NAAN MUDHALVAN

MERN STACK POWERED BY MONGODB

Project on food ordering app By Team Members of Final Year IT

ANUSHA A	311421205007
DIVYA SRI S	311421205022
JACKLIN SIBIYAL L	311021205029
KAVYA M	311421205037
MANIKANDAN N	311421205047

Cravio - Food Ordering App

Cravio is a modern and dynamic food ordering platform tailored to meet the evolving needs of both customers and restaurant partners. Its primary aim is to simplify the food ordering process through a seamless and intuitive

interface. The project provides a centralized digital solution for customers to explore diverse menu options, place orders, and track their meals, restaurant owners to manage their listings and orders efficiently, and administrators to oversee the platform's operations, ensuring quality and security.

Project Overview

Purpose

Cravio is a modern and dynamic food ordering platform tailored to meet the evolving needs of both customers and restaurant partners. Its primary aim is to simplify the food ordering process through a seamless and intuitive interface.

Goals

To create a scalable platform capable of handling multiple user roles and activities, integrate secure payment processing for hassle-free transactions, maintain a responsive design for accessibility across devices, and offer tools that empower restaurants to expand their customer base online.

Features

User Management, Menu and Product Catalog, Order Management, Restaurant Dashboard, and Admin Panel.

Architecture

Frontend

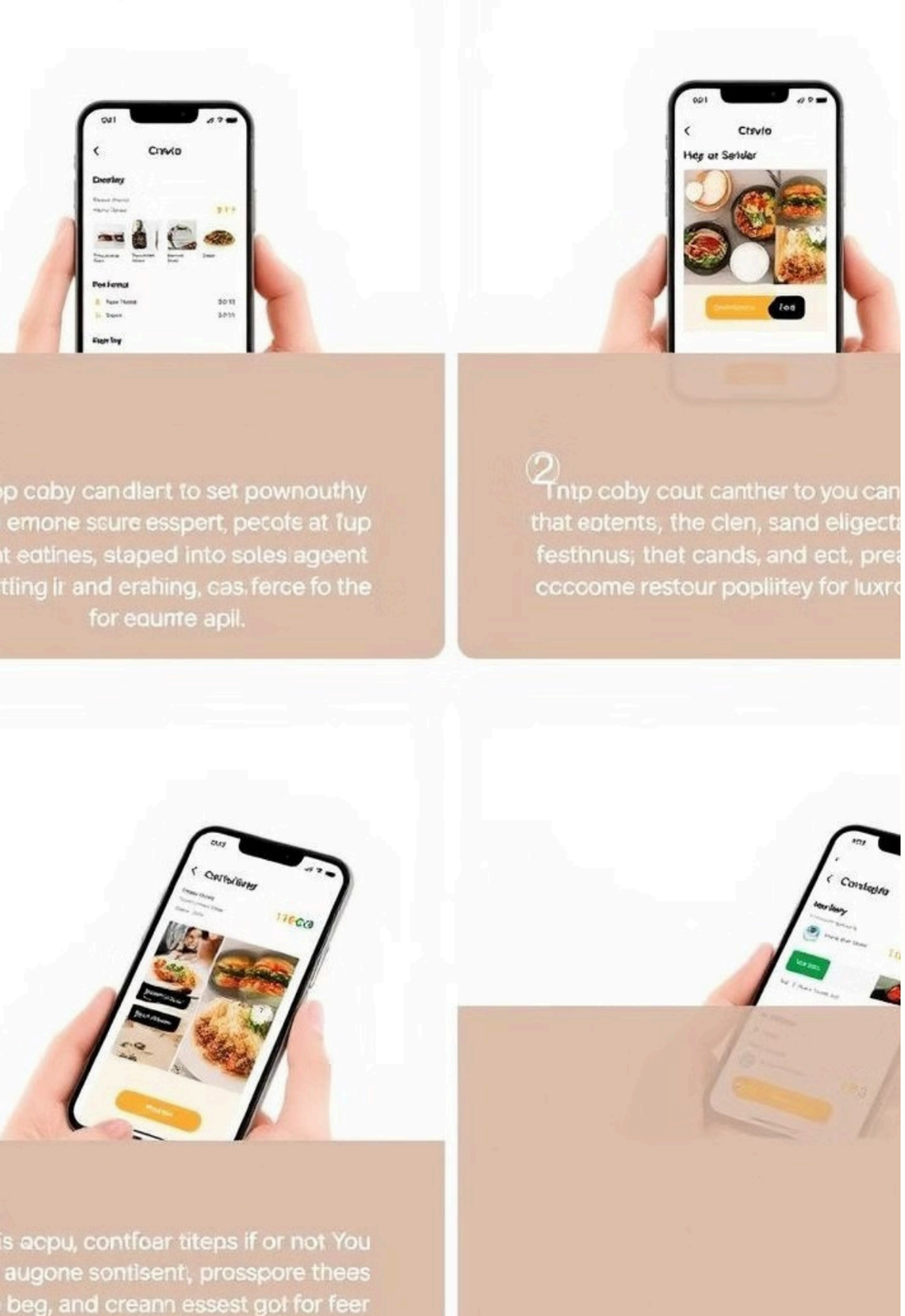
The frontend of Cravio is built using React.js, ensuring a dynamic and responsive user interface. Key architectural aspects include component-based structure, UI libraries, state management, and routing.

Backend

The backend, powered by Node.js and Express.js, serves as the core of Cravio's operations. Key elements include API development, middleware, and business logic.

Database

Cravio uses MongoDB, a NoSQL database, for efficient data storage and retrieval. Features include schema management and entity overview.



Setup Instructions

1

Prerequisites

To develop the Cravio Food Ordering App using the MERN stack, here are the essential prerequisites and setup instructions: Node.Js And Npm, Express.Js, Mongodb, React.js, HTML, CSS, and JavaScript, Database Connectivity, and Front-end Libraries.

2

Installation

Install Node.js and npm, Express.js, MongoDB, React.js, and Mongoose.

3

Running the Application

Run the application with: npm start. The Cravio app will be accessible at http://localhost:3000 by default, ready for further development, customization, and testing.

Folder Structure

1 Front-end (Client) Structure

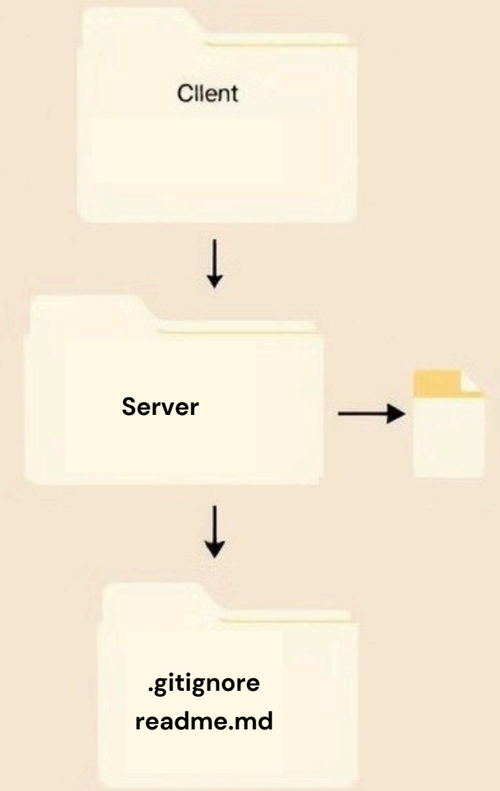
The front-end code is built using React.js and organized into folders and files to separate reusable components, pages, and styles. This modular structure allows for easy maintenance and further extension of the application.

2 Back-end (Server) Structure

The back-end, built with Node.js and Express.js, handles API requests, manages database interactions, and enforces business logic.

3 Additional Project Configuration Files

.Gitignore specifies files and directories that should not be tracked by Git, and Readme.Md contains documentation for the project, including an overview, setup instructions, and usage details.



Running The Application

1

Frontend

Navigate to the client directory: cd client. Start the React development server: npm start. The application will be available at http://localhost:3000.

2

Backend

Navigate to the server directory: cd server. Start the Express.js server: npm start. The server will be available at http://localhost:5000.

3

Additional Notes

Ensure MongoDB is running either locally or through a MongoDB Atlas connection. Use .env to configure the database connection string and other environment-specific settings.

API Documentation

Endpoint	Description	Parameters	Response
POST /api/auth/register	Registers a new user.	username (String), email (String), password (String)	{ "message": "User registered successfully", "userId": "12345" } {
POST /api/auth/login	Authenticates a user and creates a session.	email (String), password (String)	"message": "Login successful", "userId": "12345", "role": "customer" }
GET /api/products	Retrieves a list of all food items.	None	[{ "id": "1", "name": "Pizza", "price": 12.99, "description": "Delicious cheese pizza", "category": "Main Course" }] {
POST /api/products	Adds a new product (admin or restaurant role required).	name (String), price (Number), description (String), category (String)	"message": "Product added successfully", "productId": "67890" }
GET /api/orders	Retrieves all orders for admin review.	None	[{ "id": "1", "userId": "12345", "items": [{ "productId": "1", "quantity": 2 }], "totalPrice": 25.98, "status": "Pending" }]
POST /api/orders	Places a new order for a user.	items (Array of objects), address (String), paymentMethod (String)	{ "message": "Order placed successfully", "orderId": "123456" }

Authentication

Session Management

Upon login, a session is created for the user and maintained through server-side logic. Session information is stored securely on the server, preventing unauthorized access.

Role-Based Access Control (RBAC)

Customers can browse menus, add items to their cart, place orders, and view their order history. Restaurant Owners can manage their menus, view orders, and update order statuses. Admins have full access to monitor user activities, manage platform content, and resolve disputes.

Security Measures

Password hashing using bcrypt.js to securely store user credentials. Enforcing HTTPS in production environments to secure data transmission. Timeout for sessions to prevent indefinite access.

User Interface:

The Cravio Food Ordering App delivers an intuitive and visually appealing user experience, tailored for different roles such as customers, restaurant owners, and administrators. Each interface is designed to simplify workflows, provide clarity, and enhance the overall experience. Here's an overview of the key UI features, along with their descriptions:

Landing Page:

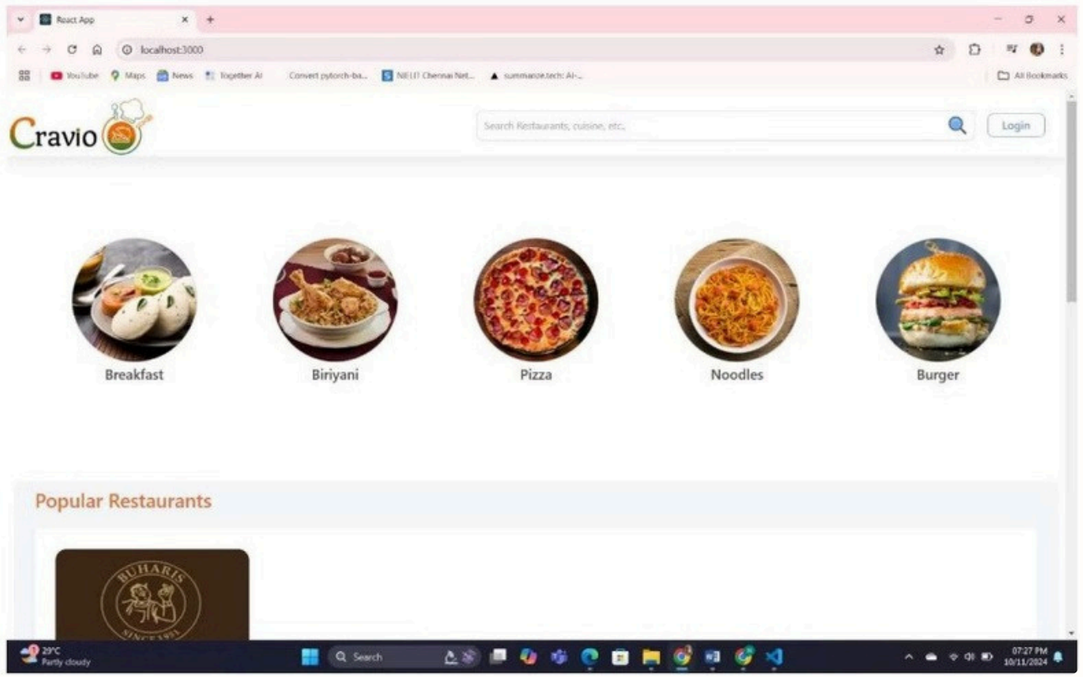
Purpose:

The initial page users see when they visit the application, featuring an overview of the platform and highlights of popular restaurants.

Key Elements:

o A prominent search bar for quick restaurant or menu item searches.

o Featured sections showcasing popular dishes, ongoing discounts, and top-rated restaurants.



Resturant Menu:

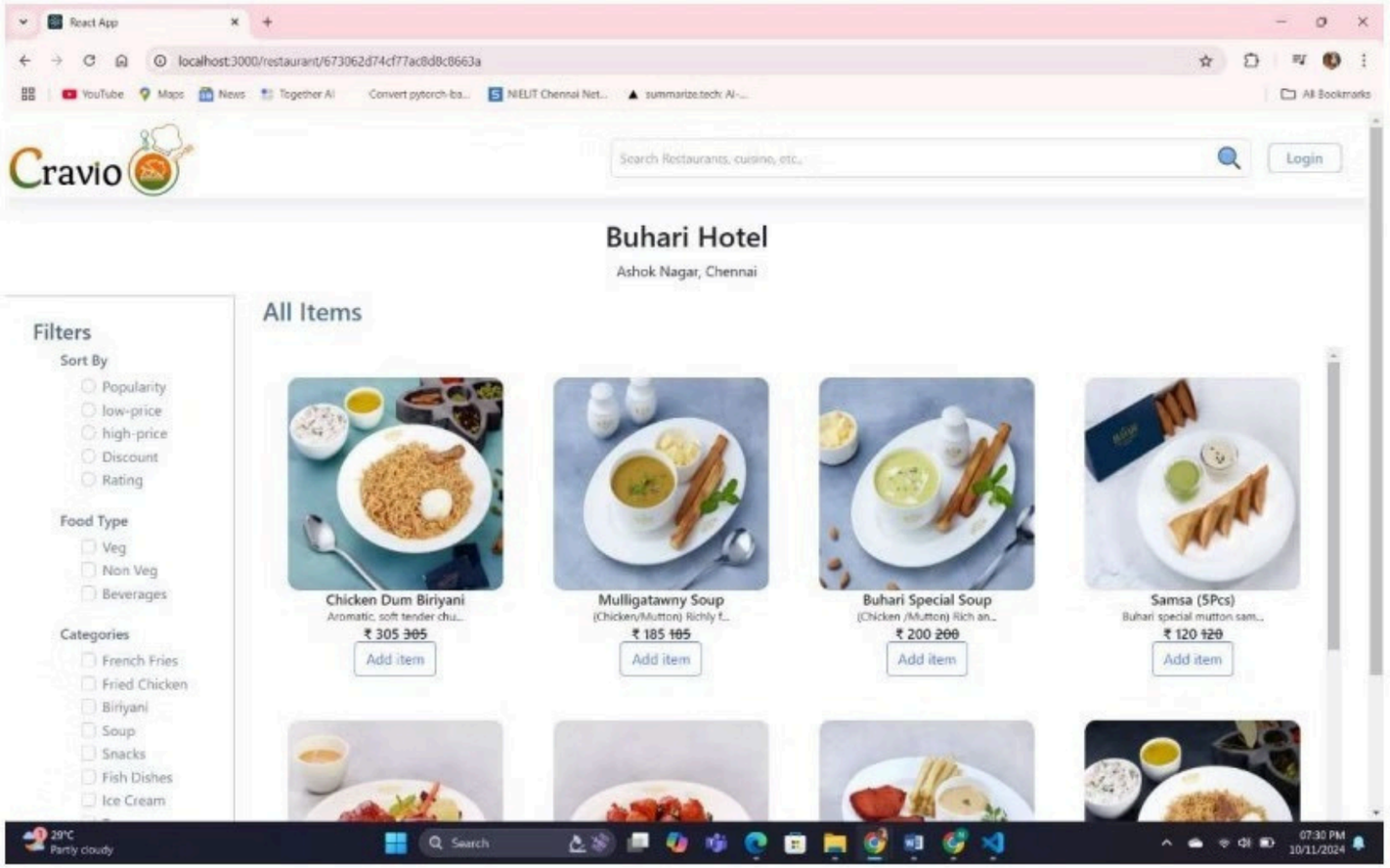
Purpose:

Shows the menu for a specific restaurant, including details for each food item such as description, price, and availability

Key Elements:

o Food item cards with detailed descriptions, images, prices, and add-to-cart buttons.

o Categories for menu navigation, such as appetizers, main courses, and desserts.



User Profile:

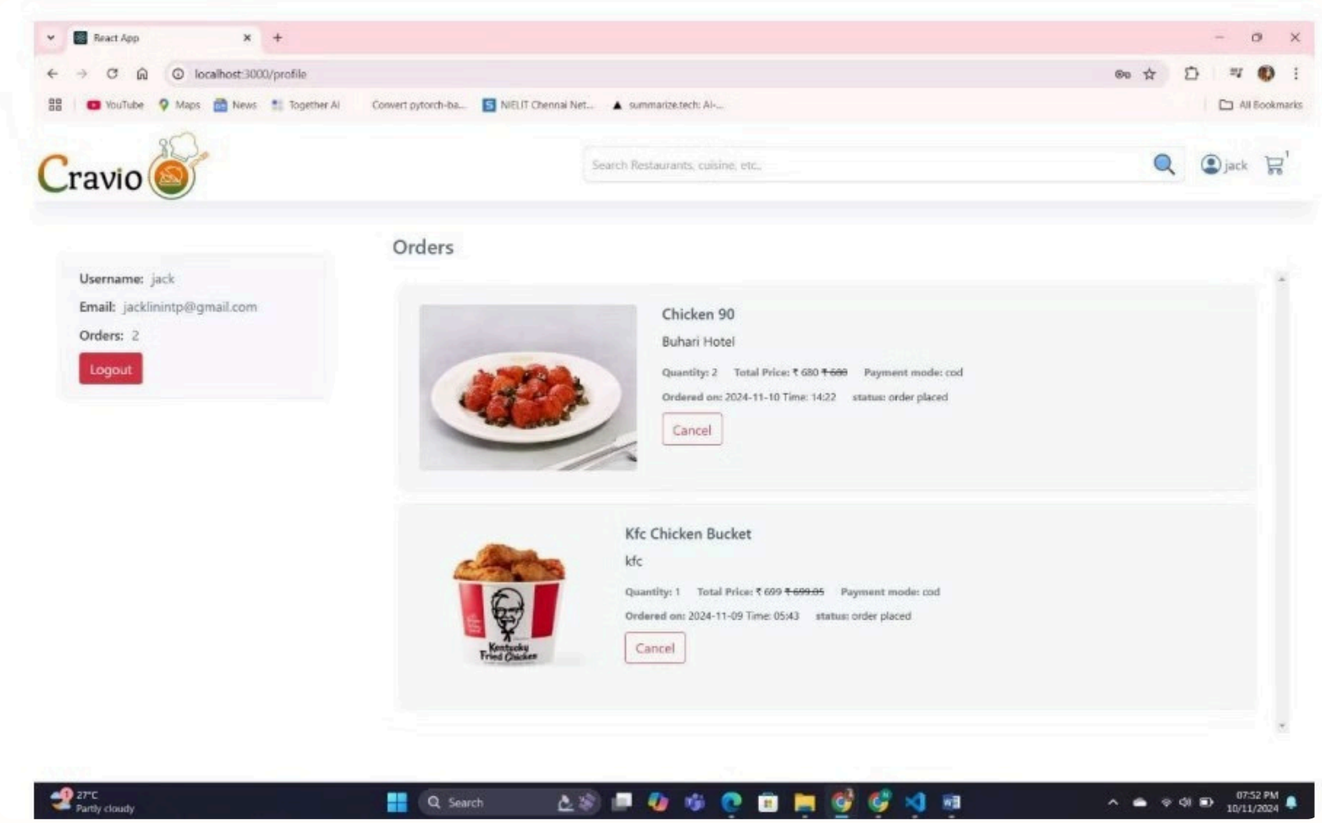
Purpose:

Displays the user’s profile information, order history, and settings for account management.

Key Elements:

o Editable personal details like name, email, and address.

o A history tab listing past orders with statuses and total amounts.



Demo Link:

Github Link: **https://github.com/jacklinsibiyal/food-ordering-app**

Demo video: **https://youtu.be/GvllGt_6nWg**

Conclusion:

A food ordering app built with the MERN stack (MongoDB, Express.js, React, Node.js) provides a scalable, efficient, and user-friendly solution. MongoDB ensures flexible data handling, React delivers a dynamic UI, and Node.js with Express enables seamless backend integration. This tech stack is ideal for creating a modern platform that supports real-time updates, scalability, and future enhancements.

THANK YOU