**What this project is**

* A full-stack cafe ordering app:
* Frontend: React + Vite, routing, auth modal, menu with cart and checkout, orders history, contact form.
* Backend: Express + MongoDB (Mongoose), JWT auth, menu retrieval, order creation/listing, contact submissions.

**How the frontend is structured**

* Entry and app shell
* src/main.jsx: mounts React app and loads global styles (index.css).
* src/App.jsx: sets up routes using react-router-dom, wraps everything in AuthProvider, and renders a Header with links and a Login/Signup button that opens AuthModal.
* Auth state and modal
* src/contexts/AuthContext.jsx:
* Keeps user, token, and loading in state.
* On load, if token exists, calls getCurrentUser to restore session.
* Provides signIn, signUp, signOut, and isAuthenticated.
* src/components/AuthModal.jsx: toggles between Login and Signup forms.
* src/components/Login.jsx and src/components/Signup.jsx:
* Call signIn/signUp from context.
* Show error messages, loading states, and close modal on success.
* API wrapper
* src/lib/api.js:
* Centralized request() that attaches Authorization: Bearer <token> from localStorage.
* Exposes functions: fetchMenu, createOrder, login, signup, getCurrentUser, fetchOrders, submitContactForm.
* Uses VITE\_API\_BASE\_URL or defaults to http://localhost:3001.
* Pages
* src/pages/Home.jsx: simple hero; if logged in, shows a personalized welcome.
* src/pages/Menu.jsx:
* Loads menu from backend (/menu); if backend not configured, shows a fallback list.
* Search and category filter.
* Add-to-cart flow with an options modal for temperature/sweetness/milk (adds small INR add-on costs).
* Cart summary bar with item count and total in INR.
* Checkout modal:
* If API not configured, simulates order with a random ID.
* If API available, calls createOrder with items, options, and totals.
* src/pages/Orders.jsx:
* Requires login: if not authenticated, asks user to log in.
* Loads your orders (/orders) and also loads menu to show item names and compute line totals based on options.
* Displays each order: ID tail, date, items with options and computed price, and total.
* src/pages/Contact.jsx:
* Simple contact form; validates inputs; posts to /contact; shows success/error states.
* src/pages/About.jsx: static content about the cafe.
* Styling
* src/index.css: dark coffee theme variables, background, base elements, responsive tweaks.
* src/App.css: layout styles, header, hero, cards, menu grid, modal, auth, and orders UI.

**How the backend is structured**

* Server bootstrap
* server/index.js:
* Uses express, cors, mongoose, bcryptjs, jsonwebtoken.
* Reads MONGODB\_URI, MONGODB\_DB, MONGODB\_MENU\_COLLECTION, JWT\_SECRET, PORT.
* Connects to MongoDB with SRV URI sanitation helpers for reliability.
* Data models
* MenuItem (menuitems collection or overridden by MONGODB\_MENU\_COLLECTION): id, name, price, category, img.
* Order: array of items with id, quantity, and options (temperature, sweetness, milk); amountInr, currency, userId.
* User: email (unique), password (hashed), name.
* Contact: name, email, message, status (default new).
* Auth
* JWT middleware authenticateToken reads Authorization: Bearer <token>, verifies token with JWT\_SECRET, sets req.user.
* Routes:
* POST /auth/signup: validates, hashes password, creates user, returns token + user.
* POST /auth/login: validates, checks credentials, returns token + user.
* GET /auth/me: protected, returns current user profile.
* Menu and orders
* GET /menu: returns menu items from MongoDB, mapping to { id, name, price, category, img }.
* POST /orders: protected. Validates items, normalizes quantity and allowed option fields, creates order record, returns new order ID.
* GET /orders: protected. Returns orders for current userId (sorted newest first).
* Contact
* POST /contact: stores contact message; returns { success: true, id }.
* GET /contact: protected; returns all contacts (for admin use).
* Admin helper
* GET /users: protected; returns all users without passwords.
* Seeding
* server/seed.js: connects to Mongo and upserts a set of menu items (coffee and bakery) with nice images. Controlled by MONGODB\_MENU\_COLLECTION.

**Step-by-step: typical user flow**

1. User opens the site:

* App.jsx renders header and routes; AuthProvider tries to restore session from localStorage via /auth/me.

1. Browsing menu:

* Menu.jsx fetches /menu (or uses fallback if backend not set).
* User filters/searches, clicks an item to customize options (temperature, sweetness, milk), then adds to cart.

1. Checking out:

* Opens checkout modal, adjusts quantities, sees totals computed in INR (base USD price × 85 + add-ons).
* If logged in, createOrder sends the order to /orders with token.
* If not logged in, header shows Login/Signup; user can authenticate via AuthModal.

1. Authentication:

* Signup.jsx posts to /auth/signup; token is saved and user state set.
* Login.jsx posts to /auth/login; token is saved and user state set.
* AuthContext provides isAuthenticated so the header shows “Logout” and “My Orders”.

1. Viewing orders:

* Orders.jsx calls /orders (needs token) and /menu.
* Shows each order with item names, options, and computed line totals; displays the grand total and date.

1. Contacting the cafe:

* Contact.jsx validates form and posts to /contact; shows confirmation.

**Environment and running**

* Frontend
* npm run dev in project root starts Vite on http://localhost:5173.
* Configure VITE\_API\_BASE\_URL if backend runs elsewhere; otherwise defaults to http://localhost:3001.
* Backend
* In server/.env, set:
* MONGODB\_URI, MONGODB\_DB, MONGODB\_MENU\_COLLECTION, JWT\_SECRET, PORT (defaults to 3001 in code; the guide shows 5174 if you prefer).
* npm run dev in server starts the API.
* Optional: npm run seed to insert sample menu items.

**Security and behavior**

* Passwords hashed with bcrypt, tokens expire in 7 days.
* Protected routes require Authorization: Bearer <token>.
* Orders store only whitelisted option fields and quantities.
* In dev, tokens are stored in localStorage for simplicity.

**High-level summary**

* Frontend:
* React app with pages for Home, Menu, About, Contact, and Orders.
* Auth via modal with context-managed session; orders are visible only when logged in.
* Menu supports search, categories, item customization, cart, and checkout.
* Backend:
* Express API with MongoDB models for users, menu items, orders, and contact messages.
* Endpoints for auth, menu retrieval, order creation/listing, and contact submission.
* JWT-secured operations and a seed script for menu data.

If you want, I can generate a quick “how to run” checklist or a diagram of the data flow.