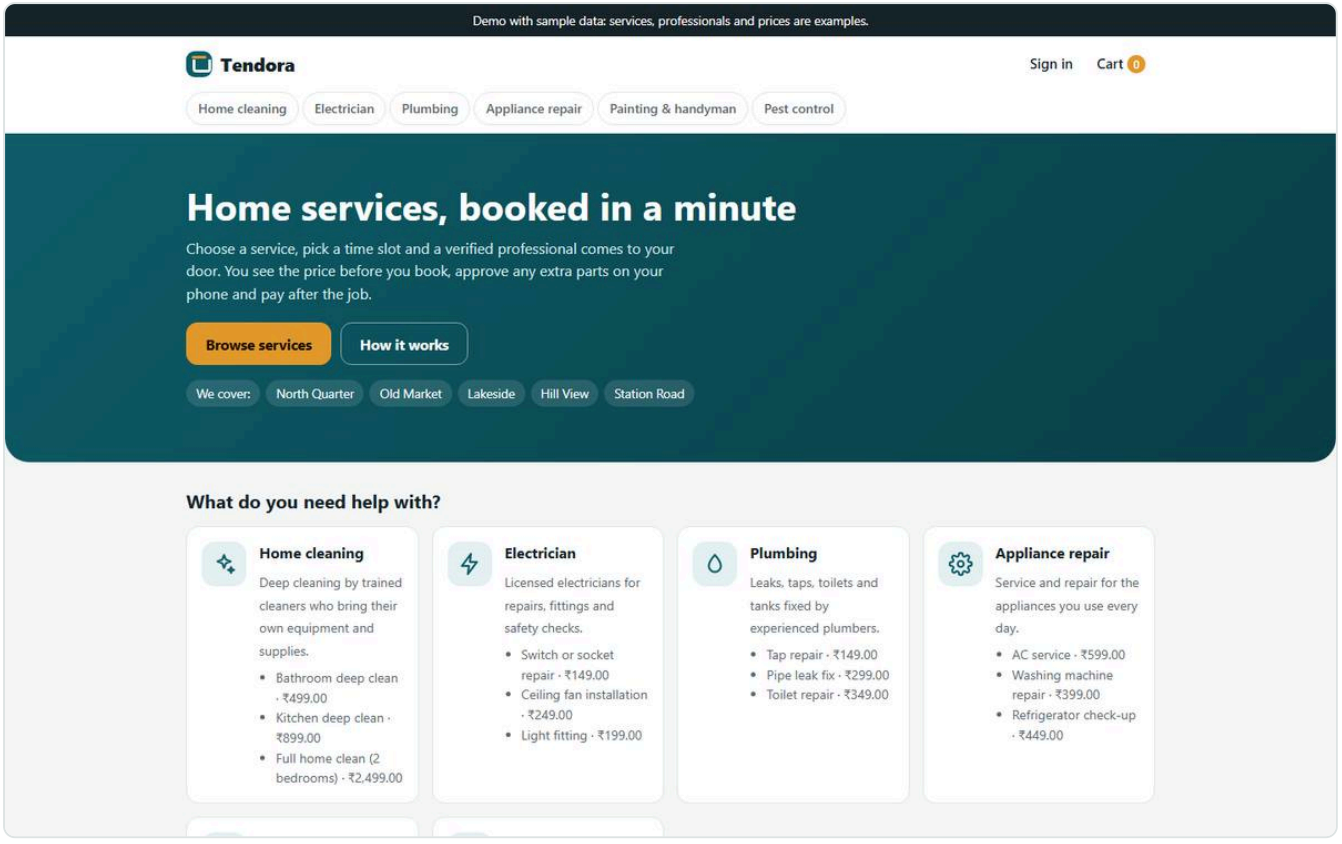
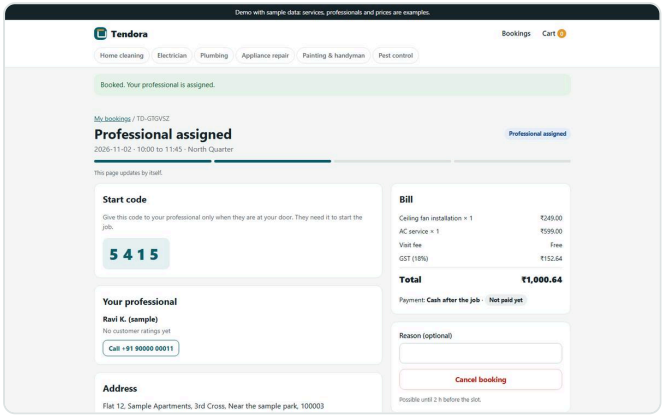
CODE CRAFT · INSTALLATION GUIDE

Tendora Home Services App

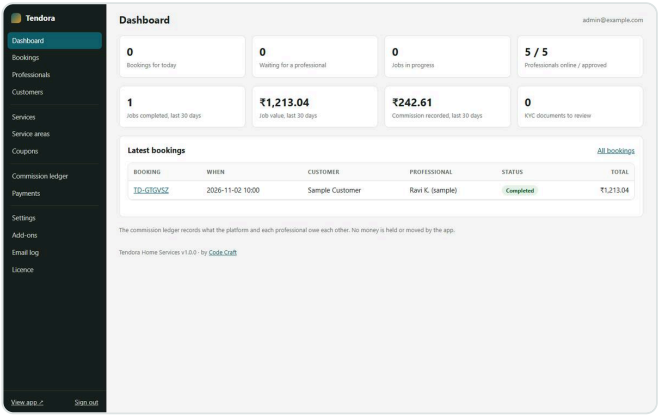
Tendora 1.0: a home services booking app with a customer web app, a professional panel, an admin panel, a REST API and an Expo mobile app. Node.js, Express, EJS and Prisma. This guide takes you from the download to a live app on Hostinger, cPanel or a VPS.



The customer web app with the sample catalogue. The brand, colours, services, areas and prices are all changed from the admin.



A booking with its 4-digit start code and the assigned professional.



Admin dashboard: bookings, jobs, job value, commission and KYC to review.

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: customer web app (`web.js`), professional panel (`pro.js`), admin (`admin.js`) and REST API v1 (`api.js`).
`src/lib/`	Booking logic: slots, assignment, start codes, extra parts, completion, ledger and ratings (`bookings.js`), slot arithmetic, pricing, UPI link and QR, settings, licence, email, add-ons.
`views/`, `public/`	EJS templates for the web app, pro panel and admin; CSS, JavaScript and icons.
`prisma/`	Database schema and the sample data (a made-up city with 5 areas, 6 categories, 20 services and 6 sample professionals).
`mobile/`	The Expo (React Native) app with a customer mode and a professional mode. See `mobile/README.md`.
`addons/`	Drop-in add-ons (for example a payment gateway). None is bundled; `ADDONS.md` explains how to write one.
`test/`	Unit, REST API and web-form tests, 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 and user.
- A domain with https for the live app, and a map tile provider (see section 7).
- For the mobile app: Node.js, an Expo account, and Google Play / Apple developer accounts to publish.

2. Run it on your computer

1. Install Node.js 20 or 22 from nodejs.org.
2. Unzip the download and open a terminal in the folder.
3. Run the three commands below. `npm run setup` creates `.env` with a random `APP_SECRET`, the tables and the sample data.

```
npm install
npm run setup
npm start
```

Page	Address
Customer web app	http://localhost:3730/
Professional panel	http://localhost:3730/pro (professionals sign in at /login)
Admin	http://localhost:3730/admin
REST API	http://localhost:3730/api/v1

Other commands

- `npm run setup -- --reseed`: wipes bookings, catalogue, professionals, customers and settings and loads the sample data again (admin users are kept).
- `npm run setup -- --no-seed`: empty tables; the first visit to /admin runs the setup wizard.
- `npm run seed -- --no-demo-users`: sample catalogue without demo logins.
- `npm run dev` restarts on file changes; `npm test` runs the tests.

Demo logins (sample data only)

Created by `npm run setup` for trying the app. **Change or delete them before you go live.**

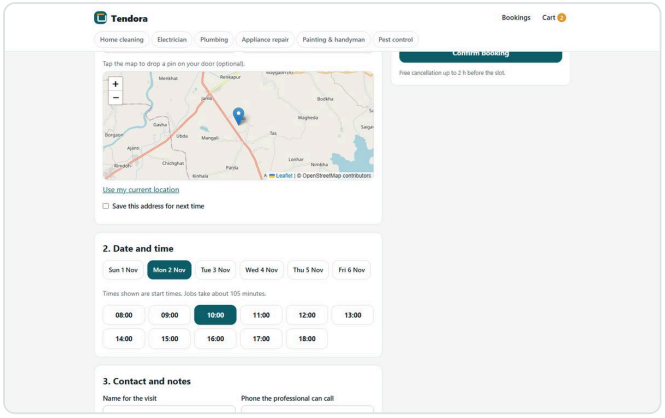
Role	Email	Password
Admin	admin@example.com	`Demo-Admin-2026!`
Customer	customer@example.com	`Demo-Customer-2026!`
Professional: electrician + appliances	pro.electric@example.com	`Demo-Pro-2026!`
Professional: cleaning (all areas)	pro.clean@example.com	`Demo-Pro-2026!`
Professional: plumbing + electrician	pro.plumb@example.com	`Demo-Pro-2026!`
Professional: cleaning + pest control	pro.pest@example.com	`Demo-Pro-2026!`
Professional: painting & handyman	pro.handy@example.com	`Demo-Pro-2026!`
Professional waiting for KYC	pro.new@example.com	`Demo-Pro-2026!`

> The city, areas, services, prices, professionals and customer are made up and marked "(sample)". No bookings, ratings or reviews are created: every rating the app shows comes from a real customer rating a

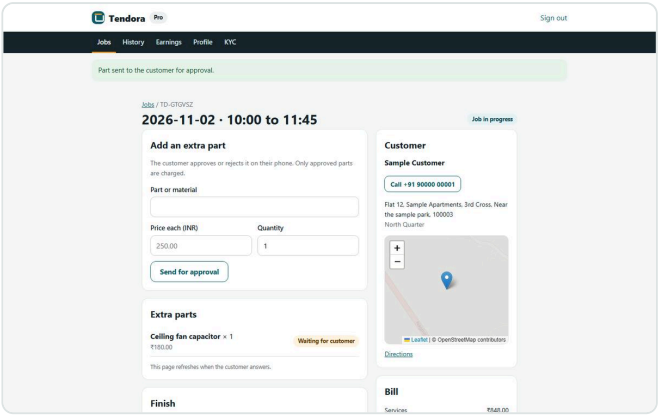
> completed job.

3. How a booking works

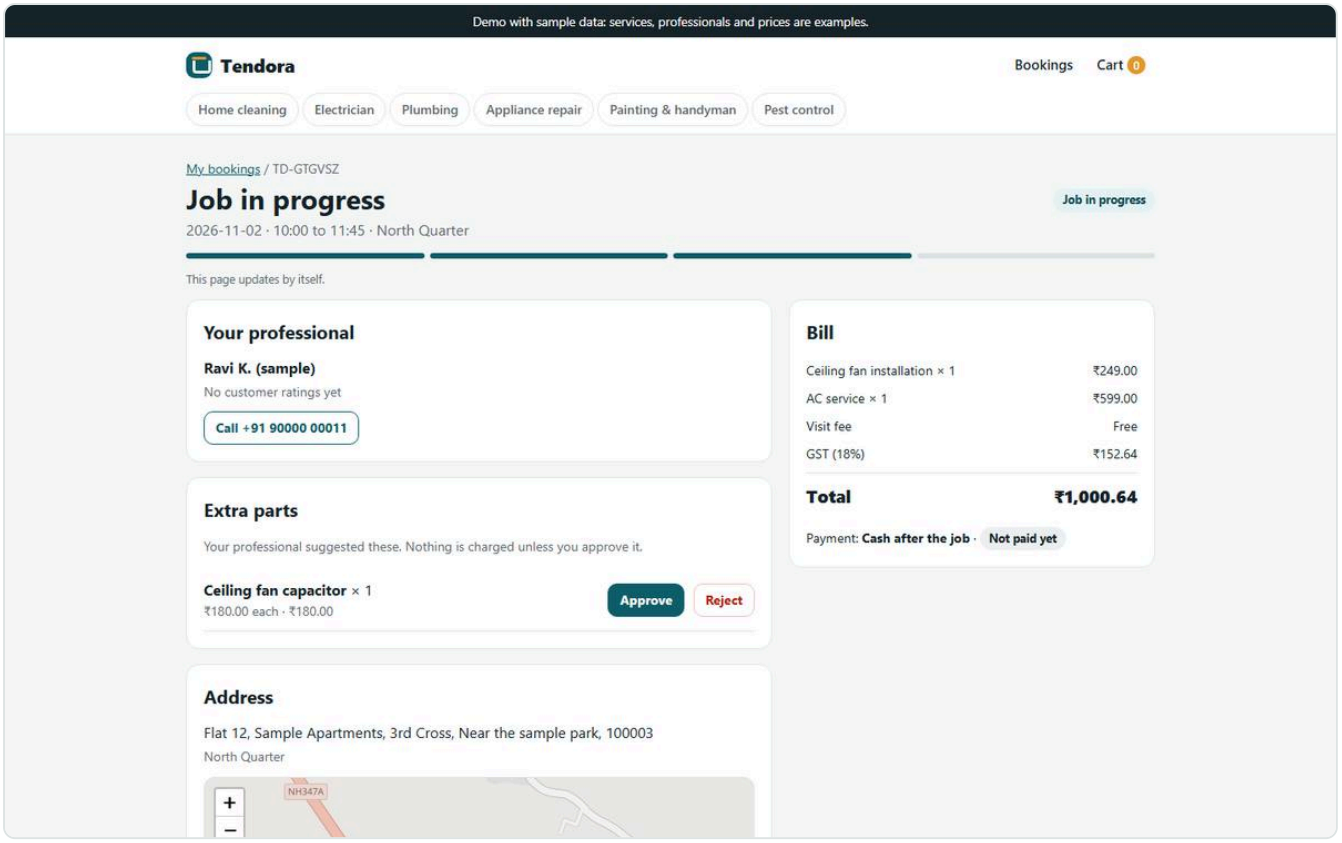
1. The customer adds services to the cart and picks an area, address, date, one of the free start times and a payment method.
2. The server re-prices everything, checks the slot and assigns a professional (or leaves the booking for the admin when auto-assign is off).
3. The customer sees the professional and a 4-digit start code. The professional never receives the code.
4. At the door the professional enters the code (after 5 wrong tries it locks until an admin resets it).
5. The professional can add extra parts; the customer approves or rejects each one. Only approved parts are charged.
6. The professional completes the job and records cash received, or the UPI reference (UTR). The ledger records the commission.
7. The customer rates the job once.



Booking form: area, address with a map pin, date strip and free start times.



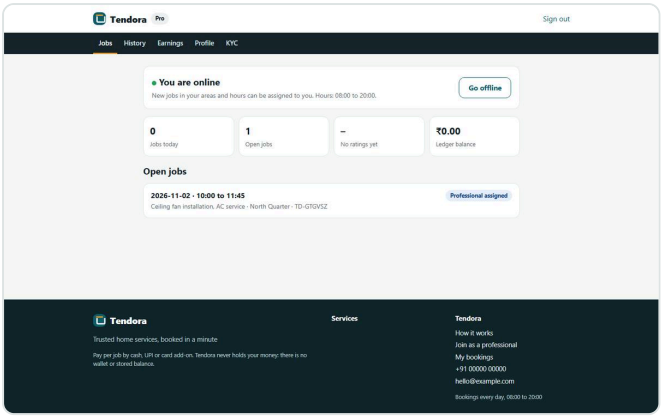
Professional job screen after the start code: add an extra part for approval.



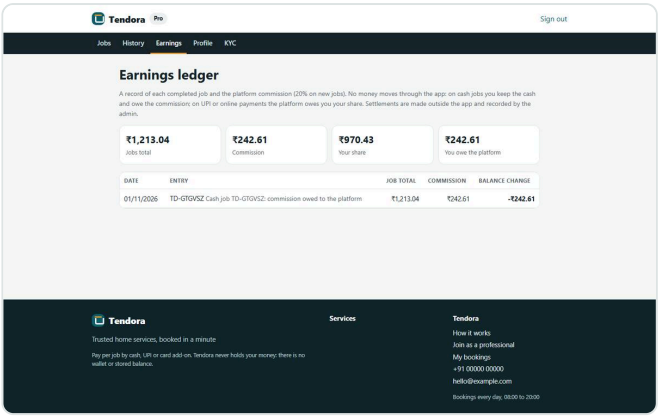
The customer approves or rejects each extra part on their booking page or in the app.

4. The professional panel

Professionals sign up at `/join` with their skills and areas, then upload an ID document at `/pro/kyc` (JPEG, PNG, WebP or PDF, up to 5 MB). KYC files are stored in `storage/kyc/`, outside the public folder, and only admins can open them. After you approve the profile and the KYC, the professional can go online and receive jobs.



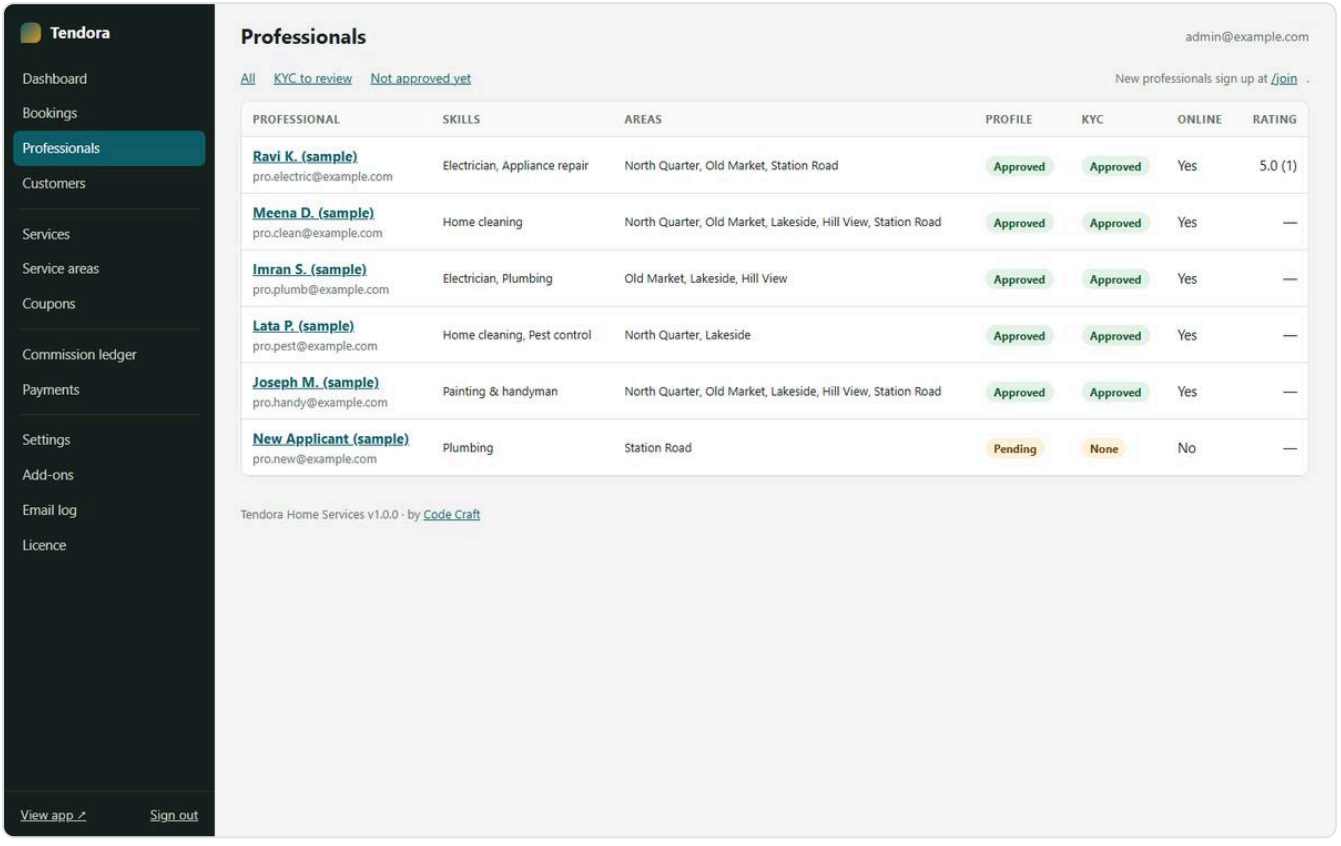
Jobs: online switch, today's numbers and open jobs (refreshes every 30 seconds).



Earnings ledger: job totals, commission and the balance with the platform.

5. The admin panel

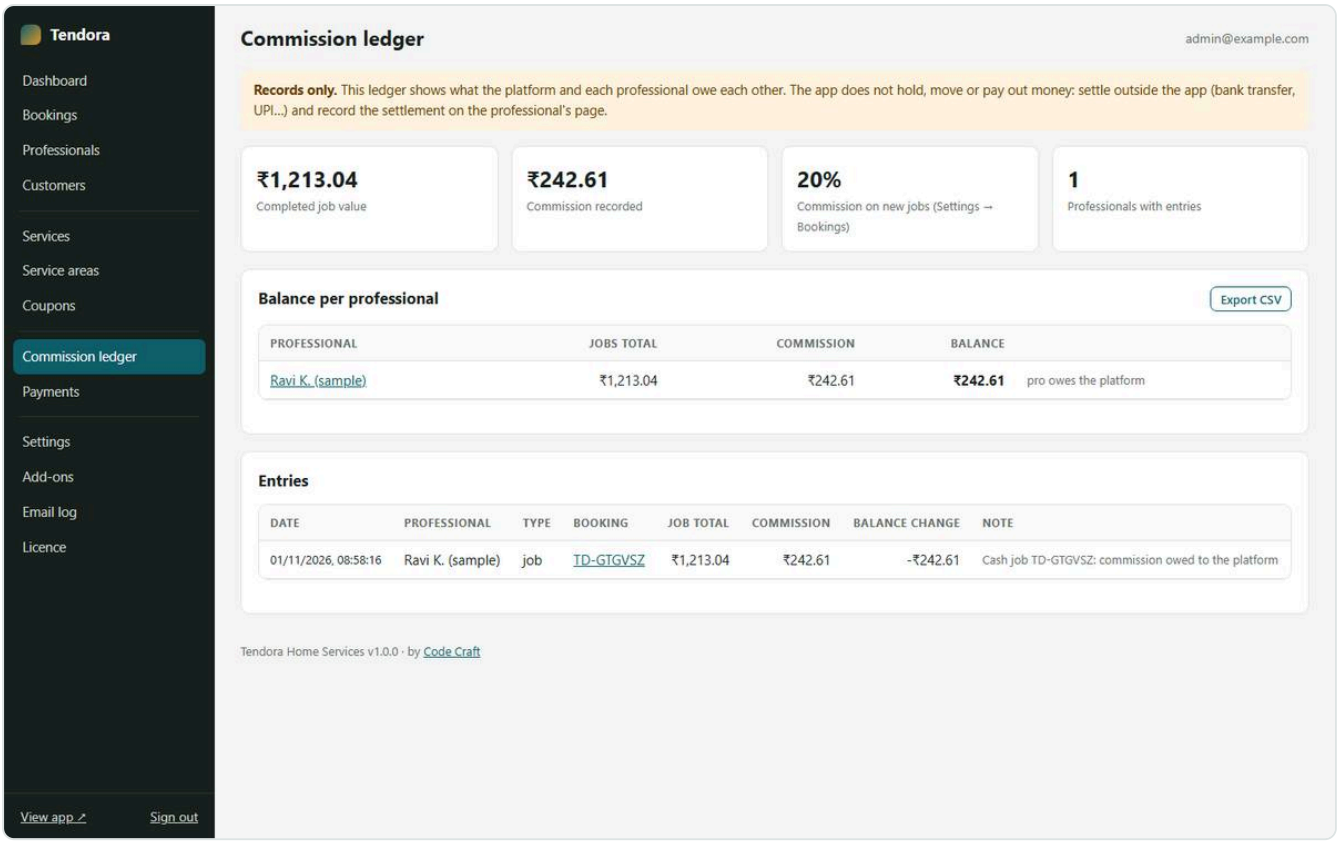
Screen	What you do there
Dashboard	Today's bookings, bookings waiting for a professional, jobs in progress, professionals online, 30-day totals, KYC to review.
Bookings	Filter, export CSV, assign or reassign (only qualified, free professionals are listed), cancel, record UPI or cash payment, reset a locked start code.
Professionals	Approve or suspend, skills, areas, working hours, KYC document viewer with approve / reject and a note, settlements.
Customers, Coupons	Block customers; percent or fixed coupons with limits and expiry.
Services, Service areas	Categories and services with price and duration; areas with PIN codes.
Commission ledger	Balance per professional and every entry, CSV export.
Payments	Your UPI ID and which payment methods are on.
Settings	Business, logo and colours, SEO, maintenance, booking rules, chat, WhatsApp, analytics, banner, cookie notice, SMTP.
Email log, Add-ons, Licence	Every email sent; add-on settings; licence key.



Professionals with skills, areas, profile and KYC status, online status and rating.

Booking rules

Admin → Settings → Bookings, slots & commission: auto-assign, commission %, visit fee and the amount above which it is free, tax, first and last slot, slot step, travel buffer, lead time, days ahead, cancel cut-off, start-code tries and the map centre. Slots use the server's time zone: set `TZ` (for example `Asia/Kolkata`).



Commission ledger: what each professional owes you or you owe them. Settle outside the app and record it.

6. Payments and the ledger

- Customers pay per job: **cash** to the professional, or **UPI** to your business UPI ID with a `upi://pay` link and QR code for the exact total. UPI payments are confirmed by hand with the UTR; nothing is confirmed automatically.
- There is **no wallet, stored balance or prepaid credit**. In India, holding customer money as a balance needs an RBI licence for prepaid payment instruments (PPI).
- The commission ledger is a **record only**. You settle with professionals outside the app (bank transfer, UPI) and record the settlement on the professional's page.
- Card gateways: write or buy an add-on using the payment contract in `ADDONS.md`. No gateway add-on is included. Test any gateway in its sandbox before taking real payments.

7. Configuration (.env)

Variable	Meaning
`DATABASE_URL`	`file:./app.db` (SQLite, default), `mysql://user:pass@host:3306/db` or `postgresql://user:pass@host:5432/db`
`PORT`	Port (default 3730; hosts usually set it)
`APP_SECRET`	Long random string that signs session cookies
`APP_URL`	Your public https address (emails, payment links, the mobile app)
`TRUST_PROXY`	`true` behind Hostinger, cPanel or Nginx with https
`LICENSE_KEY`	Optional; or paste the key in Admin → Licence
`MAP_TILE_URL`, `MAP_ATTRIBUTION`	Map tiles (see below)
`TZ`	Time zone for slots

Maps

The address pin and job maps use Leaflet with OpenStreetMap tiles. The public OpenStreetMap tile server is fine for trying the app, but its usage policy does not allow heavy traffic. **For a live site, use your own tile provider** (most need an API key):

```
MAP_TILE_URL="https://tiles.example.com/{z}/{x}/{y}.png?key=YOUR_KEY"
MAP_ATTRIBUTION="&copy; Your provider &copy; OpenStreetMap contributors"
```

MySQL or PostgreSQL

Set `DATABASE_URL` to the `mysql://` or `postgresql://` address and run `npm run setup`: it switches the Prisma schema to that provider and creates the tables. We tested on SQLite only.

8. Put it online

Hostinger (Node.js app) or cPanel (Setup Node.js App)

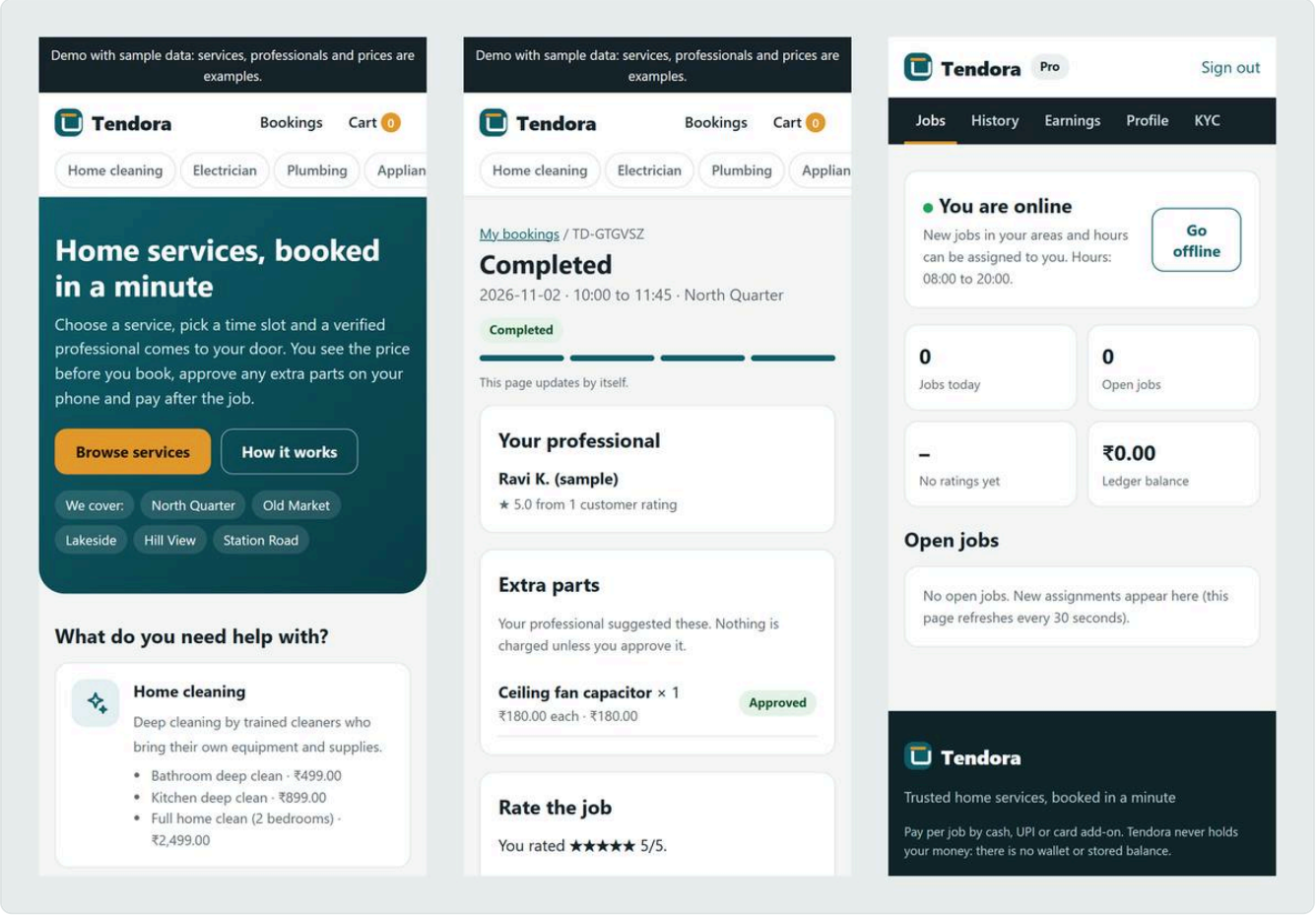
1. Upload the files without `node_modules`, `.env` and `*.db`.
2. Choose Node 20 or 22 and the startup file **server.js**.
3. Set the environment: `DATABASE_URL`, `APP_SECRET`, `APP_URL=https://your-domain`, `TRUST_PROXY=true`, `TZ` and `MAP_TILE_URL`.
4. Run `npm install`, then `npm run setup -- --no-seed` (or without `--no-seed` to start with the sample data, then delete the demo logins).
5. Start the app, open `https://your-domain/admin` and add your licence key in Admin → Licence. On a real domain new bookings stay switched off until the licence is active. 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.
6. Make sure `storage/kyc/` and `public/uploads/` are writable and included in your backups.

VPS

```
npm install
npm run setup -- --no-seed
npx pm2 start server.js --name tendora
# put Nginx in front with https (Certbot) and set TRUST_PROXY=true
```

9. The mobile app

The `mobile/` folder is an Expo SDK 58 app (TypeScript, Expo Router). The signed-in account's role decides the mode: customers book and follow jobs; professionals take jobs, start them with the code, add parts, complete and record payment. Admins use the web panel. Name, colours, currency and payment methods come from your server.



Phone-width views of the web app: home, a completed booking and the professional panel.

```
cd mobile
npm install
cp .env.example .env          # set APP_SERVER_URL=https://your-domain
npx expo start                 # w = web, a = Android, i = iOS
npm run typecheck
```

Build for the stores with your own Expo account: `npx eas-cli@latest build -p android --profile preview` for an installable APK, `npx eas-cli@latest build -p ios` with an Apple developer account. See `mobile/README.md`.

> We have not run native Android or iOS builds. The app type-checks and its web export was used against the server in a browser. Build and test on real devices before you publish.

REST API

Everything the app does goes through `/api/v1` with bearer tokens: catalogue, areas, quotes and free slots, accounts, customer bookings, professional jobs and admin bookings, professionals and ledger. `API.md`

documents every endpoint.

10. Going-live checklist

- Delete or change every demo login (Admin → Professionals / Customers), or start with `--no-seed`.
- Replace the sample city, areas, services and prices; set your business details, logo and colours.
- Set your booking rules, tax and commission; add your UPI ID in Admin → Payments and switch UPI on.
- Set `MAP_TILE_URL` to your tile provider and `TZ` to your time zone.
- Enter your SMTP server in Admin → Settings (until then, emails are kept in Admin → Email log).
- Activate the licence in Admin → Licence and back up the database, `storage/kyc/` and `public/uploads/`.

What this app does not do

- No payment gateway is bundled; UPI is confirmed by hand.
- No wallet, stored balance, refunds processing or automatic payouts.
- Native mobile builds, MySQL / PostgreSQL and real SMTP sending were not part of our test run.

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