Gym Meal Bowl App: Plan & Learning Path

Step-by-Step Plan to Build the App

1. Core Features
   Client Side (Mobile App / Web App):

- Browse Meal Bowls (vegetarian, protein-based, etc.)
- Nutrition info & macros per bowl
- Customize ingredients (swap paneer for tofu, etc.)
- Add to Cart & Order
- Weekly Subscription / Daily Order Options
- Online Payment Integration (Razorpay, Stripe, etc.)
- Order Status + Delivery Tracking

Admin/Partner Side:

- Menu management
- Order dashboard
- Delivery & logistics control
- Analytics dashboard (sales, most ordered, etc.)

2. Tech Options (Choose One)
   Platform Tools/Tech Stack
   Mobile App (iOS + Android) Flutter or React Native
   Web App React.js (frontend) + Node.js or Django (backend)
   No-Code MVP Glide, Bubble, or Adalo (for fast launch)
3. Example App Flow (User Experience)

- Home Screen – Meal categories (High-Protein, Vegetarian, Low-Carb)
- Meal Details Page – Image, macros, ingredients, allergen info, options
- Customize & Order – Add extras (e.g., ghee, sprouts), select delivery time
- My Orders – Track delivery, re-order, rate meals
- Subscription Option – Weekly plan for gym goers

4. Want to Start Now?
   I can generate:

- A working prototype using Glide/Bubble (no code, fast launch)
- A React Native app starter template with Firebase backend
- APIs and backend code for managing meal plans and orders

What You Need to Learn (In Order)

PHASE 1 — React Frontend (You Already Know)

- Build the UI:
    - Meal bowl browsing
    - Cart & customization
    - Place Order (dummy at first)
- Use mock data or local state to simulate ordering.

PHASE 2 — Firebase Setup & Persistence

1. Firebase Firestore

- Store meals, orders, users in Firestore
- Read/write data from React
- Secure data with Firestore rules

2. Firebase Auth

- Sign in with Google or email/password
- Attach orders to user accounts
- Protect Firestore data per user

PHASE 3 — App Check (Light but Important)

- Enable App Check in Firebase console
- Use reCAPTCHA or SafetyNet on web
- Add fallback logic if check fails

PHASE 4 — Cloud Functions (Core Backend Logic)

1. Trigger Functions

- Run when something happens in Firestore:
    - New order → send notification

    - Admin updates meal → notify user

2. Callable Functions

- Run from your React app:
    - Create Stripe checkout session
    - Trigger manual backend logic

3. Messaging API

- Send push notifications

PHASE 5 — Push Notifications (Android Web Only)

- Get FCM token on target Android device
- Save it in Firestore
- Cloud Function sends message only to that token
- Use firebase/messaging and service worker setup

PHASE 6 — Stripe Integration (No Full Backend Needed Yet)

- Create checkout session from React (using onCall)
- Handle payment success via webhook
- Update Firestore: “User has paid”
- Optionally, connect Stripe subscriptions

Final Cheat Sheet: Learn These in Order

Phase	**What to Learn**	**Why**
1	React + Mock Data	Build UI
2	Firestore CRUD	Store orders, meals
2	Firebase Auth	User sign-in, secure orders
3	Firebase App Check	Prevent abuse, protect API
4	Cloud Functions (onCall, onCreate)	Stripe + notifications
4	Firestore Trigger Functions	Automate backend logic

Phase	What to Learn	Why
4	Admin Messaging API	Send push to 1 user
5	Firebase Messaging	Push notifications to Android
6	Stripe via Firebase Functions	Secure payments

Summary

- Build and demo a working MVP
- Add real users and orders
- Accept payments securely
- Send alerts to one specific phone