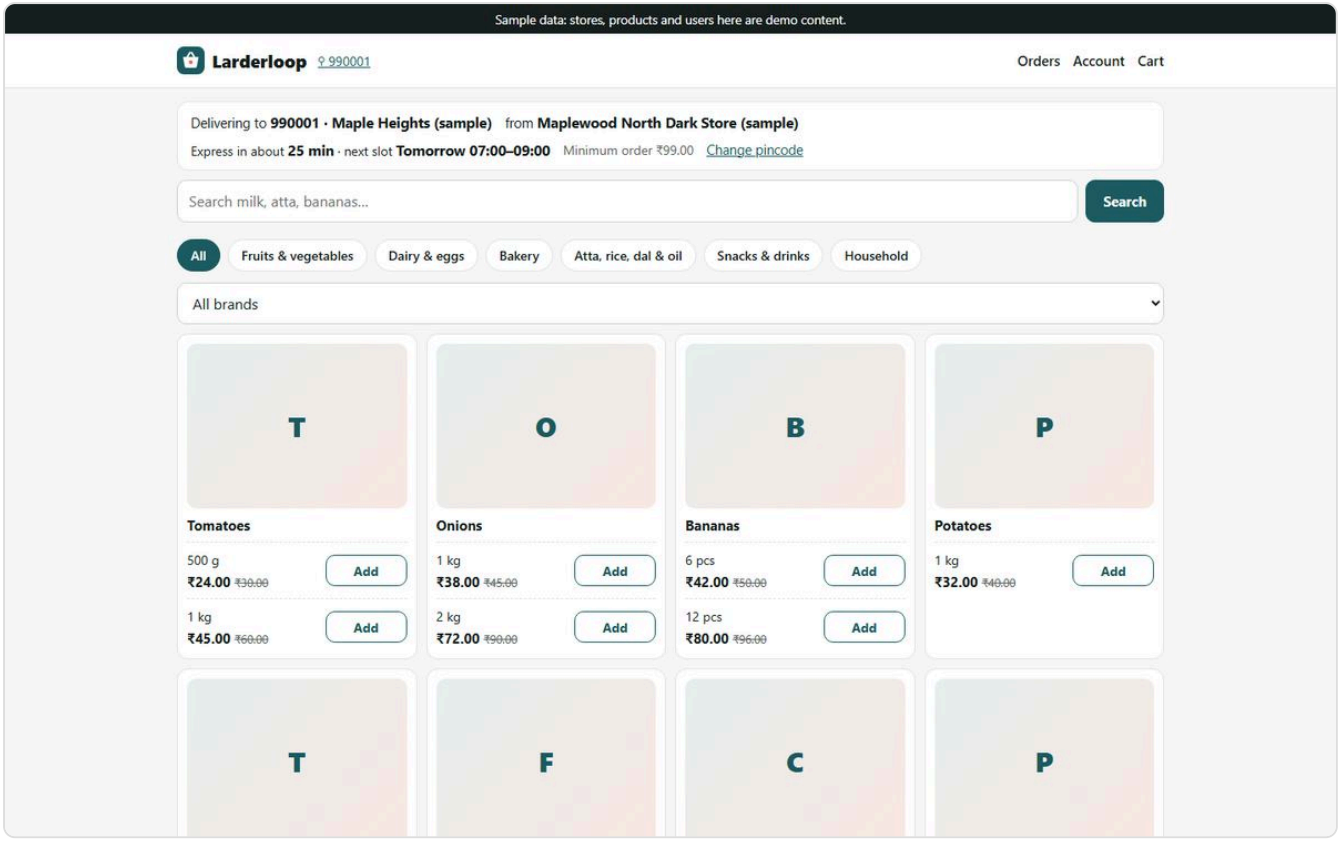
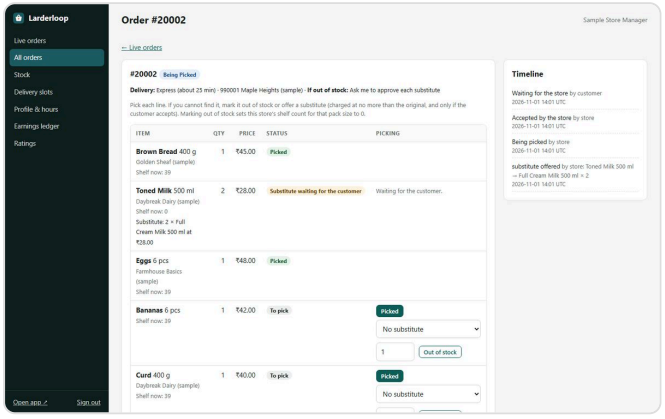
CODE CRAFT · INSTALLATION GUIDE

Grocery Delivery App

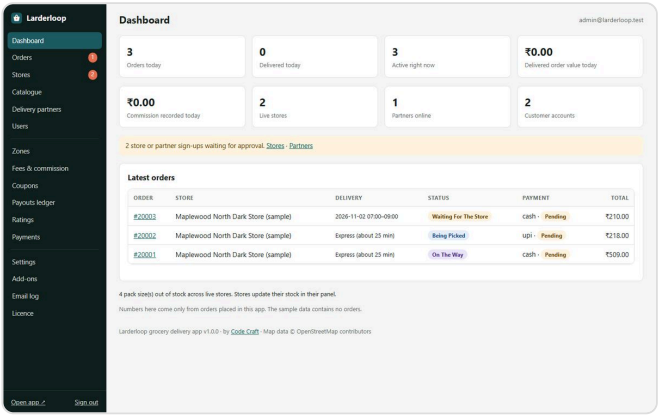
Larderloop 1.0: a customer web app, a store panel for picking and stock, a delivery-partner panel, an admin panel, a REST API and an Expo mobile app, all in one Node.js project. This guide takes you from download to a live platform on Hostinger, cPanel or a VPS.



The customer's shelf: the store chosen by pincode, express time and next slot, categories, brands and pack sizes with price and MRP.



Store panel: picking an order line by line, with a substitute waiting for the customer.



Admin dashboard.

"Larderloop", the sample city (Maplewood), stores, brands, products, pincodes and people are placeholders. Change the name, logo and colours in Admin → Settings. Sample products have no photos: a letter tile is shown until you upload your own.

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` (store and delivery-partner panel), `admin.js`, `api.js` (REST API).
`src/lib/`	Pricing and repricing after picking, the order status machine with stock and ledger, delivery slots, pincode → store, catalogue, cart, distance, opening hours, 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/`	12 unit tests, 15 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. Our tests ran on SQLite only.
- 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 data
npm start          # http://localhost:3520
```

Address	What opens
`http://localhost:3520`	Customer web app (enter a pincode first)
`/partner`	Store staff and delivery partners (sign up at `/partner/register`)
`/admin`	Admin panel
`/api/v1`	REST API

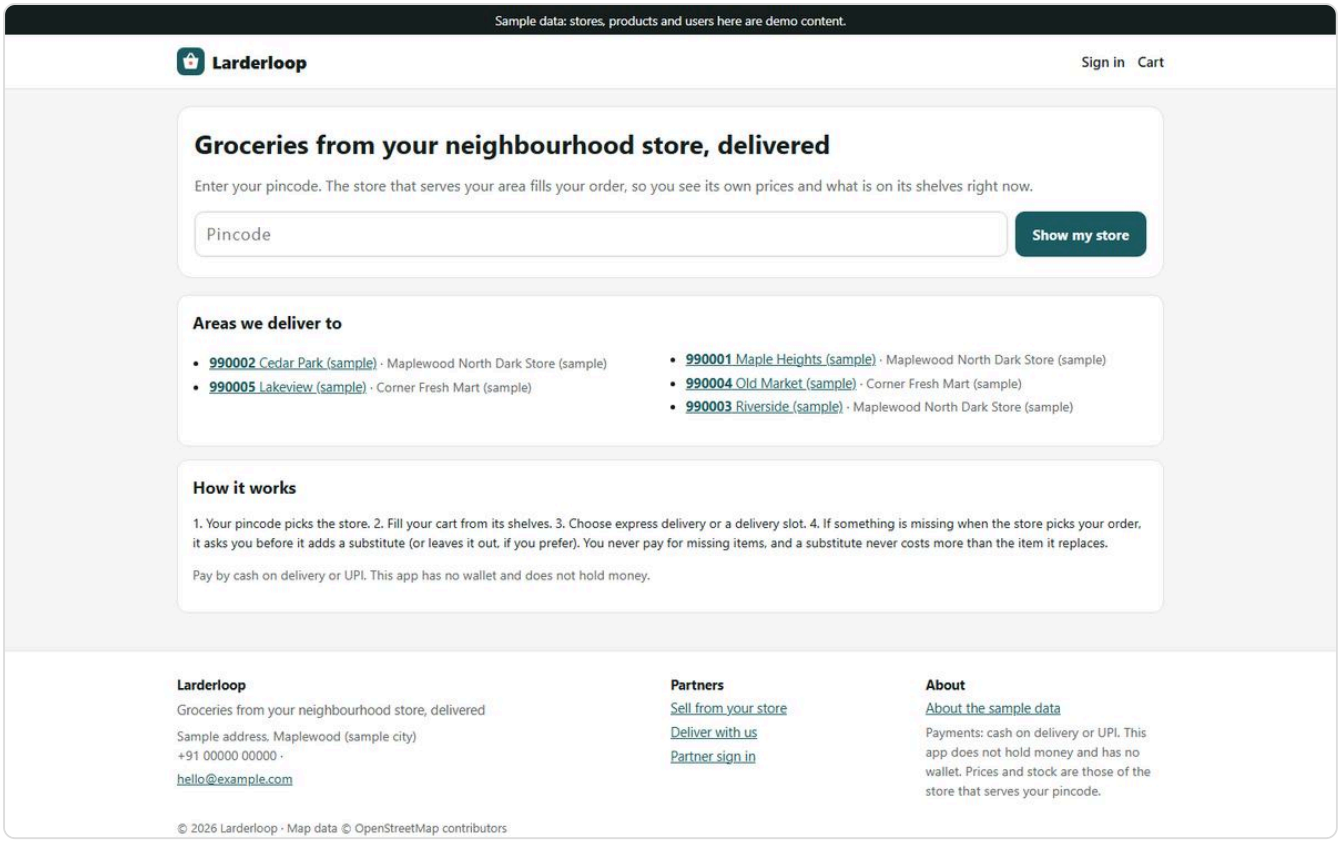
`npm run setup -- --no-seed` creates an empty app: the first visit to `/admin` then runs a setup wizard that creates your admin account. `npm run setup -- --reseed` wipes everything and loads the sample data again.

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@larderloop.test`	/admin
Customer (home in 990001, office in 990004)	`customer@larderloop.test`	/ or the mobile app
Customer (home in 990002)	`customer2@larderloop.test`	/ or the mobile app
Store staff, Maplewood North Dark Store (990001–990003)	`store1@larderloop.test`	/partner
Store staff, Corner Fresh Mart and Greenway Grocers (pending)	`store2@larderloop.test`	/partner
Delivery partner (approved)	`rider@larderloop.test`	/partner or the mobile app
Delivery partner (waiting for approval)	`rider2@larderloop.test`	/partner

Sample coupons: `WELCOME50` (platform-funded, baskets of ₹299+) and `CORNER10` (store-funded). Some stock is set low or at zero on purpose so you can see "only N left" and "out of stock". Opening hours are real: express delivery is only offered while the store is open, and today's slots close 45 minutes before they start. To try express at any hour, set a store's hours to 00:00–00:00.

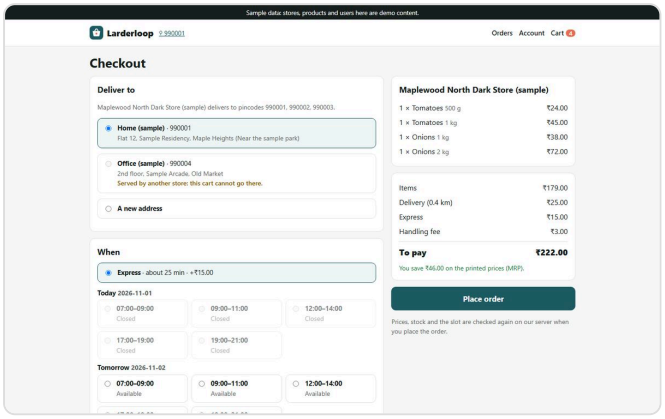


First visit: the customer enters a pincode, and the store that serves it is chosen.

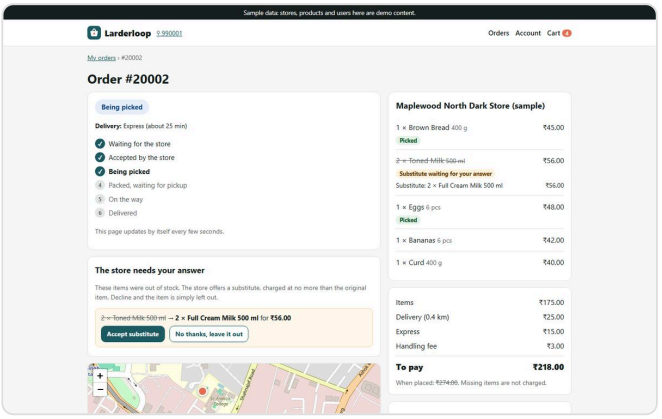
3. How an order flows

1. The customer enters a pincode and shops that store's shelves. At checkout they choose the address (in a pincode that store serves), **express** or a **delivery slot**, and what to do if something is missing: approve each substitute, or no substitutes.
2. The server prices everything again and refuses what cannot be ordered (out of stock, below the minimum, express while closed, a full slot, wrong pincode). In one transaction it **takes the units off the store's stock** and a place in the slot.
3. The store accepts, starts picking and marks each line picked or out of stock. For a missing line it may offer a **substitute**, charged at no more than the original. The customer accepts or declines it; until they answer, the order cannot be packed.
4. When the order is packed, the bill is worked out again from what was packed. Missing items and declined substitutes are not charged, so the total can only go down. UPI is offered for the final amount once packed.
5. An online delivery partner in the zone accepts the delivery (only one can win), picks it up and marks it delivered. Cash orders become paid at delivery, and the ledger records the store's earning, your commission, the partner's pay and any cash collected.

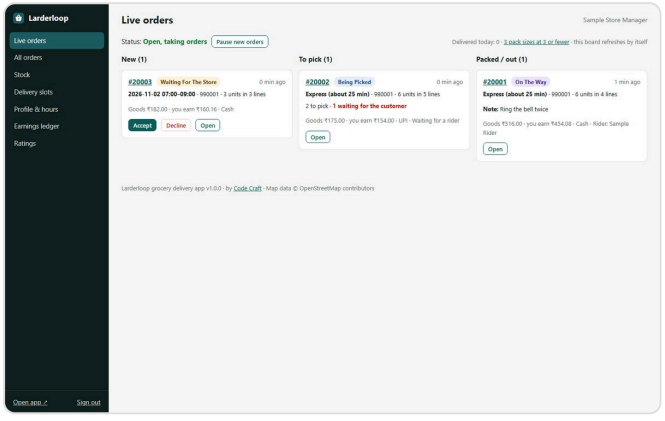
Cancelling (the customer until packed, the store by declining, the admin before delivery) puts the stock back, frees the slot place and returns the coupon use.



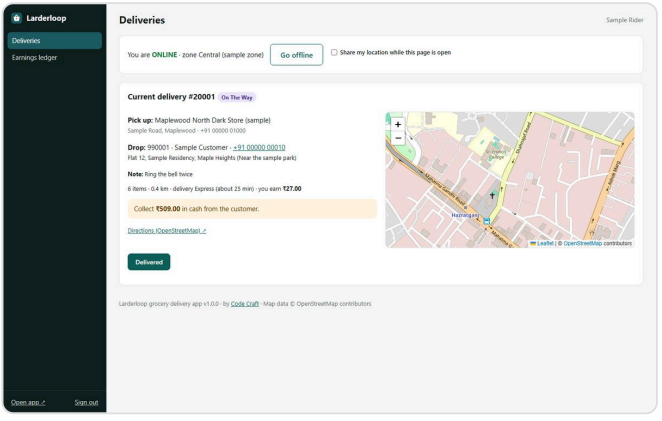
Checkout: address, express or slot, substitution preference and the server's bill.



The customer approves or declines a substitute on the order page.



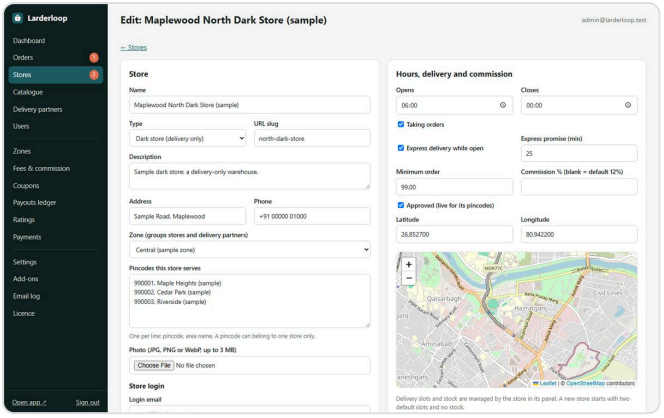
Store panel: live orders board.



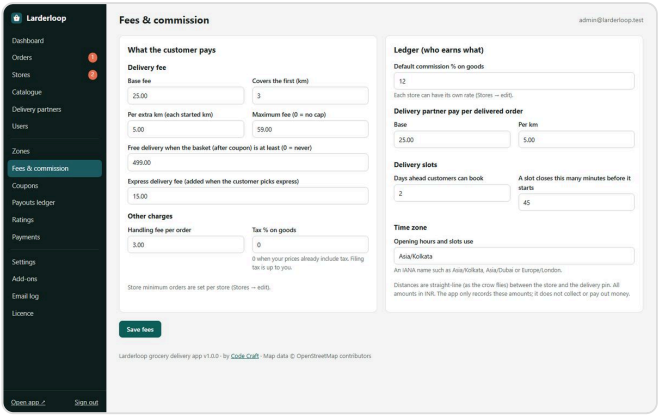
Delivery partner with the current delivery and cash to collect.

4. First things to set in the admin

1. **Settings → Business & brand:** name, logo, colours, contact details, map centre.
2. **Stores:** add your dark stores or shops, their login, hours, minimum order, express on/off and **the pincodes each one serves** (one store per pincode). Approve stores that sign up.
3. **Catalogue:** categories, brands, products and pack sizes with price and MRP, and product photos you own. Each store then sets its own stock in its panel (**Stock**), and its delivery windows in **Delivery slots**.
4. **Fees & commission:** delivery fee (base, km included, per extra km, cap, free above), express fee, handling fee, tax %, default commission, partner pay (base + per km), slot rules (days ahead, cutoff) and the time zone.
5. **Payments:** your UPI ID (the seeded one is not real) and which methods are on.
6. **Delivery partners and Zones:** approve partners; a zone groups stores and partners.
7. **Settings → Email (SMTP):** until it is set, emails are only written to the email log.



Admin → Stores: hours, express, minimum order and the pincodes a store serves.



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 stores and partners 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 larderloop && pm2 save
```

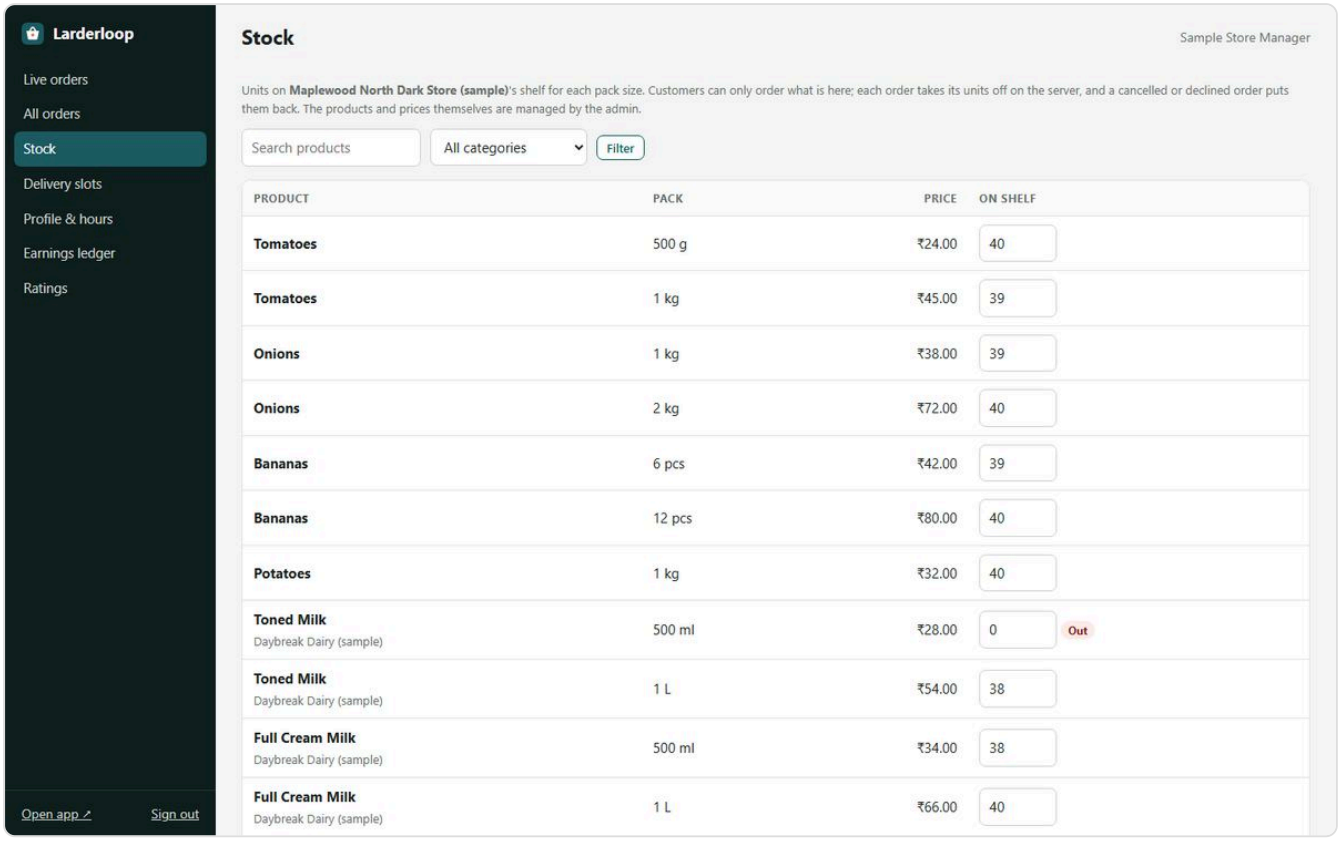
Point an Nginx `server` block at `http://127.0.0.1:3520` with `proxy_set_header X-Forwarded-Proto $scheme;`, add https (for example with Certbot) and set `TRUST_PROXY=true`.

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 3520 (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 stores, pincodes, products, 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 for testing only, not production traffic. Change `TILE` in `public/js/app.js` to your own tile provider and keep the attribution.
- Ask each store to set its real stock counts in Stock: the app takes units off on every order and puts them back on cancel, but it does not know what is physically on the shelf.
- Set SMTP and send a test email. Set `APP_URL` and `TRUST_PROXY=true`, and serve the site over https.
- Selling regulated goods (medicines, alcohol, tobacco) needs licences this app does not check.



Store panel → Stock: units on this store's shelf for each pack size.

7. REST API and the mobile app

`/api/v1` uses bearer tokens from `/auth/login` (90 days, stored hashed). It covers areas and the pincode lookup, each store's products and slots, cart quotes, addresses, orders, substitute decisions, the UPI reference, ratings, the store's order and picking actions, stock, and everything a delivery partner does. Prices sent by a client are ignored. See `API.md`.

```
curl "http://localhost:3520/api/v1/areas/lookup?pincode=990001"
curl http://localhost:3520/api/v1/stores/1/slots
```

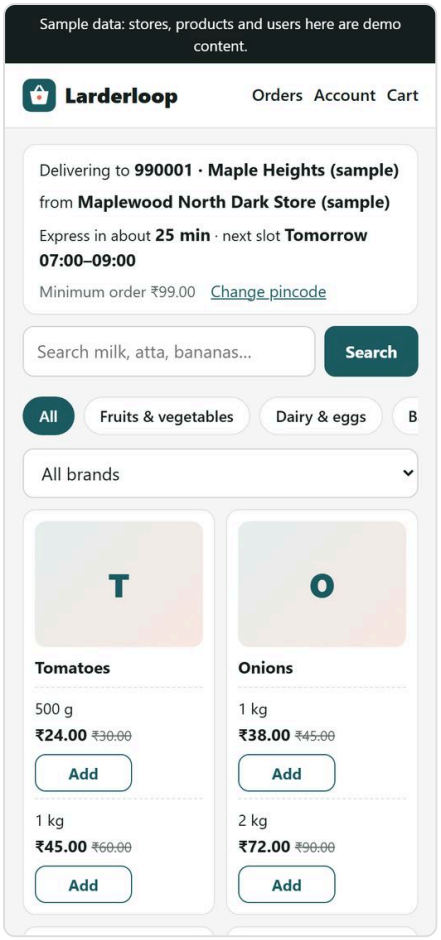
Run the mobile app

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

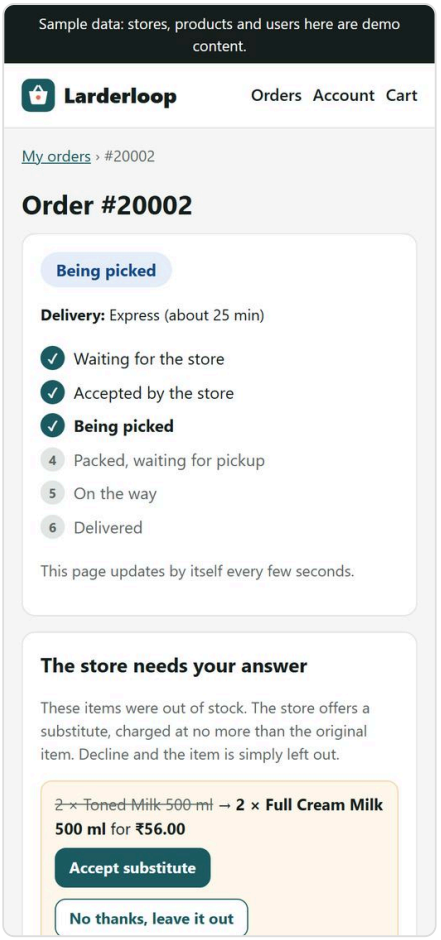
Use `http://10.0.2.2:3520` for the Android emulator, your computer's LAN IP for a phone on the same Wi-Fi, and your https domain for a published app. Set `APP_NAME`, `BUNDLE_ID` and `BRAND_COLOR` in `mobile/.env`. Store staff and admins use the web panels; the app has a customer mode and a delivery-partner mode.

Build for the stores (EAS)

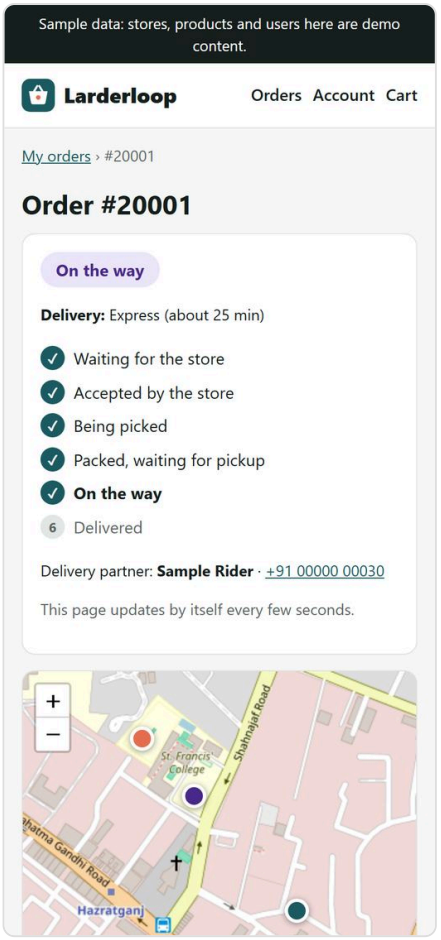
```
npx eas-cli@latest login
npx eas-cli@latest init
npx eas-cli@latest build -p android --profile preview      # installable APK
npx eas-cli@latest build -p ios --profile production       # needs an Apple developer account
```



Shelf at phone width.



Substitute to approve.



Order on the way.

8. Honest limits and troubleshooting

- Native Android and iOS builds have not been run by us: the TypeScript check passes and the web export builds. Test your EAS builds on real phones before publishing.
- No push notifications and no background GPS: the partner's position is sent every 15–20 seconds while their page or app screen is open.
- No payment gateway is included; UPI is confirmed by hand. Gateway add-ons in this product family are tested only against mocked responses. A gateway that charges up front must refund the difference when picking lowers the total.
- No wallet or stored balance. Payouts are recorded, not paid, by the app.
- No barcode scanning, purchase orders or supplier management. Sample products have no photos.
- Distances are straight-line, not road distance. Tests ran on SQLite only.

Problem	Fix
"We do not deliver to this pincode yet"	Add the pincode to a store in Admin → Stores (one store per pincode).
Express is not offered	The store is closed or express is off; check its hours and the time zone in Admin → Fees & commission.
All slots are greyed out	They are full or past today's cutoff; the store adds slots or capacity in Delivery slots.
Ordering is switched off on the live site	Activate the licence in Admin → Licence.
Logged out right away behind https	Set `TRUST_PROXY=true`.
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.