
  }
};

// Group items by category
const groupedItems = menuItems.reduce((acc, item) => {
  if (!acc[item.category]) {
    acc[item.category] = [];
  }
  acc[item.category].push(item);
  return acc;
}, {});

// Get trending items (highest rated items)
const trendingItems = menuItems
  .filter((item) => item.rating >= 4.3)
  .sort((a, b) => b.rating - a.rating)
  .slice(0, 8);

if (loading) {
  return (
    <Container className="py-8">
      <InlineLoader text="Loading delicious food..." />
    </Container>
  );
}

if (error) {
  return (
    <Container className="py-8">
      <div className="text-center">
        <div className="bg-error-red text-white p-4 rounded-lg max-w-md mx-auto">
          <h3 className="font-semibold mb-2">Oops! Something went wrong</h3>
          <p className="text-sm">{error}</p>
          <button
            onClick={loadMenuItems}
            className="mt-3 bg-white text-error-red px-4 py-2 rounded font-medium hover:bg
          >
            Try Again
          </button>
        </div>
      </div>
    </Container>
  );
}

return (
  <Container className="py-4 md:py-8">
    {/* Hero Section */}
    <HeroSection />

    {/* Simple Search Box */}
    <SimpleSearchBox />

    {/* Special Offers Slider */}
    <SpecialOffersSlider />

    {/* Trending Items */}
    {trendingItems.length > 0 && (
      <CategorySlider
        category="trending"
        title={
          <span className="flex items-center gap-2">
            <FaFire className="text-dark-red" />
            Trending Now
          </span>
        }
        items={trendingItems}
      />
    )}

    {/* Category Sections */}
    {Object.entries(groupedItems).map(([category, items]) => {
      // Category icons
  const categoryIcons = {
    Indian: <FaBowlFood className="text-dark-red" />,
    Chinese: <FaHotjar className="text-dark-red" />,
    South: <FaPepperHot className="text-dark-red" />,
    Tandoor: <FaDrumstickBite className="text-dark-red" />,
  };

  return (
    <CategorySlider
      key={category}
      category={category}
      title={
        <span className="flex items-center gap-2">
          {categoryIcons[category] || (
            <FaUtensils className="text-dark-red" />
          )}
          {category} Cuisine
        </span>
      }
      items={items}
    />
  );
})}

{/* Quick Actions */}
<div className="mt-12 bg-white rounded-lg shadow-subtle p-6">
  <h2 className="text-2xl font-bold text-dark-red mb-4 text-center">
    Why Choose The Food City?
  </h2>

  <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
    <div className="text-center">
      <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-cent
        <FaClock className="text-2xl text-white" />
      </div>
      <h3 className="font-semibold text-gray-800 mb-2">Fast Delivery</h3>
      <p className="text-gray-600 text-sm">
        Get your food delivered in 30-45 minutes, hot and fresh.
      </p>
    </div>

    <div className="text-center">
      <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-cent
        <FaStar className="text-2xl text-white" />
      </div>
      <h3 className="font-semibold text-gray-800 mb-2">Quality Food</h3>
```

```
            <p className="text-gray-600 text-sm">
              Fresh ingredients and authentic recipes for the best taste.
            </p>
          </div>

          <div className="text-center">
            <div className="w-16 h-16 bg-red-900 rounded-full flex items-center justify-cent
              <FaDollarSign className="text-2xl text-white" />
            </div>
            <h3 className="font-semibold text-gray-800 mb-2">Great Prices</h3>
            <p className="text-gray-600 text-sm">
              Affordable prices with regular deals and discounts.
            </p>
          </div>
        </div>
      </div>
    </Container>
  );
}
```

20. src/components/admin/layout/AdminLayout.jsx

```
import React, { useState } from "react";
import { Outlet, Link, useLocation } from "react-router-dom";
import { Container } from "../../shared/Layout.jsx";
import { useAuth } from "../../../context/AuthContext.jsx";
import {
  FaTachometerAlt,
  FaShoppingBag,
  FaUtensils,
  FaTruck,
  FaUsers,
  FaChartLine,
  FaUser,
  FaSignOutAlt,
  FaUserTie,
  FaBars,
  FaTimes,
} from "react-icons/fa";

export default function AdminLayout() {
  const { user, logout } = useAuth();
  const location = useLocation();
  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  const menuItems = [
  { path: "/admin", icon: FaTachometerAlt, label: "Dashboard", exact: true },
  { path: "/admin/orders", icon: FaShoppingBag, label: "Orders" },
  { path: "/admin/menu", icon: FaUtensils, label: "Menu Management" },
  { path: "/admin/delivery", icon: FaUserTie, label: "Delivery Staff" },
  { path: "/admin/customers", icon: FaUsers, label: "Customers" },
  { path: "/admin/reports", icon: FaChartLine, label: "Reports" },
];

const isActiveRoute = (path, exact = false) => {
  if (exact) {
    return location.pathname === path;
  }
  return location.pathname.startsWith(path);
};

const closeMobileMenu = () => {
  setIsMobileMenuOpen(false);
};

return (
  <div className="min-h-screen bg-light-gray">
    {/* Top Navigation */}
    <div className="bg-white shadow-sm border-b">
      <Container>
        <div className="flex items-center justify-between h-16">
          <div className="flex items-center gap-3">
            {/* Mobile Menu Button */}
            <button
              onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}
              className="lg:hidden p-2 text-gray-600 hover:text-dark-red"
            >
              {isMobileMenuOpen ? (
                <FaTimes size={20} />
              ) : (
                <FaBars size={20} />
              )}
            </button>

            <div className="text-lg lg:text-xl font-bold text-dark-red">
              The Food City
            </div>
            <div className="hidden sm:block text-xs lg:text-sm text-gray-500 bg-red-100 px
              Admin Panel
            </div>
          </div>
          <div className="flex items-center gap-2 lg:gap-4">
```

```
            <div className="flex items-center gap-2">
              <div className="w-7 h-7 lg:w-8 lg:h-8 bg-dark-red rounded-full flex items-ce
                <FaUser className="text-white text-xs lg:text-sm" />
              </div>
              <div className="hidden md:block">
                <p className="text-sm font-medium text-gray-800">
                  {user?.name}
                </p>
                <p className="text-xs text-gray-500 capitalize">
                  {user?.role}
                </p>
              </div>
            </div>
            <button
              onClick={logout}
              className="flex items-center gap-1 lg:gap-2 text-gray-600 hover:text-red-60
              title="Logout"
            >
              <FaSignOutAlt className="text-sm lg:text-base" />
              <span className="hidden lg:inline text-sm">Logout</span>
            </button>
          </div>
        </div>
      </Container>
    </div>

    <div className="flex relative">
      {/* Mobile Overlay */}
      {isMobileMenuOpen && (
        <div
          className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"
          onClick={closeMobileMenu}
        />
      )}

      {/* Sidebar */}
      <div
        className={`
        fixed lg:static inset-y-0 left-0 z-50
        w-64 bg-white shadow-lg lg:shadow-sm
        transform transition-transform duration-300 ease-in-out
        ${isMobileMenuOpen
            ? "translate-x-0"
            : "-translate-x-full lg:translate-x-0"
          }
        min-h-screen lg:min-h-0
```

```
          `}
        >
          {/* Mobile Close Button */}
          <div className="lg:hidden flex justify-end p-4">
            <button
              onClick={closeMobileMenu}
              className="p-2 text-gray-600 hover:text-dark-red"
            >
              <FaTimes size={18} />
            </button>
          </div>

          <nav className="p-4 pt-0 lg:pt-4">
            <ul className="space-y-2">
              {menuItems.map((item) => {
                const Icon = item.icon;
                return (
                  <li key={item.path}>
                    <Link
                      to={item.path}
                      onClick={closeMobileMenu}
                      className={`flex items-center gap-3 px-4 py-3 rounded-lg transition-co
                          ? "bg-dark-red text-white"
                          : "text-gray-700 hover:bg-red-50 hover:text-dark-red"
                        }`}
                    >
                      <Icon className="text-lg flex-shrink-0" />
                      <span className="text-sm lg:text-base">{item.label}</span>
                    </Link>
                  </li>
                );
              })}
            </ul>
          </nav>
        </div>

        {/* Main Content */}
        <div className="flex-1 lg:ml-0">
          <main className="p-3 sm:p-4 lg:p-6">
            <Outlet />
          </main>
        </div>
      </div>
    </div>
  );
}
```

21. src/components/shared/ConfirmDialog.jsx

```
import React from "react";
import { FaTrash, FaExclamationTriangle, FaInfoCircle } from "react-icons/fa";

export default function ConfirmDialog({
  isOpen,
  onClose,
  onConfirm,
  title = "Confirm Action",
  message = "Are you sure you want to proceed?",
  confirmText = "Confirm",
  cancelText = "Cancel",
  type = "warning", // 'warning', 'danger', 'info'
}) {
  if (!isOpen) return null;

  const typeStyles = {
    warning: {
      icon: <FaExclamationTriangle className="text-white" />,
      confirmButton: "bg-warm-yellow hover:bg-yellow-600 text-black",
      iconBg: "bg-warm-yellow",
    },
    danger: {
      icon: <FaTrash className="text-white" />,
      confirmButton: "bg-red-900 hover:bg-red-800 text-white",
      iconBg: "bg-red-900",
    },
    info: {
      icon: <FaInfoCircle className="text-white" />,
      confirmButton: "bg-info-blue hover:bg-blue-600 text-white",
      iconBg: "bg-info-blue",
    },
  };

  const currentStyle = typeStyles[type];

  const handleConfirm = () => {
    onConfirm();
    onClose();
  };

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-
      <div className="bg-white rounded-lg shadow-lg max-w-md w-full">
        <div className="p-6">
```

```
          {/* Icon and Title */}
          <div className="flex items-center mb-4">
            <div
              className={`w-10 h-10 ${currentStyle.iconBg} rounded-full flex items-center ju
            >
              {currentStyle.icon}
            </div>
            <h3 className="text-lg font-semibold text-gray-900">{title}</h3>
          </div>

          {/* Message */}
          <p className="text-gray-600 mb-6">{message}</p>

          {/* Buttons */}
          <div className="flex space-x-3">
            <button
              onClick={onClose}
              className="flex-1 bg-gray-200 text-gray-800 py-2 px-4 rounded-lg hover:bg-gray
            >
              {cancelText}
            </button>
            <button
              onClick={handleConfirm}
              className={`flex-1 py-2 px-4 rounded-lg transition-colors ${currentStyle.confi
            >
              {confirmText}
            </button>
          </div>
        </div>
      </div>
    </div>
  );
}

// Hook for easier usage
export function useConfirmDialog() {
  const [dialog, setDialog] = React.useState({
    isOpen: false,
    title: "",
    message: "",
    onConfirm: () => {},
    type: "warning",
  });

  const showConfirm = (options) => {
    return new Promise((resolve) => {
      setDialog({
        isOpen: true,
        title: options.title || "Confirm Action",
        message: options.message || "Are you sure?",
        type: options.type || "warning",
        onConfirm: () => {
          resolve(true);
          setDialog((prev) => ({ ...prev, isOpen: false }));
        },
      });
    });
  };

  const closeDialog = () => {
    setDialog((prev) => ({ ...prev, isOpen: false }));
  };

  const ConfirmDialogComponent = () => (
    <ConfirmDialog
      isOpen={dialog.isOpen}
      onClose={closeDialog}
      onConfirm={dialog.onConfirm}
      title={dialog.title}
      message={dialog.message}
      type={dialog.type}
    />
  );

  return { showConfirm, ConfirmDialogComponent };
}
```

22. src/components/shared/Toast.jsx

```
import React, { useState, useEffect } from "react";

// Toast context for managing toasts globally
const ToastContext = React.createContext();

export function ToastProvider({ children }) {
  const [toasts, setToasts] = useState([]);

  const addToast = (message, type = "info", duration = 3000) => {
    const id = Date.now();
    const toast = { id, message, type, duration };

    setToasts((prev) => [...prev, toast]);
    // Auto remove toast after duration
    setTimeout(() => {
      removeToast(id);
    }, duration);
  };

  const removeToast = (id) => {
    setToasts((prev) => prev.filter((toast) => toast.id !== id));
  };

  const value = {
    addToast,
    removeToast,
  };

  return (
    <ToastContext.Provider value={value}>
      {children}
      <ToastContainer toasts={toasts} onRemove={removeToast} />
    </ToastContext.Provider>
  );
}

// Hook to use toast
export function useToast() {
  const context = React.useContext(ToastContext);
  if (!context) {
    throw new Error("useToast must be used within a ToastProvider");
  }
  return context;
}

// Toast container component
function ToastContainer({ toasts, onRemove }) {
  if (toasts.length === 0) return null;

  return (
    <div className="fixed top-4 right-4 z-50 space-y-2">
      {toasts.map((toast) => (
        <Toast
          key={toast.id}
          message={toast.message}
          type={toast.type}
          onClose={() => onRemove(toast.id)}
        />
    ))}
  </div>
 );
}

// Individual toast component
function Toast({ message, type, onClose }) {
  const [isVisible, setIsVisible] = useState(false);

  useEffect(() => {
    // Trigger animation
    setTimeout(() => setIsVisible(true), 10);
  }, []);

  const handleClose = () => {
    setIsVisible(false);
    setTimeout(onClose, 300); // Wait for animation to complete
  };

  const typeStyles = {
    success: "bg-success-green text-white",
    error: "bg-error-red text-white",
    warning: "bg-warm-yellow text-black",
    info: "bg-info-blue text-white",
  };

  const icons = {
    success: " ",
    error: " ",
    warning: " ",
    info: " ",
  };

  return (
    <div
      className={`
        transform transition-all duration-300 ease-in-out
        ${
          isVisible ? "translate-x-0 opacity-100" : "translate-x-full opacity-0"
        }
        ${typeStyles[type]}
        px-4 py-3 rounded-lg shadow-lg max-w-sm min-w-[300px]
        flex items-center justify-between
      `}
    >
      <div className="flex items-center">
```

```
          <span className="mr-2 text-lg">{icons[type]}</span>
          <span className="text-sm font-medium">{message}</span>
        </div>
        <button
          onClick={handleClose}
          className="ml-4 text-lg hover:opacity-70 transition-opacity"
        >
          ×
        </button>
      </div>
    );
}

export default Toast;
```

23. src/components/shared/Layout.jsx

```
import React from "react";

// Main layout wrapper
export function Layout({ children, className = "" }) {
  return (
    <div className={`min-h-screen bg-light-gray ${className}`}>{children}</div>
  );
}

// Container component
export function Container({ children, size = "default", className = "" }) {
  const sizeClasses = {
    sm: "max-w-2xl",
    default: "max-w-6xl",
    lg: "max-w-7xl",
    full: "max-w-full",
  };

  return (
    <div className={`container mx-auto px-4 ${sizeClasses[size]} ${className}`}>
      {children}
    </div>
  );
}

// Card component
export function Card({ children, className = "", padding = "default" }) {
  const paddingClasses = {
    none: "",
    sm: "p-4",
    default: "p-6",
    lg: "p-8",
  };

  return (
    <div
      className={`bg-white rounded-lg shadow-subtle ${paddingClasses[padding]} ${className}
    >
      {children}
    </div>
  );
}

// Section component
export function Section({ children, className = "", padding = "default" }) {
  const paddingClasses = {
    none: "",
    sm: "py-4",
    default: "py-8",
    lg: "py-12",
  };

  return (
    <section className={`${paddingClasses[padding]} ${className}`}>
      {children}
    </section>
  );
}

// Grid component
export function Grid({ children, cols = 1, gap = 4, className = "" }) {
  const colClasses = {
    1: "grid-cols-1",
    2: "grid-cols-1 sm:grid-cols-2",
    3: "grid-cols-1 sm:grid-cols-2 lg:grid-cols-3",
    4: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4",
    5: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-5",
    6: "grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-6",
  };

  const gapClasses = {
    2: "gap-2",
    4: "gap-4",
    6: "gap-6",
    8: "gap-8",
  };

  return (
    <div className={`grid ${colClasses[cols]} ${gapClasses[gap]} ${className}`}>
      {children}
    </div>
  );
}

// Flex component
export function Flex({
  children,
  direction = "row",
  align = "start",
  justify = "start",
  wrap = false,
  gap = 0,
  className = "",
}) {
  const directionClasses = {
    row: "flex-row",
    col: "flex-col",
    "row-reverse": "flex-row-reverse",
    "col-reverse": "flex-col-reverse",
  };

  const alignClasses = {
    start: "items-start",
    center: "items-center",
    end: "items-end",
    stretch: "items-stretch",
  };

  const justifyClasses = {
    start: "justify-start",
    center: "justify-center",
    end: "justify-end",
    between: "justify-between",
    around: "justify-around",
    evenly: "justify-evenly",
  };

  const gapClasses = {
    0: "",
    1: "gap-1",
    2: "gap-2",
    3: "gap-3",
    4: "gap-4",
    6: "gap-6",
    8: "gap-8",
  };

  return (
    <div
      className={`
      flex
      ${directionClasses[direction]}
      ${alignClasses[align]}
      ${justifyClasses[justify]}
      ${wrap ? "flex-wrap" : ""}
      ${gapClasses[gap]}
      ${className}
    `}
    >
      {children}
    </div>
  );
}

// Button component
export function Button({
  children,
  variant = "primary",
  size = "md",
  disabled = false,
  loading = false,
  fullWidth = false,
  onClick,
  type = "button",
  className = "",
  ...props
}) {
  const baseClasses =
    "font-medium rounded-lg transition-colors focus:outline-none focus:ring-2 focus:ring-of

  const variantClasses = {
    primary: "bg-dark-red text-white hover:bg-hover-red focus:ring-dark-red",
    secondary:
      "bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500",
    success:
      "bg-success-green text-white hover:bg-green-600 focus:ring-success-green",
    danger: "bg-error-red text-white hover:bg-red-600 focus:ring-error-red",
    warning:
      "bg-warm-yellow text-black hover:bg-yellow-600 focus:ring-warm-yellow",
    outline:
      "border-2 border-dark-red text-dark-red hover:bg-dark-red hover:text-white focus:ring-
  };

  const sizeClasses = {
    sm: "px-3 py-1.5 text-sm",
    md: "px-4 py-2 text-sm",
    lg: "px-6 py-3 text-base",
  };

  return (
    <button
      type={type}
      onClick={onClick}
      disabled={disabled || loading}
      className={`
        ${baseClasses}
        ${variantClasses[variant]}
        ${sizeClasses[size]}
        ${fullWidth ? "w-full" : ""}
        ${className}
      `}
      {...props}
    >
      {loading ? (
        <div className="flex items-center justify-center">
          <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-current mr-2">
          Loading...
        </div>
      ) : (
        children
      )}
    </button>
  );
}
```

24. src/components/shared/ErrorBoundary.jsx

```
import React from "react";

class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null, errorInfo: null };
  }

  static getDerivedStateFromError(error) {
    // Update state so the next render will show the fallback UI
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // Log error details
    console.error(
      "ErrorBoundary caught an error:",
      error?.message || String(error),
      errorInfo?.componentStack || "No stack trace"
    );

    this.setState({
      error: error,
      errorInfo: errorInfo,
    });
  }

  render() {
    if (this.state.hasError) {
      // Custom fallback UI
      return (
        <div className="min-h-screen flex items-center justify-center bg-light-gray p-4">
          <div className="bg-white rounded-lg shadow-subtle p-8 max-w-md w-full text-center'
            <div className="mb-6">
              <div className="w-16 h-16 bg-error-red rounded-full flex items-center justify-
                <span className="text-white text-2xl"> </span>
              </div>
              <h2 className="text-2xl font-bold text-error-red mb-2">
                Oops! Something went wrong
              </h2>
              <p className="text-gray-600 mb-4">
                We're sorry, but something unexpected happened. Please try
                refreshing the page.
              </p>
            </div>

            <div className="space-y-3">
              <button
                onClick={() => window.location.reload()}
                className="w-full bg-dark-red text-white py-2 px-4 rounded-lg hover:bg-hover
              >
                Refresh Page
              </button>
              <button
                onClick={() => (window.location.href = "/")}
                className="w-full bg-gray-500 text-white py-2 px-4 rounded-lg hover:bg-gray-
              >
                Go to Home
              </button>
            </div>

            {/* Show error details in development */}
            {process.env.NODE_ENV === "development" && (
              <details className="mt-6 text-left">
                <summary className="cursor-pointer text-sm text-gray-500 hover:text-gray-70
                  Error Details (Development Only)
                </summary>
                <div className="mt-2 p-3 bg-gray-100 rounded text-xs overflow-auto max-h-40
                  <div className="mb-2">
                    <strong>Error:</strong>{" "}
                    {this.state.error && this.state.error.toString()}
                  </div>
                  <div>
                    <strong>Stack Trace:</strong>
                    <pre className="whitespace-pre-wrap">
                      {this.state.errorInfo?.componentStack ||
                        "No stack trace available"}
                    </pre>
                  </div>
                </div>
              </details>
            )}
          </div>
        </div>
      );
    }

    // No error, render children normally
    return this.props.children;
  }
}

export default ErrorBoundary;
```

25. src/components/shared/FormInput.jsx

```
import React, { useState } from "react";
```

```
// Basic input component
export function FormInput({
  label,
  name,
  type = "text",
  value,
  onChange,
  error,
  placeholder,
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  const [showPassword, setShowPassword] = useState(false);

  const inputType = type === "password" && showPassword ? "text" : type;

  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <div className="relative">
        <input
          type={inputType}
          name={name}
          value={value}
          onChange={onChange}
          placeholder={placeholder}
          required={required}
          disabled={disabled}
          className={`
            w-full px-3 py-2 border rounded-lg transition-colors
            focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
            disabled:bg-gray-100 disabled:cursor-not-allowed
            ${
              error
                ? "border-error-red focus:ring-error-red"
                : "border-gray-300 hover:border-gray-400"
            }
          `}
          {...props}
        />

        {type === "password" && (
          <button
            type="button"
            onClick={() => setShowPassword(!showPassword)}
            className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-400 hov
          >
            {showPassword ? " " : " "}
          </button>
        )}
      </div>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Textarea component
export function FormTextarea({
  label,
  name,
  value,
  onChange,
  error,
  placeholder,
  required = false,
  disabled = false,
  rows = 3,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <textarea
        name={name}
        value={value}
        onChange={onChange}
```

```
          placeholder={placeholder}
          required={required}
          disabled={disabled}
          rows={rows}
          className={`
            w-full px-3 py-2 border rounded-lg transition-colors resize-vertical
            focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
            disabled:bg-gray-100 disabled:cursor-not-allowed
            ${
              error
                ? "border-error-red focus:ring-error-red"
                : "border-gray-300 hover:border-gray-400"
            }
          `}
          {...props}
        />

        {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
      </div>
  );
}

// Select component
export function FormSelect({
  label,
  name,
  value,
  onChange,
  error,
  options = [],
  placeholder = "Select an option",
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-1">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <select
```

```
        name={name}
        value={value}
        onChange={onChange}
        required={required}
        disabled={disabled}
        className={`
          w-full px-3 py-2 border rounded-lg transition-colors
          focus:outline-none focus:ring-2 focus:ring-dark-red focus:border-transparent
          disabled:bg-gray-100 disabled:cursor-not-allowed
          ${
            error
              ? "border-error-red focus:ring-error-red"
              : "border-gray-300 hover:border-gray-400"
          }
        `}
        {...props}
      >
        <option value="">{placeholder}</option>
        {options.map((option) => (
          <option key={option.value} value={option.value}>
            {option.label}
          </option>
        ))}
      </select>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Checkbox component
export function FormCheckbox({
  label,
  name,
  checked,
  onChange,
  error,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      <label className="flex items-center cursor-pointer">
        <input
          type="checkbox"
          name={name}
          checked={checked}
          onChange={onChange}
          disabled={disabled}
          className={`
            mr-2 h-4 w-4 text-dark-red border-gray-300 rounded
            focus:ring-dark-red focus:ring-2
            disabled:cursor-not-allowed
          `}
          {...props}
        />
        <span className="text-sm text-gray-700">{label}</span>
      </label>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}

// Radio group component
export function FormRadioGroup({
  label,
  name,
  value,
  onChange,
  error,
  options = [],
  required = false,
  disabled = false,
  className = "",
  ...props
}) {
  return (
    <div className={`mb-4 ${className}`}>
      {label && (
        <label className="block text-sm font-medium text-gray-700 mb-2">
          {label}
          {required && <span className="text-error-red ml-1">*</span>}
        </label>
      )}

      <div className="space-y-2">
        {options.map((option) => (
          <label
            key={option.value}
            className="flex items-center cursor-pointer"
          >
            <input
              type="radio"
              name={name}
              value={option.value}
              checked={value === option.value}
              onChange={onChange}
              disabled={disabled}
              className={`
                mr-2 h-4 w-4 text-dark-red border-gray-300
                focus:ring-dark-red focus:ring-2
                disabled:cursor-not-allowed
              `}
              {...props}
            />
            <span className="text-sm text-gray-700">{option.label}</span>
          </label>
        ))}
      </div>

      {error && <p className="mt-1 text-sm text-error-red">{error}</p>}
    </div>
  );
}
```

26. src/components/shared/LoadingSpinner.jsx

```
import React from "react";

export default function LoadingSpinner({
  size = "md",
  color = "dark-red",
  text = "",
}) {
  const sizeClasses = {
    sm: "h-4 w-4",
    md: "h-8 w-8",
    lg: "h-12 w-12",
    xl: "h-16 w-16",
  };

  const colorClasses = {
    "dark-red": "border-dark-red",
    white: "border-white",
    gray: "border-gray-500",
  };
  return (
    <div className="flex flex-col items-center justify-center">
      <div
        className={`animate-spin rounded-full border-b-2 ${sizeClasses[size]} ${colorClasses
      ></div>
      {text && <p className="mt-2 text-sm text-gray-600">{text}</p>}
    </div>
  );
}

// Full screen loading component
export function FullScreenLoader({ text = "Loading..." }) {
  return (
    <div className="fixed inset-0 bg-white bg-opacity-90 flex items-center justify-center z-
      <LoadingSpinner size="lg" text={text} />
    </div>
  );
}

// Inline loading component
export function InlineLoader({ text = "Loading..." }) {
  return (
    <div className="flex items-center justify-center py-8">
      <LoadingSpinner size="md" text={text} />
    </div>
  );
}
```

27. src/utils/validation.js

```
// Form validation utilities

// Email validation
export const validateEmail = (email) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!email) return "Email is required";
  if (!emailRegex.test(email)) return "Please enter a valid email address";
  return "";
};

// Password validation
export const validatePassword = (password) => {
  if (!password) return "Password is required";
  if (password.length < 8) return "Password must be at least 8 characters long";
  if (!/(?=.*[a-z])/.test(password))
```

```
    return "Password must contain at least one lowercase letter";
  if (!/(?=.*[A-Z])/.test(password))
    return "Password must contain at least one uppercase letter";
  if (!/(?=.*\d)/.test(password))
    return "Password must contain at least one number";
  if (!/(?=.*[!@#$%^&*])/.test(password))
    return "Password must contain at least one special character";
  return "";
};

// Phone validation
export const validatePhone = (phone) => {
  const phoneRegex = /^[\+]?[1-9][\d]{0,15}$/;
  if (!phone) return "Phone number is required";
  if (!phoneRegex.test(phone.replace(/[\s\-\(\)]/g, "")))
    return "Please enter a valid phone number";
  return "";
};

// Name validation
export const validateName = (name) => {
  if (!name) return "Name is required";
  if (name.length < 2) return "Name must be at least 2 characters long";
  if (!/^[a-zA-Z\s]+$/.test(name))
    return "Name can only contain letters and spaces";
  return "";
};

// PIN code validation
export const validatePinCode = (pinCode) => {
  const pinRegex = /^[1-9][0-9]{5}$/;
  if (!pinCode) return "PIN code is required";
  if (!pinRegex.test(pinCode)) return "Please enter a valid 6-digit PIN code";
  return "";
};

// Address validation
export const validateAddress = (address) => {
  const errors = {};

  if (!address.street || address.street.trim().length < 5) {
    errors.street = "Street address must be at least 5 characters long";
  }

  if (!address.city || address.city.trim().length < 2) {
    errors.city = "City is required";
  }

  if (!address.state || address.state.trim().length < 2) {
    errors.state = "State is required";
  }

  const pinError = validatePinCode(address.pinCode);
  if (pinError) {
    errors.pinCode = pinError;
  }

  return errors;
};

// Price validation
export const validatePrice = (price) => {
  if (!price && price !== 0) return "Price is required";
  if (isNaN(price) || price < 0) return "Price must be a valid positive number";
  if (price > 10000) return "Price cannot exceed 10,000";
  return "";
};

// Quantity validation
export const validateQuantity = (quantity) => {
  if (!quantity && quantity !== 0) return "Quantity is required";
  if (isNaN(quantity) || quantity < 1) return "Quantity must be at least 1";
  if (quantity > 50) return "Quantity cannot exceed 50";
  return "";
};

// Promo code validation
export const validatePromoCode = (code) => {
  if (!code) return "Promo code is required";
  if (code.length < 3) return "Promo code must be at least 3 characters long";
  if (!/^[A-Z0-9]+$/.test(code))
    return "Promo code can only contain uppercase letters and numbers";
  return "";
};

// Generic required field validation
export const validateRequired = (value, fieldName) => {
  if (!value || (typeof value === "string" && value.trim() === "")) {
    return `${fieldName} is required`;
  }
  return "";
};
```

```
// Form validation helper
export const validateForm = (formData, validationRules) => {
  const errors = {};

  Object.keys(validationRules).forEach((field) => {
    const rules = validationRules[field];
    const value = formData[field];

    for (const rule of rules) {
      const error = rule(value);
      if (error) {
        errors[field] = error;
        break; // Stop at first error for this field
      }
    }
  });

  return {
    errors,
    isValid: Object.keys(errors).length === 0,
  };
};

// Common validation rule sets
export const validationRules = {
  login: {
    email: [validateEmail],
    password: [validateRequired],
  },
  register: {
    name: [validateName],
    email: [validateEmail],
    password: [validatePassword],
    phone: [validatePhone],
  },
  menuItem: {
    name: [(value) => validateRequired(value, "Item name")],
    description: [(value) => validateRequired(value, "Description")],
    price: [validatePrice],
    category: [(value) => validateRequired(value, "Category")],
  },
  address: {
    street: [(value) => validateRequired(value, "Street address")],
    city: [(value) => validateRequired(value, "City")],
    state: [(value) => validateRequired(value, "State")],
    pinCode: [validatePinCode],
  },
};
```