**1. ER Diagram**

A typical ER model for an on‑demand food ordering app includes these entities:

* **Customer/User** (id, name, email, phone, address)
* **Restaurant** (id, name, address, cuisine, opening\_hours)
* **MenuItem** (id, name, description, price, restaurant\_id)
* **MenuItemExtra** for options/add-ons (id, item\_id, name, extra\_price) [reddit.com+15geeksforgeeks.org+15creately.com+15](https://www.geeksforgeeks.org/how-to-design-er-diagrams-for-online-food-ordering-and-delivery/?utm_source=chatgpt.com)[edrawmax.wondershare.com+5softwareideas.net+5geeksforgeeks.org+5](https://www.softwareideas.net/a/1597/Food-Ordering-System--ER-Diagram-?utm_source=chatgpt.com)
* **Order** (id, customer\_id, restaurant\_id, datetime, status, total\_amount)
* **OrderItem** (order\_id, item\_id, quantity, price\_at\_order)
* **OrderItemExtra** (linking ordered extras)
* **DeliveryPerson** (id, name, contact, vehicle) and assign to Order [geeksforgeeks.org+1gleek.io+1](https://www.geeksforgeeks.org/how-to-design-er-diagrams-for-online-food-ordering-and-delivery/?utm_source=chatgpt.com)
* **Payment** (id, order\_id, amount, method, status) [reddit.com+10edrawmax.wondershare.com+10gleek.io+10](https://edrawmax.wondershare.com/templates/er-diagram-for-food-ordering-system.html?utm_source=chatgpt.com)
* Optionally **Review/Rating** (id, customer\_id, restaurant\_id, menu\_item\_id, rating, comment) [geeksforgeeks.org+3geeksforgeeks.org+3inettutor.com+3](https://www.geeksforgeeks.org/food-delivery-application-project-in-software-development/?utm_source=chatgpt.com)

This supports relationships like one‑to‑many (Customer → Orders, Restaurant → MenuItems) and many‑to‑many via junction tables (Orders → MenuItems + Extras).

For visuals, see templates from Creately or softwareideas.net [reddit.com+13softwareideas.net+13geeksforgeeks.org+13](https://www.softwareideas.net/a/1597/Food-Ordering-System--ER-Diagram-?utm_source=chatgpt.com)[gleek.io+5creately.com+5softwareideas.net+5](https://creately.com/diagram/example/jdybvh1h1/food-ordering-system-er-diagram?utm_source=chatgpt.com).

**2. Pre-Requisites**

You’ll need:

* **Technical stack**: DBMS (MySQL/PostgreSQL), backend (Node, Django, Spring Boot), frontend frameworks, mobile (React Native/Flutter)
* **Dev tools**: Git/GitHub, CI/CD pipelines, container tools for deployment
* **Design assets**: ER diagramming tool, wireframes
* **Accounts**: Payment gateway & push notification services

**3. Application Flow**

1. **User interface**: Login/signup → Browse restaurants → Select items & extras → Place order
2. **Backend processing**: Validate payment → Create Order, OrderItems → Assign DeliveryPerson
3. **Delivery flow**: Update status (Accepted → Preparing → Out-for-delivery → Delivered)
4. **Notifications**: Push alerts for each status change
5. **Post‐order**: Payment confirmation → Ask for reviews

You can enhance with cart persistence, order tracking (maps), promo/coupons, multi-address support.

**4. Project Structure**

**Backend (e.g., Node.js):**

bash

CopyEdit

/src

/models (User, Restaurant, MenuItem, Order…)

/routes (auth, restaurants, orders…)

/controllers

/services (payment, delivery assignment)

/config (DB, env)

/middleware (auth, error handling)

**Frontend (React):**

bash

CopyEdit

/src

/components (Header, MenuList, Cart)

/pages (Home, Restaurant, Checkout, OrderStatus)

/hooks

/services (API calls)

/context (auth, cart state)

**5. Project Flow (Execution Timeline)**

1. **Kickoff**: Gather specs, diagram modeling
2. **Database Design**: Build schema & run migrations
3. **Backend MVP**: Implement CRUD + auth + payment integration
4. **Frontend MVP**: Build user flows + API integration
5. **Delivery logic**: Assign drivers, track status
6. **Notifications**: Push or WebSocket updates
7. **Analytics & Reviews**: Capture order data & feedback
8. **Testing / Deployment**: Unit, integration tests, then deploy

**6. Project Setup & Configuration**

* Clone/init repo with .gitignore, Dockerfile, environment config files (.env)
* Set up your DB and run model migrations
* Configure API keys (payment, push, maps)
* Seed initial data: restaurants, menu items, test users

**7. Database Development**

* Define tables based on ERD
* Enforce FK constraints and indexes
* Use migrations (e.g., Sequelize, Alembic)
* Implement seed scripts and test queries

**8. Backend Development**

* **Auth**: JWT-based login/signup
* **APIs**: Restaurants listing, menu retrieval
* **Cart/order logic**: Calculate totals, store in db
* **Payment**: Integrate with Stripe or Razorpay
* **Delivery assignment**: Simple nearest-driver logic (or queue-based) [reddit.com](https://www.reddit.com/r/FlutterDev/comments/kjezp3?utm_source=chatgpt.com)[inettutor.com+9reddit.com+9reddit.com+9](https://www.reddit.com/r/Backend/comments/15arf8x?utm_source=chatgpt.com)[edrawmax.wondershare.com+7reddit.com+7gleek.io+7](https://www.reddit.com/r/reactjs/comments/fqiesz?utm_source=chatgpt.com)[gleek.io+3reddit.com+3geeksforgeeks.org+3](https://www.reddit.com/r/learnprogramming/comments/1btog1a?utm_source=chatgpt.com)[gleek.io](https://gleek.io/templates/food-delivery-erd?utm_source=chatgpt.com)[reddit.com+2geeksforgeeks.org+2inettutor.com+2](https://www.geeksforgeeks.org/how-to-design-er-diagrams-for-online-food-ordering-and-delivery/?utm_source=chatgpt.com)[reddit.com+1geeksforgeeks.org+1](https://www.reddit.com/r/HuaweiDevelopers/comments/l9yy35?utm_source=chatgpt.com)[edrawmax.com+5reddit.com+5gleek.io+5](https://www.reddit.com/r/businessanalysis/comments/fswu3f?utm_source=chatgpt.com)
* **Notifications**: Push or realtime updates
* Optionally implement webhook handling for payments or driver statuses

**9. Frontend Development**

* **UI components**: Restaurant list, item card, cart panel
* **Forms**: Login/signup, checkout
* **Order tracking**: Show real-time status updates
* **Maps**: Show delivery person location if supported
* **State management**: Context or Redux for cart and auth
* **Mobile considerations**: Use React Native or Flutter

**10. Project Implementation & Execution**

* Iterate in sprints (define, develop, demo, review)
* Use Agile/Git branching (feature, dev, main)
* Monitor performance, errors
* Prototype with testers, refine features
* Deploy backend (Heroku/Docker/Kubernetes), frontend (Netlify/Vercel)