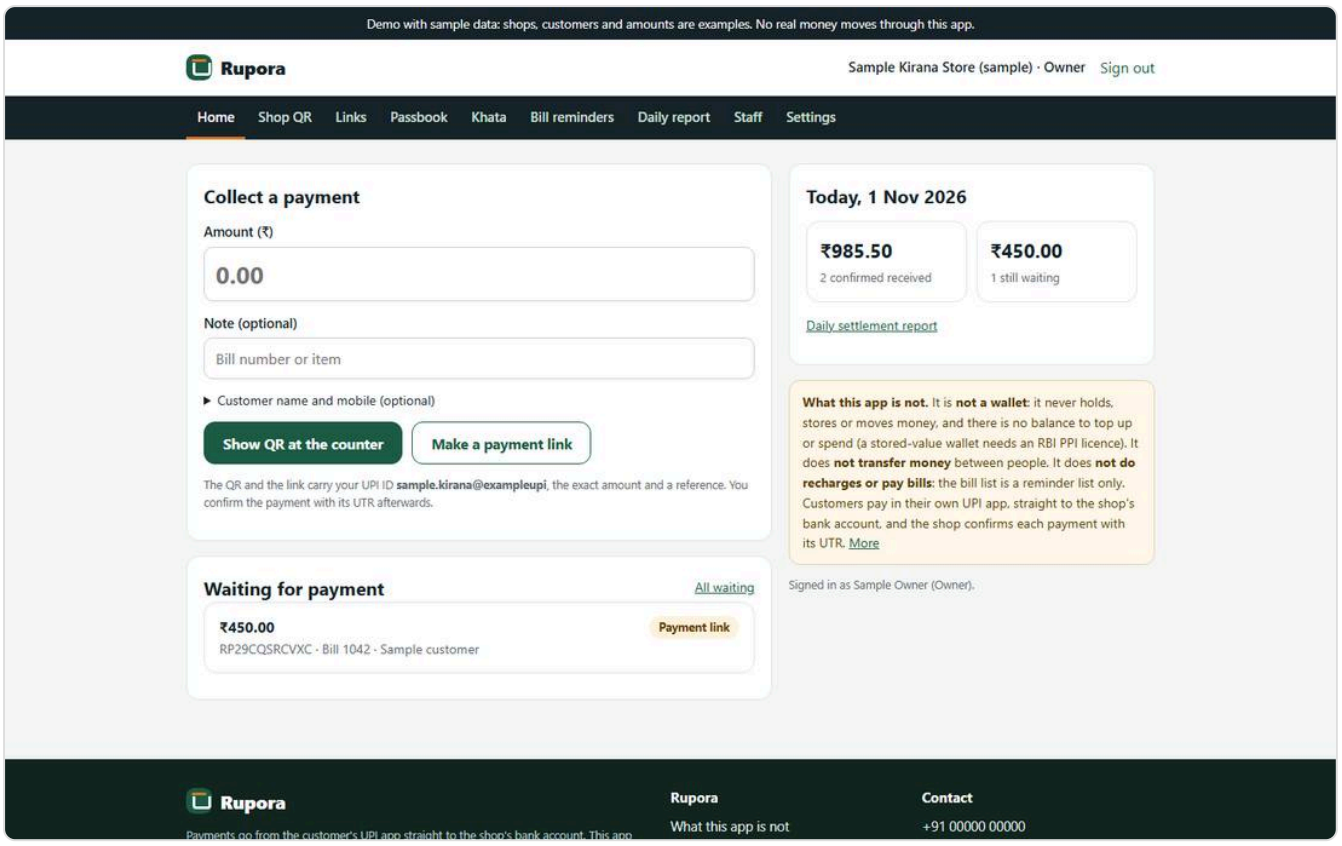
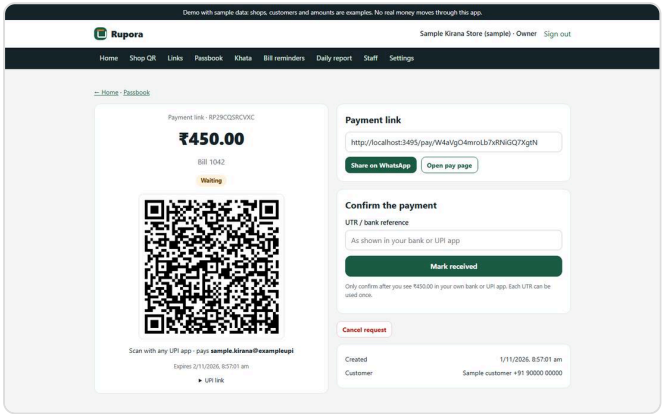
CODE CRAFT · INSTALLATION GUIDE

Merchant Payments App

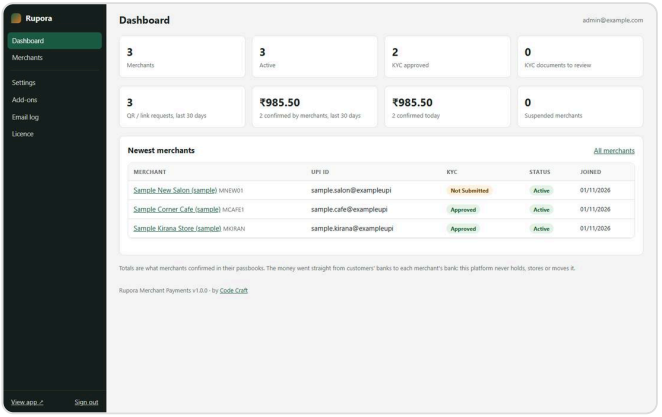
Rupora 1.0: a self-hosted app for shops to collect UPI payments with a QR or a payment link, keep a passbook confirmed by UTR, run a customer credit book (khata) and close the day with a settlement report. Node.js + Express + EJS + Prisma, a REST API and an Expo mobile app.



The merchant panel: type an amount, show a counter QR or make a payment link. Today's confirmed and waiting totals are on the right.



A payment link with its QR, share-on-WhatsApp button and the UTR confirmation form.



Platform admin: merchants, KYC status and totals that merchants confirmed.

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/`	Express app: public site and pay page (`web.js`), merchant panel `/m` (`merchant.js`), platform admin (`admin.js`), REST API v1 (`api.js`).
`src/lib/`	Payment requests and UTR confirmation, UPI links and QR codes, khata, bill reminders, merchants, staff roles, settings, licence, email, add-on loader.
`views/`, `public/`	EJS templates, CSS, JavaScript and images.
`prisma/`	Database schema and the sample data script (`seed.js`).
`addons/`	Drop gateway add-on folders here. None is bundled.
`mobile/`	Expo (React Native) app source for owners, managers and cashiers.
`test/`	Unit, REST API and web-form tests (`npm test`), plus a curl smoke test.
`README.md`, `API.md`, `ADDONS.md`	Reference documentation.

Requirements

- Node.js 18.18 or newer (20 or 22 LTS recommended).
- Database: nothing extra for SQLite (default). For MySQL 8+ or PostgreSQL 13+, an empty database. We tested on SQLite only.
- A domain with https for the live app, and your own UPI ID for each shop.
- For the mobile app: Node.js on your computer, Expo Go or an emulator; an Expo (EAS) account to build store apps.

2. Run it on your computer

```
npm install          # packages + Prisma client
npm run setup        # creates .env (random APP_SECRET), the tables and the sample data
npm start            # http://localhost:3495
```

Page	Address
Public site and sign-up	http://localhost:3495/
Merchant panel	http://localhost:3495/m (merchants sign in at /login)
Platform admin	http://localhost:3495/admin
REST API	http://localhost:3495/api/v1

Other commands: `npm run setup -- --reseed` wipes everything and loads the sample data again; `npm run setup -- --no-seed` gives empty tables and `/admin` runs the setup wizard; `npm run dev` restarts on changes; `npm test` runs the tests.

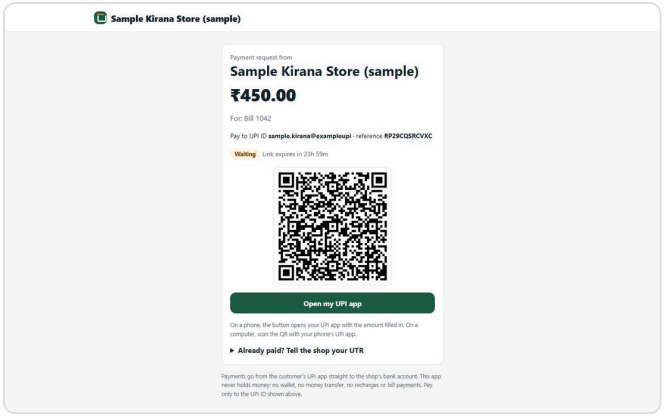
Demo logins (sample data only)

Role	Email	Password
Platform admin (/admin/login)	admin@example.com	`Demo-Admin-2026!`
Shop owner, Sample Kirana Store	kirana.owner@example.com	`Demo-Shop-2026!`
Manager, Sample Kirana Store	kirana.manager@example.com	`Demo-Shop-2026!`
Cashier, Sample Kirana Store	kirana.cashier@example.com	`Demo-Shop-2026!`
Shop owner, Sample Corner Cafe	cafe.owner@example.com	`Demo-Shop-2026!`
Shop owner, Sample New Salon (KYC not submitted)	salon.owner@example.com	`Demo-Shop-2026!`

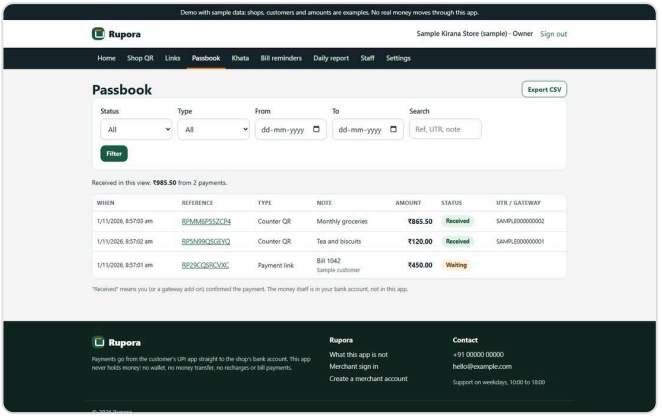
> The shops, staff, khata customers and reminders are made up and marked "(sample)". The UPI IDs end in `@exampleupi`, which is not a real UPI handle: do not try to pay the sample QR codes. No payments are created by the seed. Change or delete these logins before you go live.

3. How a payment works

1. The shop types the amount and taps **Show QR at the counter** or **Make a payment link**.
2. The server builds `upi://pay?pa=<shop UPI ID>&pn=<shop name>&am=<amount>&cu=INR&tn=<note + ref>&tr=<ref>` from the saved UPI ID and the amount, and a QR code of exactly that link.
3. The customer pays in their own UPI app. The money goes straight to the shop's bank account.
4. The shop checks its bank or UPI app and enters the **UTR**. Only now is the request "received". The same UTR cannot be used twice by that shop.
5. Unconfirmed requests expire (counter QR after 30 minutes by default, links as chosen, 1 hour to 30 days). An expired link shows no QR. A late payment can still be confirmed and is flagged "after expiry".



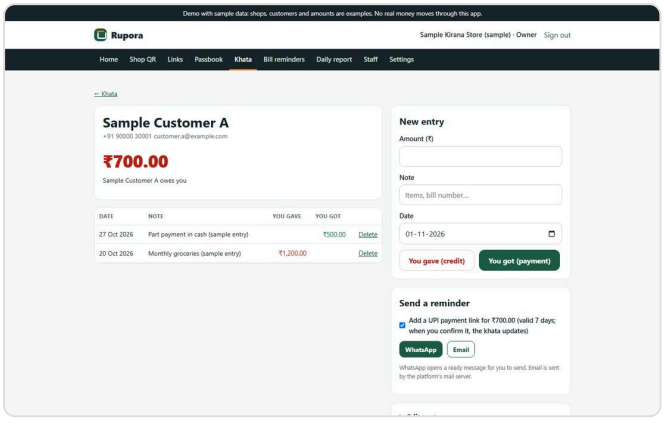
The public pay page the customer opens from a link: amount, QR, Open my UPI app, countdown.



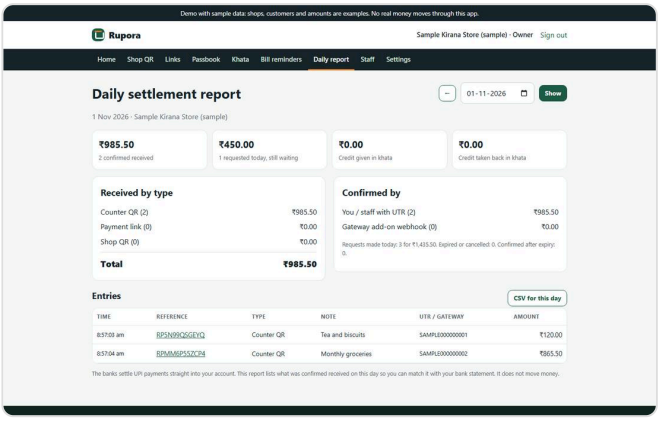
Passbook with filters, status per request, UTR and CSV export.

4. The merchant panel

Screen	What it does
Home	Collect a payment (counter QR or link), today's confirmed and waiting totals, waiting requests.
Shop QR	Static shop QR without an amount, a print view, and "record a shop-QR payment" with the UTR.
Links / Passbook	Every request with status (waiting, received, expired, cancelled), filters and CSV.
Khata	Customers, "you gave" / "you got" entries, balances, WhatsApp share link and email reminders, optional 7-day payment link for the balance.
Bill reminders	Due dates for electricity, recharges, rent, EMI and so on, with a monthly repeat and an email a few days before. Reminders only: the app pays nothing.
Daily report	Received today by type and by who confirmed, still waiting, expired, khata given and taken back, CSV for the day.
Staff	Managers and cashiers with limited permissions. Disabling a login signs it out of the app.
Settings	Shop name, UPI ID, KYC document upload.



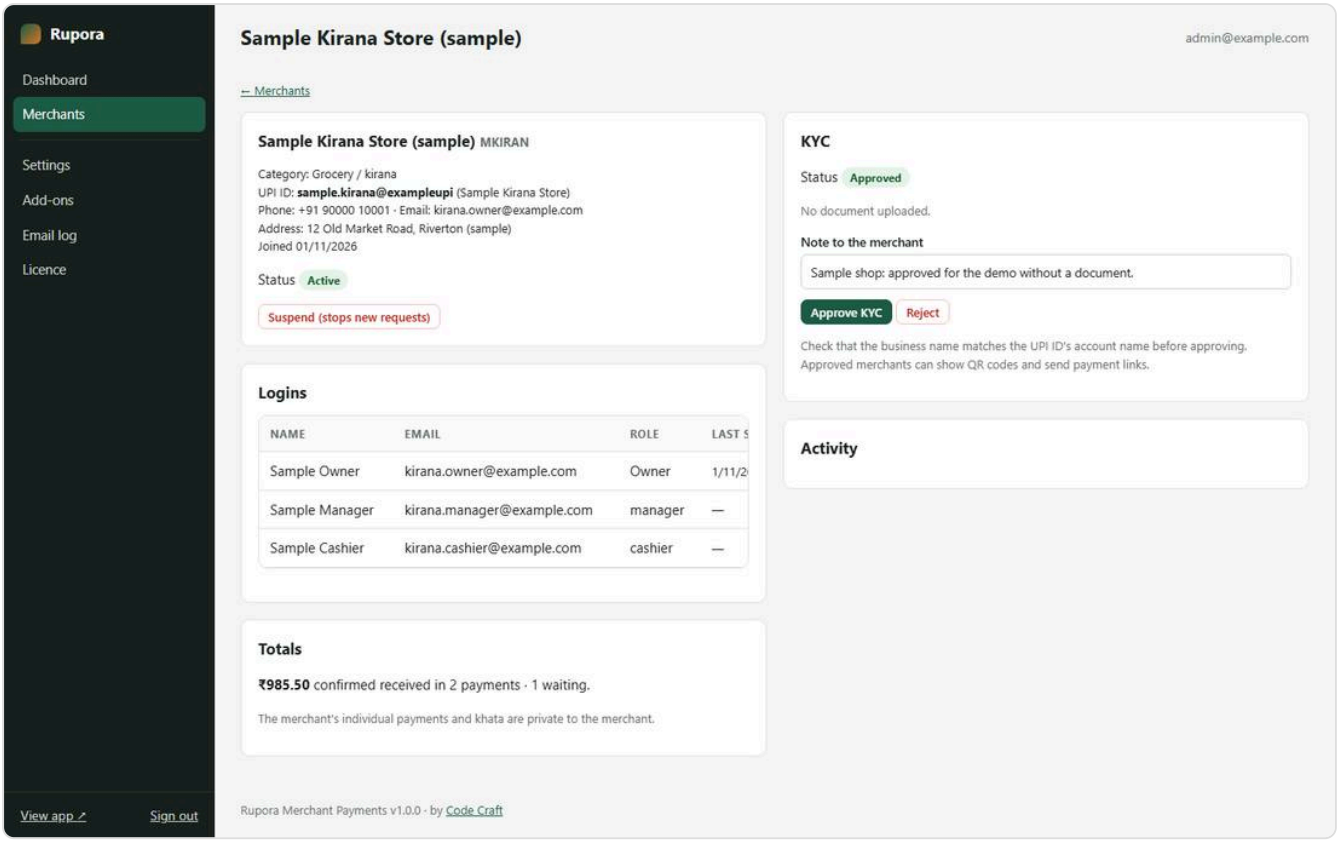
A khata customer: balance, entries, and a reminder by WhatsApp or email.



Daily settlement report: a record of what was confirmed, not a payout.

5. Platform admin

The admin at `/admin` runs many merchants: dashboard, merchant list with search and CSV, a merchant page with the KYC document viewer, approve or reject with a note and email, and suspend. Settings cover business details, brand colours, SEO, maintenance mode, collection rules (minimum and maximum per request, default and longest link expiry, counter QR lifetime, KYC required to collect, open sign-ups, bill reminder days), live chat, WhatsApp button, analytics, banners and SMTP. Emails that cannot be sent yet are kept in the email log. Add-ons and Licence have their own screens.



Merchant page in the admin: details, logins, KYC approval and totals.

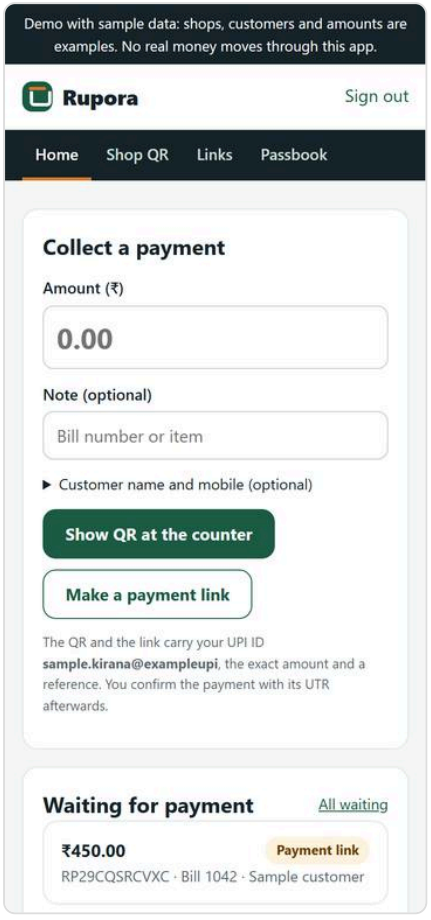
6. REST API and mobile app

The REST API at `/api/v1` (bearer tokens) covers sign-in and sign-up, the shop, collecting, the passbook and UTR confirmation, khata, bill reminders, reports, staff, the public pay page, platform admin and gateway add-on webhooks. `API.md` documents every endpoint. `GET /api/v1` returns the "what this app is not" text.

The `mobile/` folder holds an Expo SDK 58 + Expo Router app in TypeScript: sign in, Collect, the request screen with QR, share link and "Mark received" with the UTR, Passbook, Khata with reminders, and Shop (today's totals and the static QR). Tabs follow the login's permissions.

```
cd mobile
cp .env.example .env      # set APP_SERVER_URL to your server
npm install
npx expo start            # Expo Go, emulator or web
npx eas-cli build -p android   # store build, after eas init
```

> Native Android and iOS builds have not been run by us. The app type-checks and its web export was tested against the server in a browser.



The merchant panel at phone width, signed in as a cashier.

7. Put it online

1. Upload the files without `node_modules`, `.env` and `*.db` (Hostinger Node.js Web App, cPanel "Setup Node.js App", or a VPS with PM2 behind Nginx).
2. Node 20 or 22, startup file `server.js`.
3. Environment: `DATABASE_URL`, `APP_SECRET`, `APP_URL=https://your-domain`, `TRUST_PROXY=true`, `TZ=Asia/Kolkata`.
4. `npm install` and `npm run setup -- --no-seed` (or seed, then delete the demo logins).
5. Open `https://your-domain/admin`, run the setup wizard and add your licence key in Admin → Licence. On a real domain, new payment requests stay off until the licence is active; no key is needed on localhost.
6. Set up SMTP in Admin → Settings → Email so reminders and KYC emails are delivered.
7. Keep `storage/kyc/` private, writable and backed up.

Going-live checklist

- Demo logins deleted or passwords changed; sample shops removed.
- Each shop has its real UPI ID saved; test with a small amount from your own UPI app.
- Collection limits, link expiry and KYC requirement set in Admin → Settings.
- https working, `TRUST_PROXY=true`, backups of the database and `storage/kyc/`.

8. What it is not, and what was not tested

- **Not a wallet.** It never holds, stores or moves money; there is no balance to top up or spend. A stored-value wallet needs an RBI PPI licence; this app holds nothing.
- **Not a money transfer service.** Nobody can send money to another person or shop through it.
- **Not a recharge or bill payment service.** Bill reminders are a to-do list with due dates and emails.
- **Not a bank or payment aggregator.** UPI payments go from the customer's bank straight to the merchant's bank. The daily settlement report lists what was confirmed; it is not a payout.
- **No gateway is bundled.** The add-on system can add "Pay online" and webhook confirmation; it is tested only with a mocked gateway and a test signing secret. If one gateway account collects for many merchants, the platform acts as a payment aggregator, which needs RBI authorisation.
- Not tested: native mobile builds, paying a real UPI QR with a real UPI app, real gateways, MySQL and PostgreSQL, sending through a real SMTP server.

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