Great — you're ready to **start coding** your **Online Pharmacy Web App** using **MongoDB** (instead of MySQL). Below is a **complete project roadmap**, tailored for a full-stack web app using:

* **Frontend**: React.js (possibly with Next.js)
* **Backend**: Node.js with Express.js
* **Database**: MongoDB (using Mongoose)
* **Other features**: Secure login, prescription upload, chat, real-time tracking

**✅ Project Roadmap: Online Pharmacy Web App (with MongoDB)**

**🔹 Phase 1: Setup & Planning**

**1. Project Initialization**

* Create your project folder:
* mkdir online-pharmacy && cd online-pharmacy

**2. Initialize Frontend + Backend**

* Use Next.js for the frontend:
* npx create-next-app@latest client
* Initialize backend:
* mkdir server && cd server
* npm init -y
* npm install express mongoose cors dotenv jsonwebtoken bcryptjs multer

**🔹 Phase 2: Database Design with MongoDB**

**1. Define Mongoose Schemas**

Create models for:

* User
* Product
* Prescription
* Order
* Chat

Example: /server/models/User.js

const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({

name: String,

email: { type: String, unique: true },

password: String,

role: { type: String, enum: ['user', 'admin', 'chemist'], default: 'user' }

});

module.exports = mongoose.model('User', UserSchema);

**🔹 Phase 3: Backend API (Node.js + Express)**

**1. Setup server**

// server/index.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

require('dotenv').config();

const app = express();

app.use(cors());

app.use(express.json());

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI).then(() => console.log('MongoDB connected'));

// Routes

app.use('/api/auth', require('./routes/auth'));

app.use('/api/products', require('./routes/products'));

app.use('/api/orders', require('./routes/orders'));

app.listen(5000, () => console.log('Server running on port 5000'));

**🔹 Phase 4: Authentication System**

**1. Auth Routes: /api/auth**

* Register
* Login (JWT)
* Protect routes

Use **bcryptjs** for hashing, **jsonwebtoken** for token handling.

**🔹 Phase 5: Frontend Development (React.js / Next.js)**

**1. Setup Pages & Routing**

* pages/index.js: Landing page
* pages/login.js: Login
* pages/register.js: Register
* pages/shop.js: Browse products
* pages/profile.js: User dashboard
* pages/prescriptions.js: Upload & view prescriptions

**2. Auth Context**

Create global state (with Context API or Redux) to manage user session.

**3. File Upload (Prescription)**

Use **Multer** in backend to handle uploads, with frontend input type="file".

**🔹 Phase 6: Core Features**

**1. Product Catalog**

* Admin can add/edit/delete medications
* Users can view/search/filter products

**2. Prescription Upload**

* Upload scanned prescriptions
* Admin/chemist reviews them

**3. Chat with Chemist**

* Use **Socket.IO** (realtime chat)
* Create ChatRoom model in MongoDB

**4. Order Management**

* User places order → stored in orders collection
* Status updates: "Pending" → "Approved" → "Shipped"

**5. Order Tracking**

* Show order history
* Real-time status updates

**🔹 Phase 7: Security & Validation**

* Secure all APIs with **JWT Auth**
* Validate user input
* Only authenticated users can:
  + Upload prescriptions
  + Place orders
* Only chemists/admins can:
  + Approve prescriptions
  + Update order status

**🔹 Phase 8: Deployment**

**Backend**

* Deploy to **Render**, **Railway**, or **Heroku**

**Frontend**

* Deploy to **Vercel** or **Netlify**

**MongoDB**

* Use **MongoDB Atlas** for cloud database

**🔹 Optional Enhancements**

* 💬 Chat via Socket.IO
* 📱 Mobile responsiveness
* 📦 Add Stripe/Paystack for payment
* 🧠 Add AI feature to pre-validate prescriptions

**🚀 Quick Start Summary**

| **Area** | **Stack** |
| --- | --- |
| Frontend | React.js / Next.js |
| Backend | Node.js, Express.js |
| Database | MongoDB (Mongoose) |
| Auth | JWT, bcryptjs |
| Uploads | Multer |
| Real-time | Socket.IO |
| Deployment | Vercel + Render + MongoDB Atlas |

Would you like me to scaffold the backend or frontend folder for you now? Or generate sample code for a specific module like login or product upload?