Code-Tree, RoadMap & Breakdown

Creating a fully functional delivery system with React Native for the mobile apps and Node.js for the backend is an ambitious but highly achievable goal. This system will consist of multiple components, including user-facing apps (for merchants, customers, and delivery agents) and an administrative panel, all interacting with a robust backend.

Below is a comprehensive **code tree**, **system breakdown**, and **development roadmap** that will guide you through the architecture and implementation of such a system.

📌 SYSTEM OVERVIEW

Stakeholders:

- **Customer**: Places orders, tracks deliveries.
- **Merchant**: Manages inventory, receives orders.
- **Delivery Agent**: Accepts/fulfills delivery tasks.
- **Admin**: Monitors system, manages users/roles/orders.

🗂️ CODE TREE STRUCTURE

delivery-system/

├── backend/ (Node.js + Express + MongoDB/PostgreSQL)

│ ├── config/

│ ├── controllers/

│ ├── middlewares/

│ ├── models/

│ ├── routes/

│ ├── services/

│ ├── utils/

│ ├── tests/

│ └── index.js

│

├── apps/

│ ├── customer-app/ (React Native)

```
│   ├── merchant-app/ (React Native)
│   ├── delivery-agent-app/ (React Native)
│   └── admin-panel/ (Web, React.js or Next.js)
│
├── shared/
│   ├── components/
│   ├── hooks/
│   └── utils/
│
├── .env
├── README.md
└── docker-compose.yml
```

🚧 DEVELOPMENT ROADMAP

✅ Phase 1: Project Planning and Setup

1. **Requirements Gathering**:
    - Features for each user role.
    - Tech stack finalization: React Native, Node.js, MongoDB/PostgreSQL, Firebase (optional), etc.
2. **Environment Setup**:
    - Initialize backend/ with Express + DB.
    - Initialize each app in apps/ with Expo CLI or React Native CLI.
    - Set up monorepo with Lerna or Turborepo for shared utilities/components.
3. **Version Control & CI/CD**:
    - Git with GitHub/Bitbucket.
    - GitHub Actions for CI/CD.
    - Docker for backend and admin panel.

✅ Phase 2: Backend API Development

✳️ Core Modules

- **Authentication** (JWT-based)
    - Signup/login (Merchant, Customer, Agent, Admin)
    - Role-based access control
- **User Profiles**
    - CRUD for all user roles
- **Orders**
    - Place order
    - Assign agent
    - Track order status
- **Inventory (Merchant)**
    - Product CRUD
    - Stock management
- **Payments**
    - Integrate with Stripe, PayPal, or Flutterwave
- **Notifications**
    - Push notifications (Firebase Cloud Messaging)
    - Email/SMS (Nodemailer/Twilio)

✳️ API Structure Example

/api/

├── auth/

│ └── login, register, logout

├── users/

│ └── profile, update

├── merchants/

│ └── products, orders

├── customers/

│ └── orders, addresses

├── agents/

│ └── accept-order, delivery-status

├── admin/

│ └── dashboard, analytics

✅ Phase 3: Frontend App Development (React Native)

👨‍💻 1. Customer App

- Register/Login
- Browse merchants/products
- Place order
- Track orders (map integration)
- Notifications & ratings

👨‍💼 2. Merchant App

- Register/Login
- Manage products/inventory
- View incoming orders
- Dispatch orders

🛵 3. Delivery Agent App

- Register/Login
- View available deliveries
- Accept delivery
- Navigate to pickup/drop
- Update delivery status

👨‍💻 4. Admin Panel (React.js/Next.js)

- User Management
- Order Monitoring
- Reports & Analytics Dashboard

- Flag/disable users

✅ Phase 4: Integration and Communication

- RESTful API integration for all apps
- WebSocket/Socket.IO for live order tracking
- Firebase Cloud Messaging (FCM) for push notifications
- Google Maps API for delivery tracking

🔐 SECURITY & AUTHORIZATION

- HTTPS for all requests
- Input validation with Joi/Yup
- Rate-limiting middleware
- Admin-only routes secured by RBAC
- Data encryption where needed

📈 DATABASE DESIGN (Simplified Example)

Users (id, name, email, password, role)

Products (id, merchant_id, name, price, stock)

Orders (id, customer_id, product_id, agent_id, status, timestamps)

Addresses (id, user_id, street, city, ...)

DeliveryLogs (order_id, timestamp, location)

🧪 TESTING & DEPLOYMENT

- **Backend**: Jest + Supertest
- **Frontend**: React Native Testing Library
- **End-to-End**: Detox/Appium
- **Deployment**:
    - Backend → Render/Heroku/DigitalOcean

- Apps → Google Play Store, Apple App Store
- Admin → Vercel/Netlify

⚙️ OPTIONAL ENHANCEMENTS

- Multilingual support
- Payment escrow system
- Admin revenue dashboard
- Delivery agent geofencing
- AI route optimization (Google Directions API or Mapbox)

💡 RECOMMENDATIONS

- **Use TypeScript** across the board for type safety.
- **Use a design system** (like NativeBase or React Native Paper) to maintain consistency.
- **Modular architecture** for scalability.
- **Error monitoring** with Sentry or LogRocket.
- **Data analytics** for admin dashboards (e.g., Chart.js or D3.js).

Would you like me to generate:

- ERD (Entity-Relationship Diagram)?
- Postman Collection template for APIs?
- Sample UI wireframes/mockups?
- CI/CD workflow for GitHub Actions?

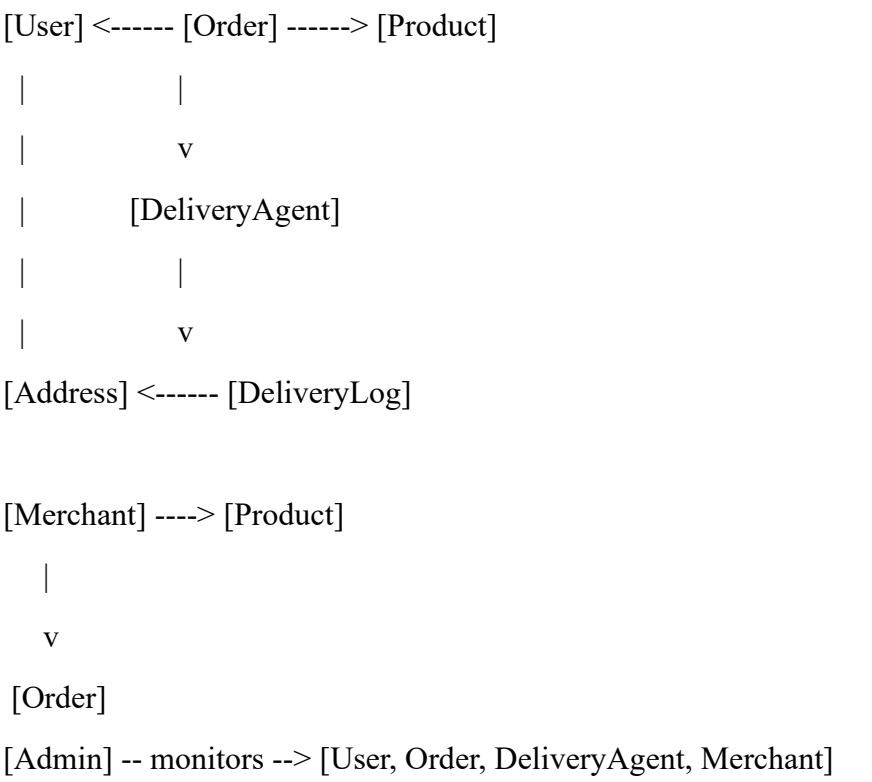
Let me know, and I'll proceed with the next steps.

Action-Tree, PostMan & UI Template

Excellent. Let us tackle each component in detail, ensuring your delivery system project achieves a professional, scalable, and maintainable architecture. I will divide the response into four sections, corresponding to your request.

📊 I. ENTITY-RELATIONSHIP DIAGRAM (ERD)

This diagram reflects the relationships among core entities in your system: customers, merchants, agents, orders, products, addresses, and admins.



Tables and Fields

Users

- id (PK)
- name
- email
- passwordHash
- role (Enum: 'admin', 'customer', 'merchant', 'agent')
- createdAt
- updatedAt

Merchants

- id (PK)
- userId (FK to Users)
- storeName
- storeLocation

Products

- id (PK)
- merchantId (FK)
- name
- description
- price
- stock
- imageUrl

Orders

- id (PK)
- customerId (FK to Users)
- merchantId (FK to Merchants)
- agentId (FK to Users)
- totalAmount
- status (Enum: pending, accepted, in_transit, delivered, cancelled)
- createdAt

OrderItems

- id (PK)
- orderId (FK to Orders)
- productId (FK)
- quantity
- price

DeliveryLogs

- id (PK)
- orderId (FK)
- location
- timestamp

Addresses

- id (PK)
- userId (FK)
- type (home, work, other)
- latitude
- longitude
- addressLine

📪 II. POSTMAN COLLECTION TEMPLATE STRUCTURE

Here's a suggested structure and content layout for a Postman Collection. You can import/export as JSON from the Postman app, but here's a human-readable outline:

🌐 Base URL

http

https://api.yourdeliveryapp.com/api/v1

📁 Folders & Sample Endpoints

🧍 Auth

- POST /auth/register
- POST /auth/login
- POST /auth/logout
- GET /auth/profile

🥷 User

- GET /users/me
- PUT /users/update-profile
- DELETE /users/delete

🛍️ Merchants

- GET /merchants
- POST /merchants/create
- PUT /merchants/update
- GET /merchants/:id/orders
- GET /merchants/:id/products

📦 Products

- POST /products
- GET /products
- PUT /products/:id
- DELETE /products/:id

📜 Orders

- POST /orders
- GET /orders/:id
- PUT /orders/:id/status
- GET /customers/:id/orders

🚚 Delivery Agents

- GET /agents
- POST /agents/accept-order
- PUT /agents/update-status
- GET /agents/:id/orders

🛠️ Admin

- GET /admin/dashboard
- DELETE /admin/user/:id
- GET /admin/orders

I can provide a downloadable .postman_collection.json file upon request.

🎨 III. SAMPLE UI WIREFRAMES/MOCKUPS

Here’s a description of recommended wireframes for each user role. These can be implemented using Figma or any wireframing tool:

1. 📱 Customer App

- **Home**: List of nearby merchants/products
- **Product Details**: Add to cart
- **Checkout**: Payment & delivery address
- **Order Tracking**: Real-time map
- **Profile**: View/update info

2. 📜 Merchant App

- **Dashboard**: Sales, orders overview
- **Inventory**: Add/edit/remove products
- **Orders**: Accept/dispatch view
- **Settings**: Store details

3. 🛵 Delivery Agent App

- **Available Orders**: List + Map view
- **Delivery Task**: Start navigation
- **Update Status**: Pickup → In Transit → Delivered
- **Profile**: Ratings, earnings

4. 🖥️ Admin Panel

- **User Management**: View/ban users
- **Order Monitor**: Live tracking dashboard
- **Analytics**: Revenue, daily orders
- **Reports**: Export CSV, filter

📝 Would you like high-fidelity wireframe images to be generated? If yes, please specify platform (mobile/web) and key screens.

🔄 IV. CI/CD WORKFLOW FOR GITHUB ACTIONS

Here's a generic main.yml for Node.js backend + React Native apps:

📂 **.github/workflows/main.yml**

```
name: CI/CD Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  backend:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: npm install
        working-directory: ./backend

      - name: Run tests
        run: npm test
        working-directory: ./backend
```

```
    - name: Deploy to Render (or Vercel/DigitalOcean)
      run: echo "Add deployment step here..."

  frontend-customer:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Install dependencies
        run: npm install
        working-directory: ./apps/customer-app

      - name: Build
        run: npm run build
        working-directory: ./apps/customer-app
```

You can duplicate similar jobs for merchant-app, delivery-agent-app, and admin-panel.

For Expo-based deployment, consider using:

yaml

CopyEdit

```
- name: Publish to Expo
  run: npx expo upload:android --type apk
```

✅ **NEXT STEPS**

Would you like me to:

1. Generate downloadable ERD diagram (PNG or PDF)?
2. Create and export the actual Postman Collection JSON?

3. Provide real mockups using Figma/Sketch-style visuals?
4. Customize the CI/CD pipeline for Dockerized deployment or Firebase/Expo auto-deploy?

Please specify which output formats you prefer (e.g., diagrams, code files, mockup images), and I will proceed accordingly.