🛵 BiteNow Roadmap — 4 Weeks (28 Days)

Each week is one "sprint". Each day builds on the previous. This is designed so you'll have a working MVP at the end of Week 4.

Week 1 — Auth + Models + Restaurant Management

Goal: Users can register/login with role selection, superadmin seeded, owners can create restaurants & menus.

Day 1

- Setup project structure (backend + frontend folders).
- Initialize Node.js + Express + MongoDB connection.
- Setup dotenv, nodemon.

Day 2

- Create User model (fields: name, email, passwordHash, role, etc.).
- Setup JWT auth (register, login, middleware).
- Seed **superadmin** from env/config.

Day 3

- Build Auth APIs (register/login/logout/refresh).
- Test APIs in Postman.

Day 4

- Setup React frontend.
- Add React Router, Axios, basic folder structure.
- Build Register + Login pages (UI only).

Day 5

- Connect frontend Auth with backend APIs.
- Add role selection at registration (customer, owner, rider).
- Store JWT token in frontend (localStorage).

Day 6

- Build Restaurant model (fields: ownerId, name, address, phone, etc.).
- API endpoints: create, update, list restaurants.
- Protect routes (only owner can create/edit).

Day 7

- Restaurant frontend: owner dashboard → restaurant profile form.
- Connect with backend API.

✅ End of Week 1:

- Register/Login works.
- Superadmin seeded.
- Owners can create restaurant profile.

Week 2 — Customer Ordering & Payments

Goal: Customers browse restaurants, add menu items to cart, checkout with Stripe or COD.

Day 8

- Build Category + MenuItem models.
- API endpoints for owner: add category, add food item.

Day 9

- Restaurant frontend: menu management (owner adds categories/items).
- Customer: restaurant page shows categories + items.

Day 10

- Setup Cart system (frontend state: add/remove items).
- Compute totals client-side.

Day 11

- Create Order model (customerId, restaurantId, items, total, status).
- API: place order, get customer orders.

Day 12

- Setup Stripe integration: backend PaymentIntent endpoint.
- Frontend checkout → Stripe card payment.

Day 13

- Add COD (cash-on-delivery) option in checkout.
- Save paymentStatus in Order model.

Day 14

- Build Customer Order History page.
- Owner dashboard: show incoming orders (pending).

✅ End of Week 2:

- Customers can browse menus, add to cart, checkout.
- Payments integrated (Stripe + COD).
- Orders flow into restaurant dashboards.

Week 3 — Rider Flow + Real-Time Tracking

Goal: Orders can be assigned to riders, riders broadcast location, customers track on map.

Day 15

- Build Rider dashboard basics.
- API: rider sees available deliveries.

Day 16

- Extend Order model with riderId + status updates.
- API: assign rider (owner/admin).

Day 17

- Setup Socket.IO backend.
- Create namespace/room per order.

Day 18

- Rider app: emit location (lat,lng) every 3–5s to socket room.
- Save minimal location history to DB.

Day 19

- Customer tracking page: connect to order room via Socket.IO.
- Display rider location on Leaflet map.

Day 20

- Add order status lifecycle (placed → accepted → ready → picked → delivered).
- Update UI per role (owner marks ready, rider marks delivered).

Day 21

- Test full flow: customer places → owner accepts → rider assigned → rider delivers → customer sees live map.

✅ End of Week 3:

- Rider live tracking working.
- Orders move through full lifecycle.

Week 4 — Chat + Admin + Polish

Goal: Real-time chat per order, admin management, final polish, deploy.

Day 22

- Add ChatMessage model (orderId, senderId, text).
- API: get messages for order.

Day 23

- Socket.IO: order room also handles chat messages.
- Store chats in DB + broadcast in real time.

Day 24

- Build chat UI (customer/owner/rider chat box).
- Integrate with order tracking page + rider dashboard.

Day 25

- Build Admin dashboard.
- Features: view users, block/delete, approve restaurants.

Day 26

- Add basic analytics: daily orders, top items (for owners).
- Admin: system logs.

Day 27

- UI polish with CSS/Bootstrap (responsive grids, cards, navbar).
- Add order status timeline UI for customers.

Day 28

- Final testing: run through each role.
- Deploy backend (Heroku/Render) + frontend (Netlify/Vercel).
- Setup MongoDB Atlas.

✅ End of Week 4 (MVP Complete):

- Auth & roles working.
- Restaurants + menus.
- Orders + payments.
- Rider flow + tracking.
- Chat system.
- Admin dashboard.
- Deployed live.