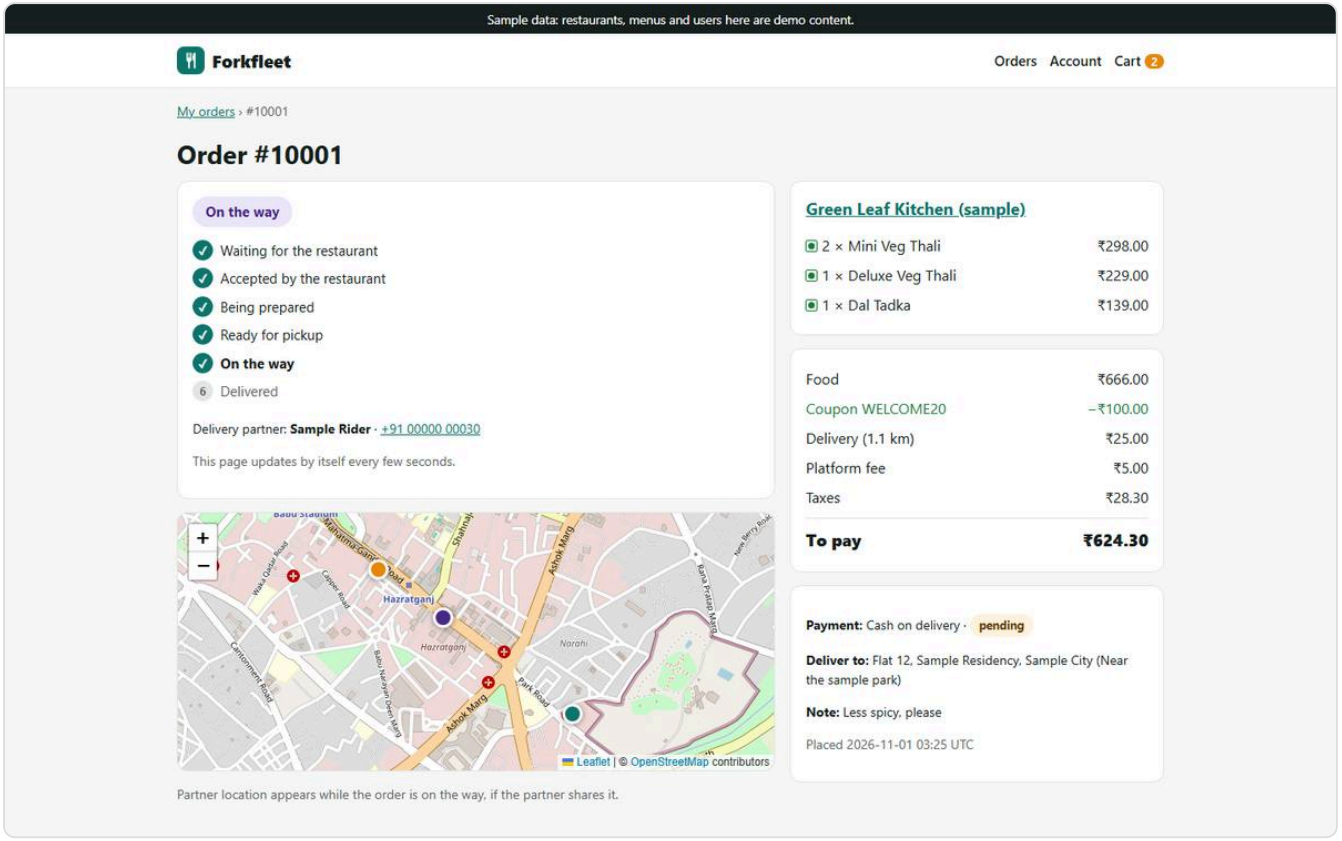
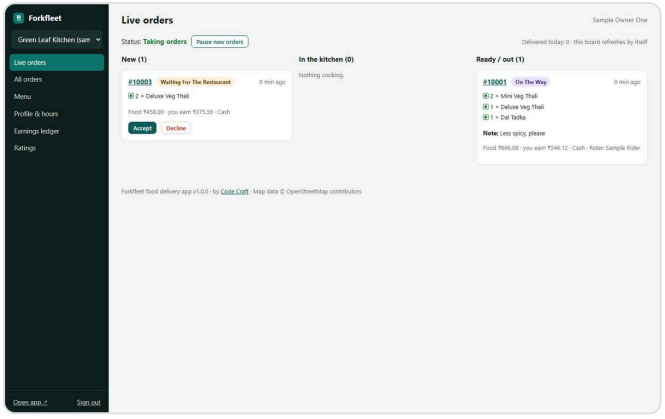
CODE CRAFT · INSTALLATION GUIDE

Food Delivery App

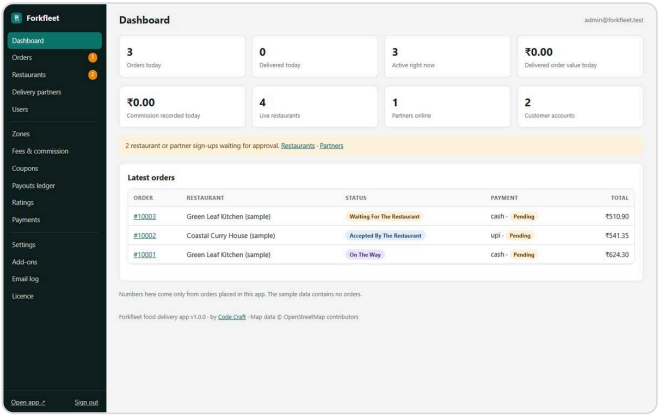
Forkfleet 1.0: a customer web app, a restaurant and delivery-partner panel, an admin panel, a REST API and an Expo mobile app, all in one Node.js project. This guide takes you from download to a live platform on Hostinger, cPanel or a VPS.



A customer's order on the way: status steps, the delivery partner and their position on the map, and the bill calculated by the server.



Restaurant panel: the live orders board.



Admin dashboard.

"Forkfleet", the sample city, restaurants and people are placeholders. Change the name, logo and colours in Admin → Settings.

1. What is in the download

Folder / file	What it is
`server.js`	Start file. Hostinger, cPanel and PM2 run this.
`src/app.js`, `src/routes/`	The Express app: `web.js` (customer web app), `partner.js` (restaurant and delivery-partner panel), `admin.js`, `api.js` (REST API).
`src/lib/`	Pricing, order status machine and ledger, distance and zones, opening hours, cart, UPI links, settings, licence, mailer, add-on loader.
`views/`, `public/`	EJS templates, CSS, small browser scripts and icons. `public/uploads/` holds uploaded photos.
`prisma/`	Database schema and the sample-data script.
`addons/`	Put payment gateway or feature add-ons here. None are included.
`mobile/`	The Expo (React Native) app with a customer mode and a delivery-partner mode.
`test/`	11 unit tests, 16 flow tests and `api-smoke.sh`.
`README.md`, `API.md`	Reference documentation (API.md lists every REST endpoint).

Requirements

- Node.js 18.18 or newer.
- Database: nothing extra for SQLite (default). For MySQL 8 or PostgreSQL 13+, an empty database and a user.
- A domain with https for the live platform.
- For production maps: an account with a map tile provider (see section 6).
- For the mobile app: an Expo account, and Google Play / Apple developer accounts to publish.

2. Run it on your computer

```
npm install
npm run setup      # .env with a random secret, the tables and the sample city
npm start          # http://localhost:3461
```

Address	What opens
`http://localhost:3461`	Customer web app
`/partner`	Restaurant owners and delivery partners (sign up at `/partner/register`)
`/admin`	Admin panel
`/api/v1`	REST API

`npm run setup -- --no-seed` creates an empty app: the first visit to `/admin` then runs a setup wizard that creates your admin account. `npm run setup -- --reseed` wipes everything and loads the sample data again.

Demo logins (sample data, on your own copy only)

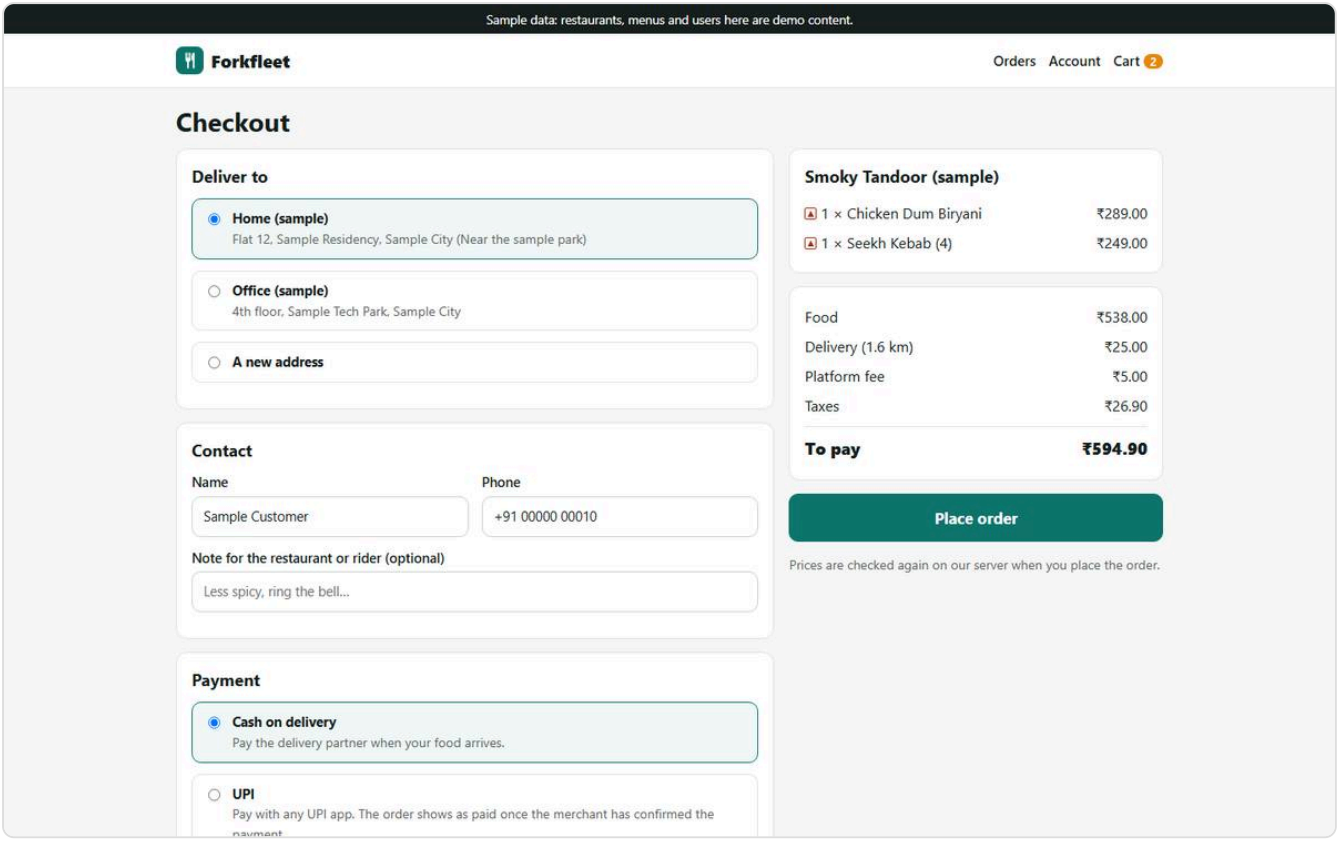
All demo accounts use the password `Demo@12345`. Change or delete them before going live.

Role	Email	Where
Admin	`admin@forkfleet.test`	/admin
Customer (2 saved addresses)	`customer@forkfleet.test`	/ or the mobile app
Customer	`customer2@forkfleet.test`	/ or the mobile app
Restaurant owner (2 restaurants)	`owner1@forkfleet.test`	/partner
Restaurant owner (3 restaurants)	`owner2@forkfleet.test`	/partner
Delivery partner (approved)	`rider@forkfleet.test`	/partner or the mobile app
Delivery partner (waiting for approval)	`rider2@forkfleet.test`	/partner

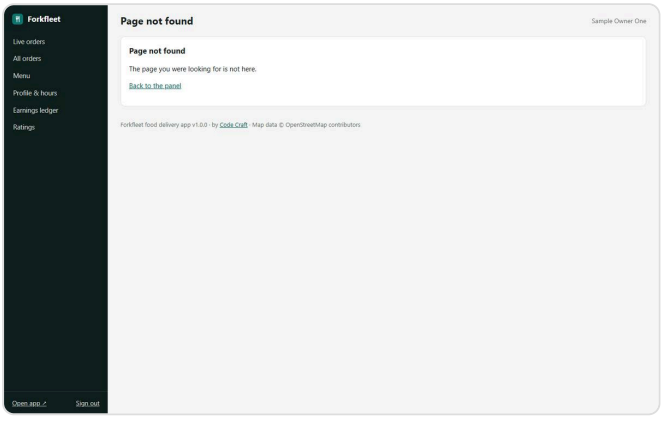
Sample coupons: `WELCOME20` (platform-funded) and `TANDOOR50` (restaurant-funded). Opening hours are real: outside a restaurant's hours it shows "Closed now". To test at any hour, set its hours to 00:00–00:00 in its profile.

3. How an order flows

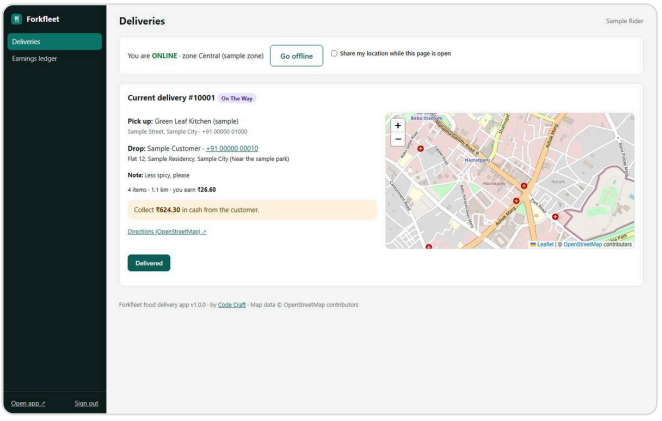
1. The customer fills a cart from one restaurant and checks out with a saved or new address, a coupon and cash or UPI.
2. The server prices everything again (food, coupon, delivery fee by distance, platform fee, tax) and refuses orders out of range, outside hours, below the minimum or with sold-out dishes.
3. The restaurant accepts or declines, then marks the order preparing and ready.
4. An online delivery partner in the same zone accepts it (only one can win), picks it up and marks it delivered. Cash orders become paid at delivery.
5. The ledger records the restaurant's earning, your commission, the partner's pay and any cash collected. The customer can rate the order once.



Checkout: saved addresses, contact, payment method and the bill from the server.



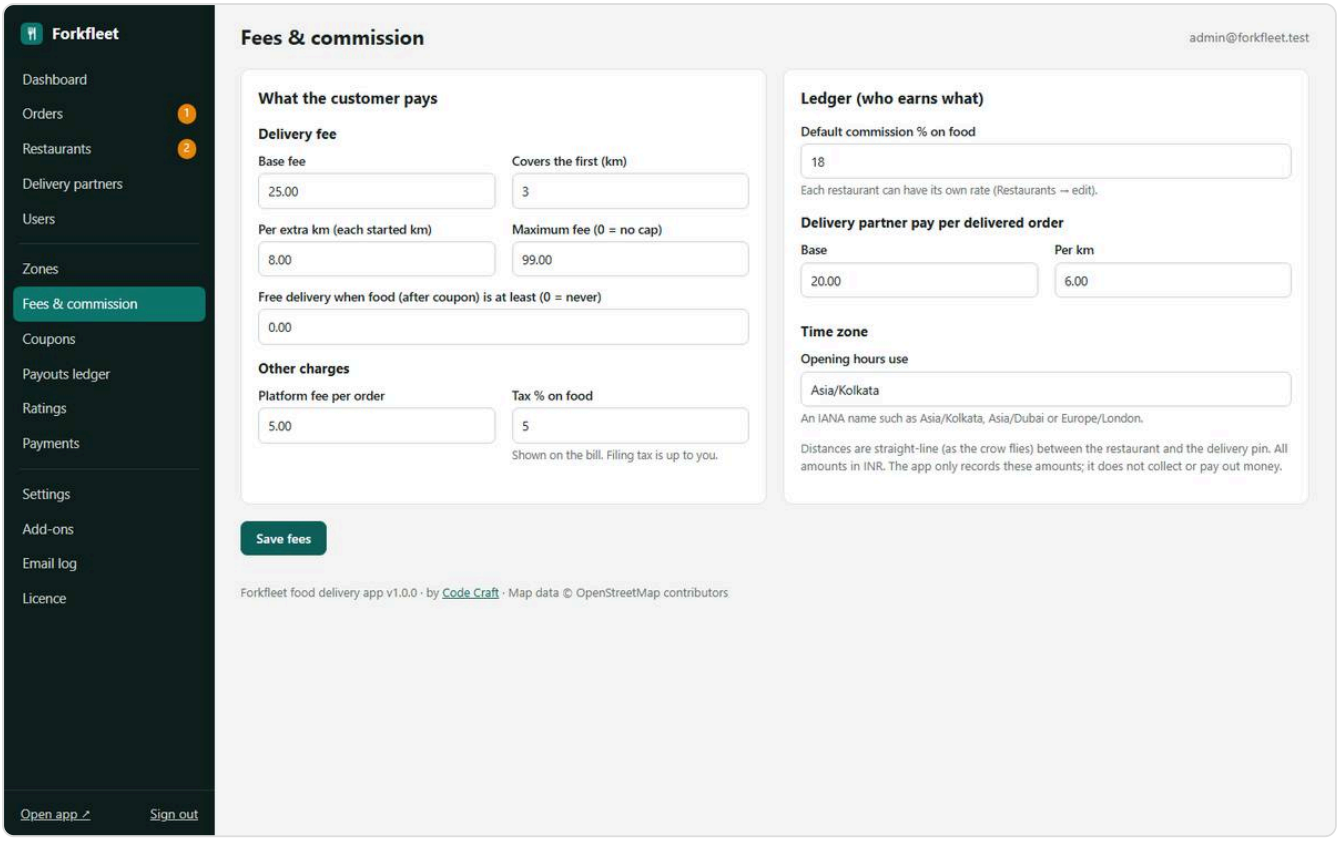
Restaurant menu editor with photos and the sold-out switch.



Delivery partner with a current delivery and cash to collect.

4. First things to set in the admin

1. **Settings → Business & brand:** name, logo, colours, contact details, map centre.
2. **Zones:** your delivery areas, each a centre point and a radius.
3. **Fees & commission:** base delivery fee, kilometres included, price per extra km, cap, free-above amount, platform fee, tax %, default commission %, partner pay (base + per km) and the time zone.
4. **Payments:** your UPI ID (the seeded one is not real) and which methods are on.
5. **Restaurants and Delivery partners:** approve sign-ups or add restaurants yourself.
6. **Settings → Email (SMTP):** until it is set, emails are only written to the email log.



Admin → Fees & commission.

UPI payments: the app builds a `upi://pay` link and QR code for the exact amount to your UPI ID. It cannot see your bank account, so you confirm each UPI payment on the order (Admin → Orders) after checking your bank or UPI app.

Payouts: commission and payouts are a ledger. Pay restaurants and riders outside the app and record each payout in Admin → Payouts ledger.

5. Put it online

Hostinger (Node.js Web App) or cPanel "Setup Node.js App"

1. Upload the folder (without `node_modules`, `.env` or `prisma/*.db`).
2. Start file `server.js`, Node version 18 or newer.
3. Add the environment variables below.
4. Run `npm install` and `npm run setup` from the panel's terminal, then start the app.

VPS with PM2 and Nginx

```
npm ci && npm run setup
pm2 start server.js --name forkfleet && pm2 save
```

Point an Nginx `server` block at `http://127.0.0.1:3461` with `proxy_set_header X-Forwarded-Proto $scheme;`, add https (for example with Certbot) and set `TRUST_PROXY=true`.

Variable	What it is
`DATABASE_URL`	`file:./app.db` (SQLite), `mysql://…` or `postgresql://…`. For MySQL or PostgreSQL set it before `npm run setup`.
`PORT`	Default 3461 (Hostinger and cPanel set it for you).
`APP_SECRET`	A long random string that signs session cookies.
`APP_URL`	Your public https address, used in emails and payment return links.
`TRUST_PROXY`	`true` behind https on Hostinger, cPanel or Nginx.
`LICENSE_KEY`	Your Code Craft licence key (or paste it in Admin → Licence).

Backups: copy `prisma/app.db` (SQLite) or use your host's database backup, and always copy `public/uploads/`.

6. Going live checklist

- Activate the licence in Admin → Licence. Until it is active on a live domain, new orders are switched off. One key covers one business on one live domain at a time. The app checks the key with coodecraft.com once a day and keeps a signed token, so it keeps working offline for at most 14 days. You can move the key to another domain up to 3 times a year (deactivate it on the old domain first). Local preview (`localhost`, `127.0.0.1`, `*.localhost` and `*.test`) needs no key.
- Change or delete every demo account, and replace the sample restaurants, zones and coupons (or start with `--no-seed`).
- Set your real UPI ID in Admin → Payments.
- **Maps:** the app uses OpenStreetMap's public tiles, which are not meant for production traffic. Change `TILE` in `public/js/app.js` to your own tile provider (for example MapTiler, Stadia Maps, Mapbox or Thunderforest) and keep the attribution.
- Set SMTP and send a test email.
- Set `APP_URL` and `TRUST_PROXY=true`, and serve the site over https.
- If you add a payment gateway add-on, test it in the gateway's sandbox before going live.

7. REST API and the mobile app

`/api/v1` uses bearer tokens from `/auth/login` (90 days, stored hashed, revoked on sign-out or when an account is blocked). It covers app info, restaurants and menus, cart quotes, addresses, orders, UPI references, ratings, restaurant order actions and everything a delivery partner does. Prices sent by a client are ignored. See `API.md`.

```
curl http://localhost:3461/api/v1/restaurants?veg=1
curl -X POST http://localhost:3461/api/v1/cart/quote -H "Content-Type: application/json" \
  -d '{"restaurantId":1,"items":[{"itemId":3,"qty":2}],"coupon":"WELCOME20"}'
```

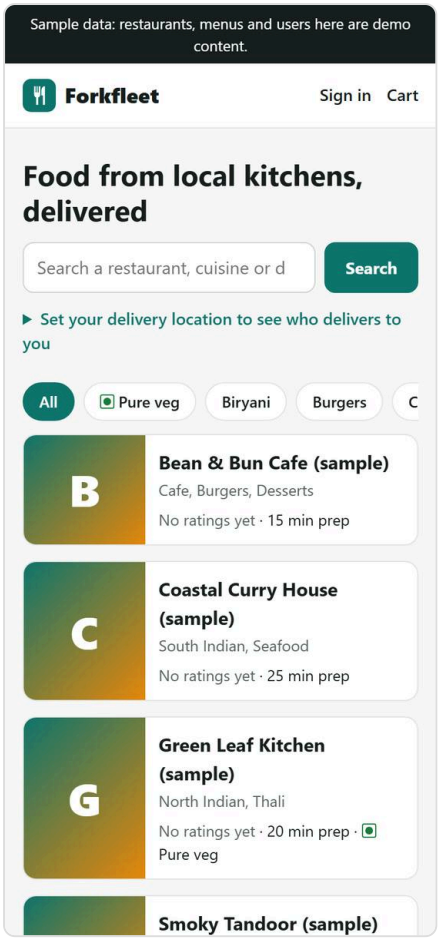
Run the mobile app

```
cd mobile
cp .env.example .env      # set SERVER_URL to your server
npm install
npx expo start            # a (Android), i (iOS) or w (web)
```

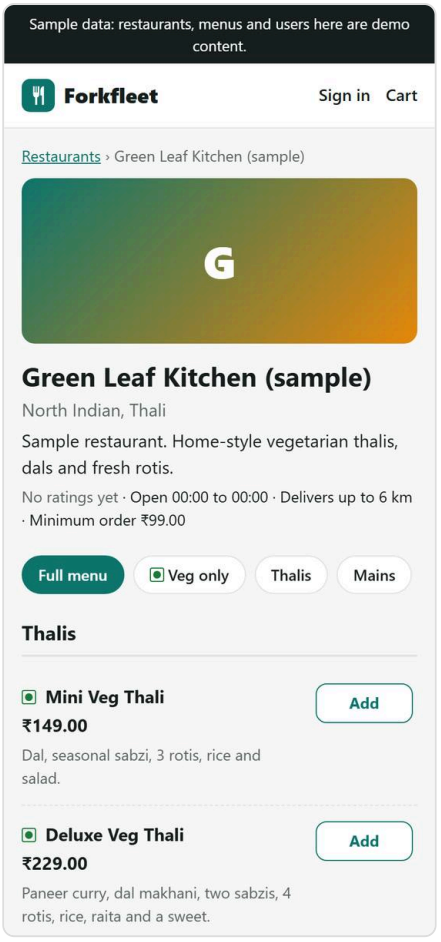
Use `http://10.0.2.2:3461` for the Android emulator, your computer's LAN IP for a phone on the same Wi-Fi, and your https domain for a published app. Set `APP_NAME`, `BUNDLE_ID` and `BRAND_COLOR` in `mobile/.env` and replace the images in `mobile/assets/`.

Build for the stores (EAS)

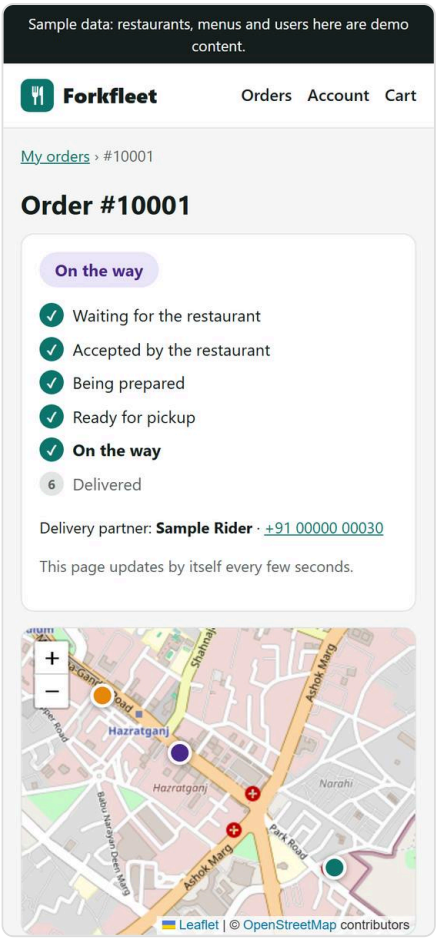
```
npx eas-cli@latest login
npx eas-cli@latest init
npx eas-cli@latest build -p android --profile preview      # installable APK
npx eas-cli@latest build -p android --profile production
npx eas-cli@latest build -p ios --profile production       # needs an Apple developer account
```



Home at phone width.



Menu.



Live order.

8. Honest limits and troubleshooting

- Native Android and iOS builds have not been run by us: the TypeScript check passes and the web build was tried against the server. Test your EAS builds on real phones before publishing.
- No push notifications and no background GPS: the rider's position is sent every 15–20 seconds while their page or app screen is open.
- No payment gateway is included; gateway add-ons in this product family are tested only against mocked responses.
- No wallet or stored balance (in India that needs an RBI PPI licence). Payouts are recorded, not paid, by the app.
- Distances are straight-line, not road distance. Tests ran on SQLite.

Problem	Fix
Restaurants show "Closed now"	Check their hours and the time zone in Admin → Fees & commission.
"Does not deliver to this address"	The address is outside the restaurant's radius or zone; check Admin → Zones and the restaurant's radius.
Ordering is switched off on the live site	Activate the licence in Admin → Licence.
Logged out right away behind https	Set `TRUST_PROXY=true`.
No emails arrive	Check Admin → Email log; set SMTP in Settings.
Mobile app cannot connect	Check `SERVER_URL` in `mobile/.env`; a phone cannot reach `localhost` on your computer.

Support: **coodecraft.com/contact**. Packages with setup include installation on your hosting by us.