PHASE 1 — Backend First (Because Frontend needs APIs)

✅ Step 1. Create Backend Project Skeleton

Do first:

1. mern-rbac-store/backend
2. Install packages (express, mongoose, dotenv, jwt, bcrypt, stripe)
3. Create folder structure (models/controllers/routes/middleware)

✅ Output: Server runs with npm run dev

✅ Step 2. Connect MongoDB

Create:

- config/db.js
- Connect in server.js

✅ Output: Console says "Mongo connected ✅"

✅ Step 3. Build RBAC Database Models (Core foundation)

Create Models (in order):

1. **Permission** (permission keys like product:create)
2. **Role** (has list of permissions)
3. **User** (has a role)

✅ Output: DB can store roles + permissions dynamically

Cheat code:
RBAC = "Keys 🔑 → Role bag 🎒 → User wears bag"

✅ Step 4. Seed Super Admin (So you can control everything)

Create a script:

- Insert default permissions

- Insert super_admin role with all permissions
- Insert super admin user account

✅ Output: You can login as Super Admin immediately

✅ Step 5. Auth System (Login + JWT)

Build:

- Register (optional)
- Login
- JWT token generation

✅ Output: You can log in and receive token

✅ Step 6. Auth Middleware (Protect routes)

Create middleware:

- Reads token
- Finds user
- Loads role + permissions into req.user

✅ Output: Protected route works

✅ Step 7. Permission Middleware (Real RBAC check)

Create:

- requirePermission("product:create")

✅ Output: User cannot call API unless allowed

Cheat code:

Permission middleware = “Bouncer checks your pass 🎟️”

PHASE 2 — Ecommerce Core (Products + Orders)

✅ Step 8. Product System

Create:

- Product model
- Product CRUD routes
- Protect create/update/delete using RBAC

✅ Output:

- Admin can manage products (if permitted)
- Customer can only view (if permitted)

✅ Step 9. Order System (Without payment first)

Create:

- Order model
- Create pending order endpoint
- View orders endpoint

✅ Output:

- Orders can exist even before Stripe

PHASE 3 — Stripe Payments (Secure flow)

✅ Step 10. Stripe Checkout API

Flow:

1. Create Order → status **PENDING**
2. Create Stripe Session
3. Return Stripe URL to frontend

✅ Output:

- You get a Stripe checkout URL working

✅ Step 11. Stripe Webhook (Most important security part)

Flow:

1. Stripe calls your webhook after payment
2. Verify signature
3. Update order → **PAID**

✅ Output:

- Order updates automatically after payment

Cheat code:
Webhook = "Bank confirms money came ✅ "

PHASE 4 — Frontend React

✅ Step 12. React Setup + Pages

Build pages in order:

1. Login page (store token)
2. Product list page (public/allowed)
3. Cart page
4. Checkout button (calls backend)
5. Success/Cancel pages

✅ Output: User can browse → pay → see confirmation

✅ Step 13. Admin Dashboard (RBAC Driven UI)

Build pages:

- Manage roles
- Manage permissions
- Assign permissions to roles

- Assign role to user
- Product management
- View orders

✅ Output: Super Admin controls access **from DB** 🔥

PHASE 5 — Testing + Lockdown

✅ **Step 14. Test everything (must-do)**

Test order:

1. Auth login works
2. RBAC blocks unauthorized
3. RBAC allows authorized
4. Stripe checkout session works
5. Webhook updates order to PAID

✅ Output: "Real-world ready"

🗺️ **One-Line Big Picture Flow (mini map)**

DB Models → Super Admin Seed → Auth(JWT) → RBAC middleware →
Products → Orders → Stripe Checkout → Stripe Webhook →
React UI → Admin Panel → Final Testing

⭐ **Best "Start Point" (Where you should start right now)**

✅ Start with: **Backend → DB connect → RBAC models → Seed Super Admin**

Because if RBAC is not ready first, everything else becomes messy.