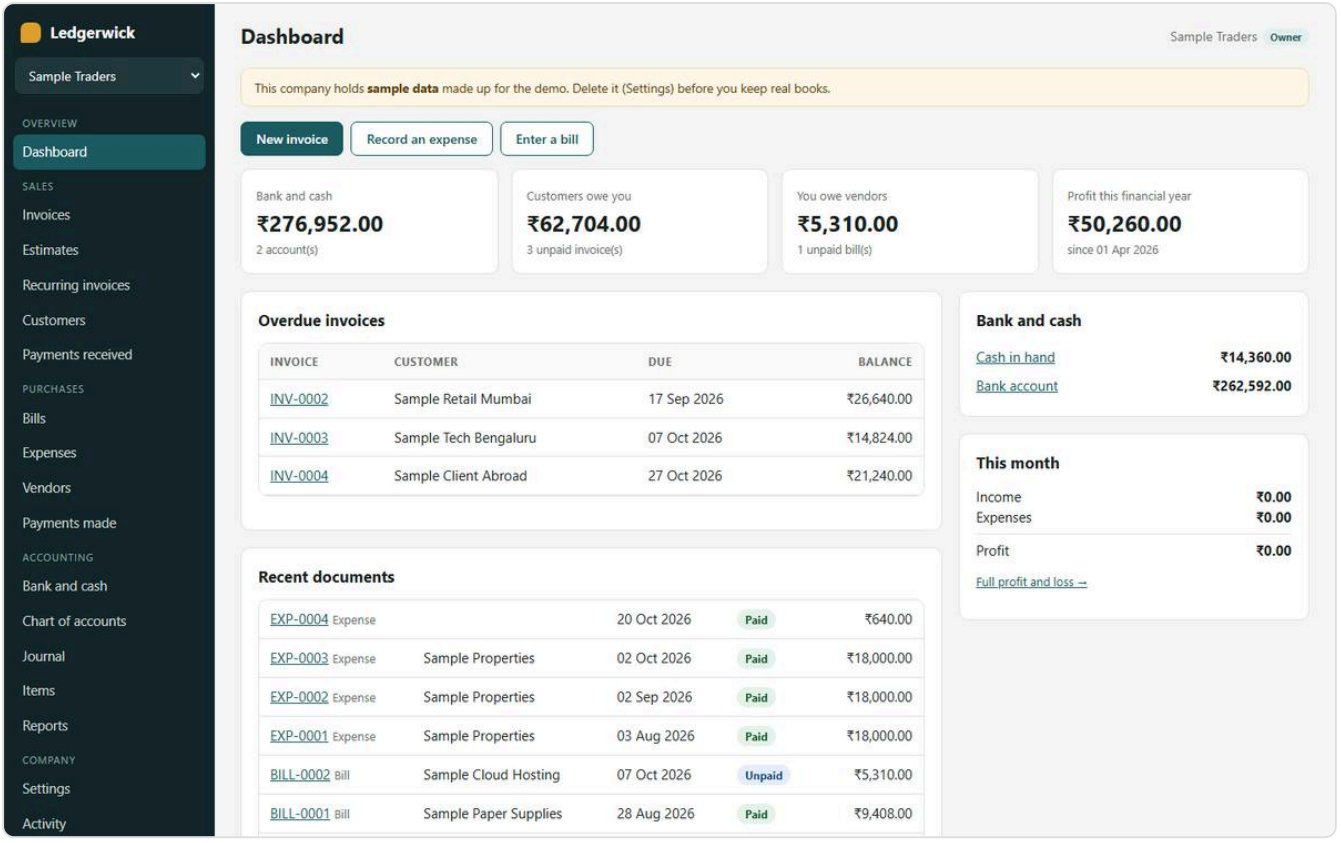
CODE CRAFT · INSTALLATION GUIDE

Ledgerwick Accounting App

Double-entry books for small businesses: GST or VAT invoices, bills, expenses, bank ledgers and reports, with an admin panel, a REST API and an Expo mobile app. Node.js + Express + EJS + Prisma. This guide takes you from download to a live installation on Hostinger, cPanel or a VPS.



A company's dashboard with the sample data: bank and cash, receivables, payables, profit this financial year and overdue invoices.

> **Tax.** The GST and VAT rules are a starting point, not tax advice, and the app is not certified by any tax authority. Have your accountant check the tax settings (regime, rates, place of supply, invoice wording, rounding) before you rely on them. The app does not file returns.

1. What is in the download

Folder / file	What it is
`server.js`	Start file (PORT, default 3480) and the hourly recurring-invoice timer. Hostinger, cPanel and PM2 run this.
`src/app.js`	Express app: security headers, sessions, CSRF, routing.
`src/lib/ledger.js`	The double-entry core: chart of accounts, balance checks, posting.
`src/lib/tax.js`	GST (CGST/SGST/IGST) and VAT per line, Indian state codes.
`src/lib/books.js`	Invoices, estimates, bills, expenses, payments, transfers, journals, contacts, items.
`src/lib/reports.js`, `recurring.js`, `pdf.js`	Reports and CSV, recurring invoices, the invoice PDF.
`src/routes/`	`web.js` (sign-in, companies, account), `company.js` (the workspace at `/c/:id/…`), `admin.js` (`/admin`), `api.js` (`/api/v1`).
`views/`, `public/`	EJS templates, CSS, JavaScript and images.
`prisma/`	Database schema and the sample-data seed.
`mobile/`	Expo (React Native) app source for Android and iOS.
`test/`	Unit tests, API tests and a curl smoke test.
`README.md`, `API.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 a user.
- A domain with https for the live app, and an SMTP mailbox for invoice and invitation emails.

2. Run it on your computer

1. Install Node.js, unzip the download and open a terminal in the folder.
2. Install and set up: this creates `.env` with a random `APP_SECRET`, the tables and the sample data.

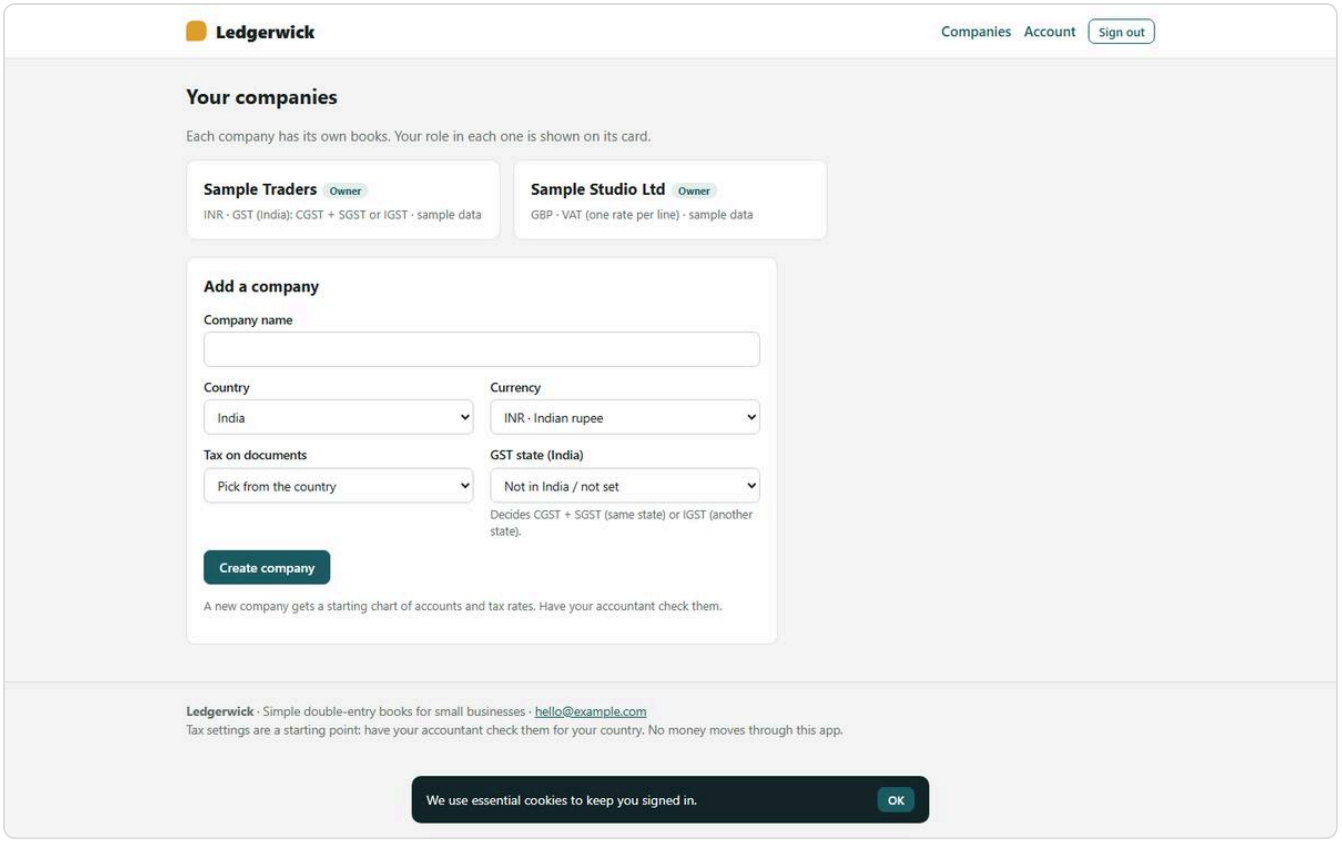
```
npm install
npm run setup
npm start          # http://localhost:3480
```

- App: `http://localhost:3480/` (sign in, then pick a company). Admin: `http://localhost:3480/admin`.
- `npm run setup -- --reseed` wipes everything and loads the sample data again.
- `npm run setup -- --no-seed` creates empty tables; the first visit to `/admin` runs a setup wizard for your admin account.
- Port: `PORT` in `.env`. If the port is busy, the app says so and stops.

Sample logins (local copy only)

All sample accounts and companies are flagged as sample and made up. Delete them, or at least change the passwords, before going live.

Who	Email	Password	What they see
Admin of the installation	`admin@example.com`	`Demo-Admin-2026!`	`/admin` (no company books)
Owner	`owner@example.com`	`Demo-Owner-2026!`	Sample Traders (GST, INR) and Sample Studio Ltd (VAT, GBP)
Accountant	`accountant@example.com`	`Demo-Accountant-2026!`	Sample Traders as accountant (no settings)
Another owner	`other@example.com`	`Demo-Other-2026!`	Sample Other Books only, to show companies are kept apart

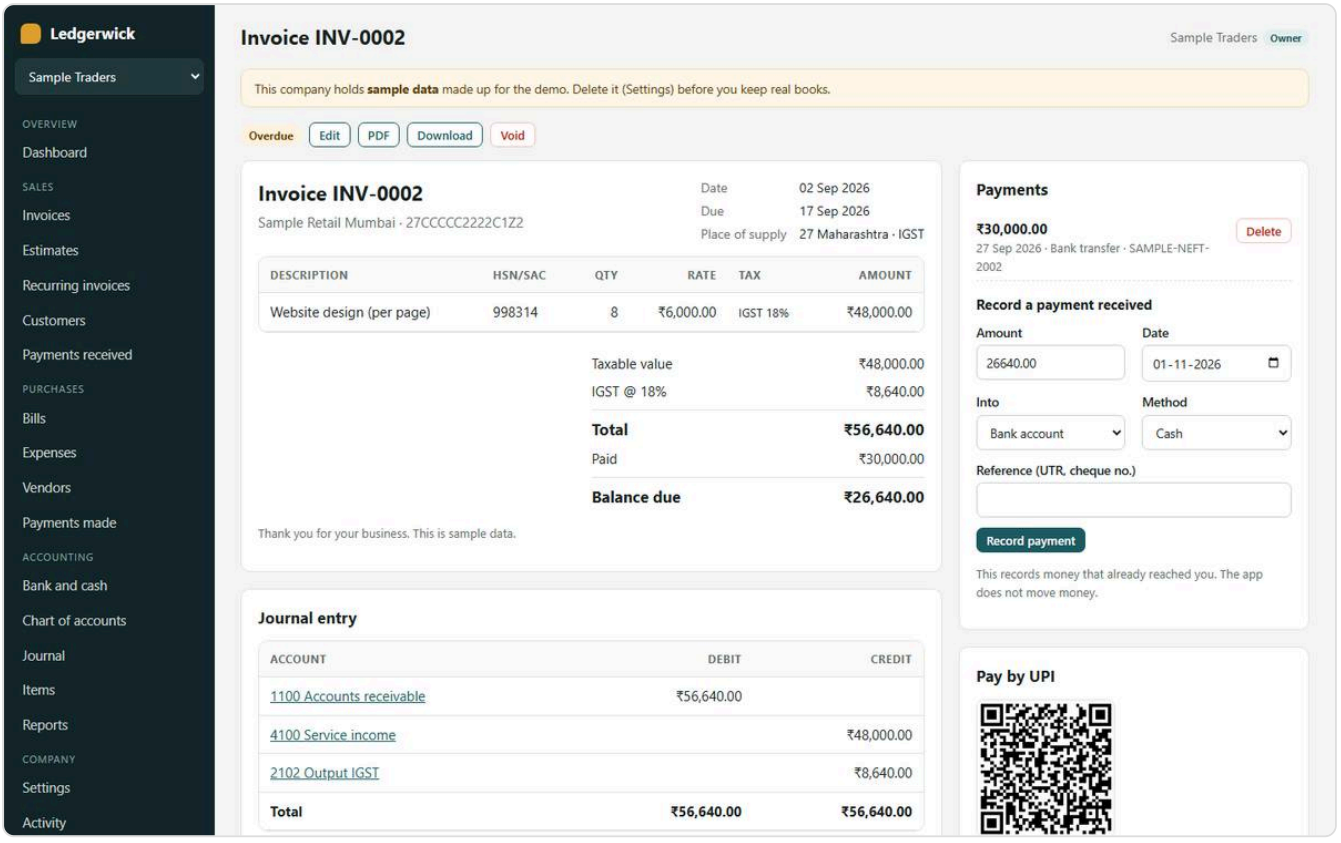


After signing in, pick a company. One login can belong to several companies, as owner or accountant.

3. How the books work

Every issued invoice, bill, expense, payment, transfer and manual journal posts one journal entry whose debits equal its credits. Drafts, estimates and void documents post nothing. Editing an issued document replaces its entry; voiding removes it. Reports are read from the journal only, so they always agree.

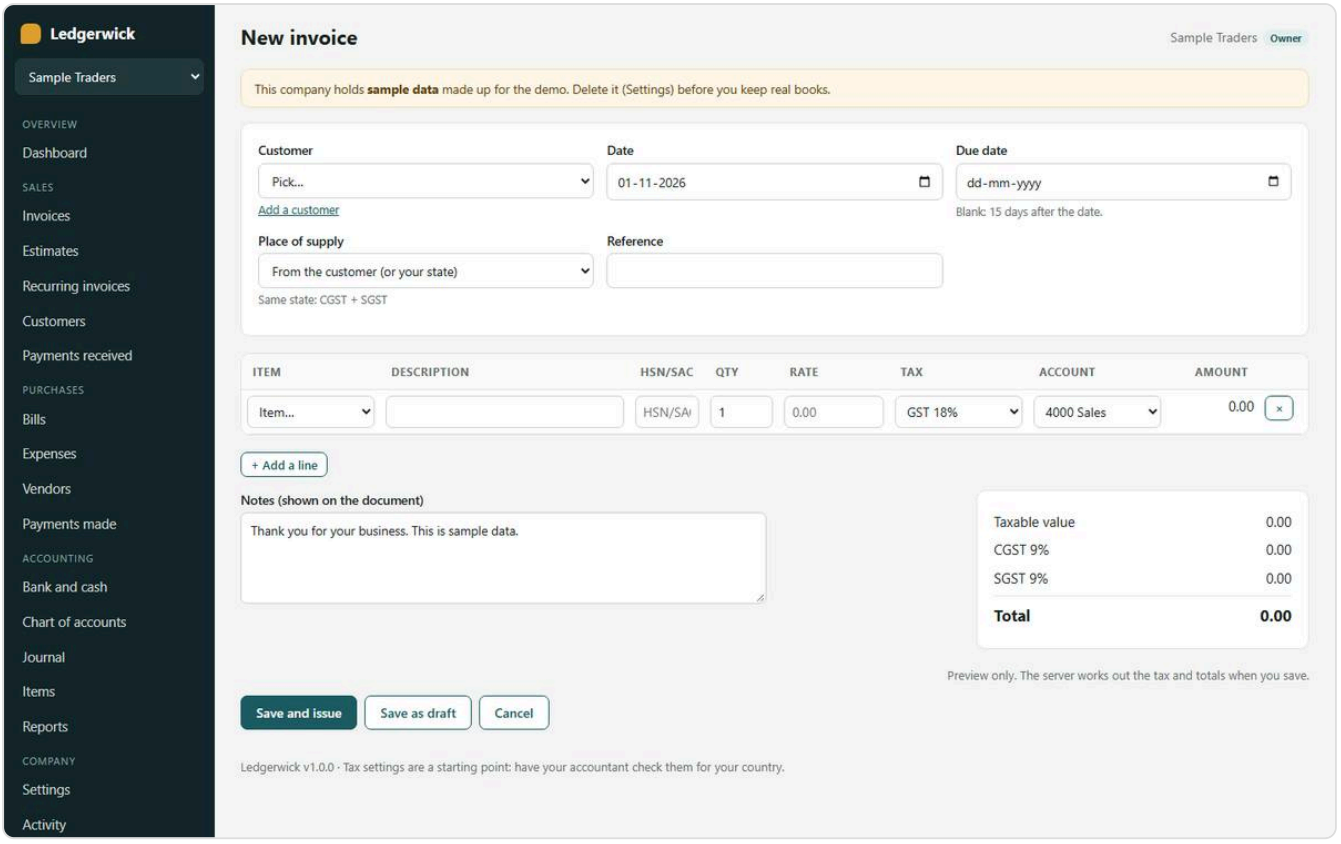
Document	Journal entry
Invoice	Dr Accounts receivable · Cr income account(s) · Cr output CGST/SGST, IGST or VAT
Bill	Dr expense or asset account(s) · Dr input tax · Cr Accounts payable
Expense	Dr expense or asset account(s) · Dr input tax · Cr the bank or cash account it was paid from
Payment received	Dr bank or cash · Cr Accounts receivable
Payment made	Dr Accounts payable · Cr bank or cash
Transfer	Dr the "to" account · Cr the "from" account



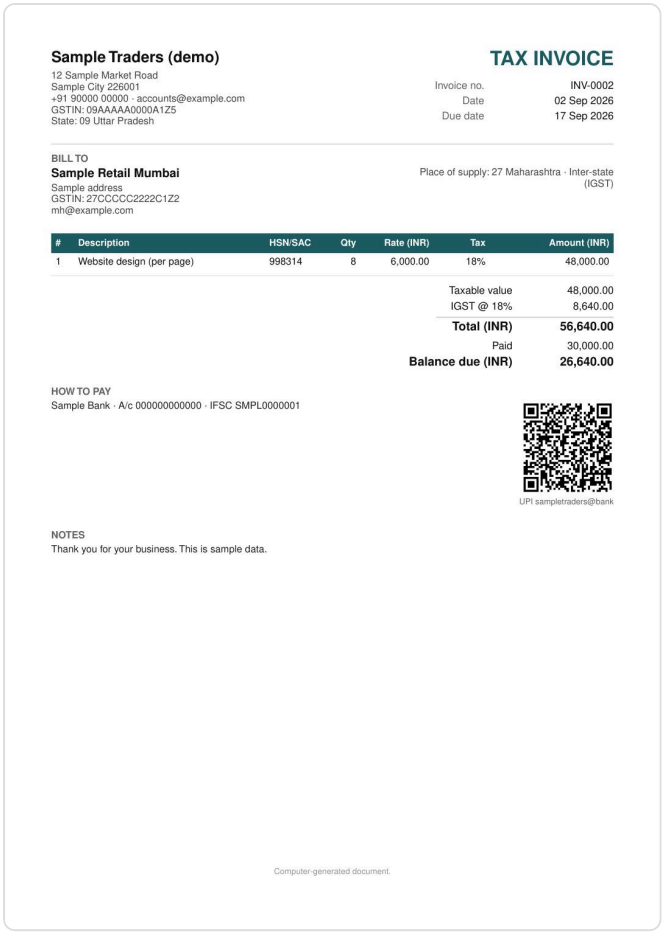
An issued, part-paid invoice: IGST for a customer in another state, the payment recorded against it, the posted journal entry and the UPI QR for the balance.

4. Invoices, tax and payments

- **GST:** an 18 % rate becomes CGST 9 % + SGST 9 % when the place of supply is your own state, and IGST 18 % for another state or a customer abroad. The place of supply comes from the customer's state and can be changed per document.
- **VAT:** one VAT line per rate. **No tax:** for businesses that do not charge tax.
- Totals and tax are calculated on the server; the form only shows a preview.



New invoice: customer, place of supply, lines from items or free text, tax rate and income account per line.



Sample Traders (demo)
12 Sample Market Road
Sample City 226001
+91 90000 00000 · accounts@example.com
GSTIN: 09AAAAA0000A1Z5
State: 09 Uttar Pradesh

TAX INVOICE

Invoice no. INV-0002
Date 02 Sep 2026
Due date 17 Sep 2026

BILL TO
Sample Retail Mumbai
Sample address
GSTIN: 27CCCCC2222C1Z2
mh@example.com

Place of supply: 27 Maharashtra · Inter-state (IGST)

#	Description	HSN/SAC	Qty	Rate (INR)	Tax	Amount (INR)
1	Website design (per page)	998314	8	6,000.00	18%	48,000.00

Taxable value 48,000.00
IGST @ 18% 8,640.00
Total (INR) 56,640.00
Paid 30,000.00
Balance due (INR) 26,640.00

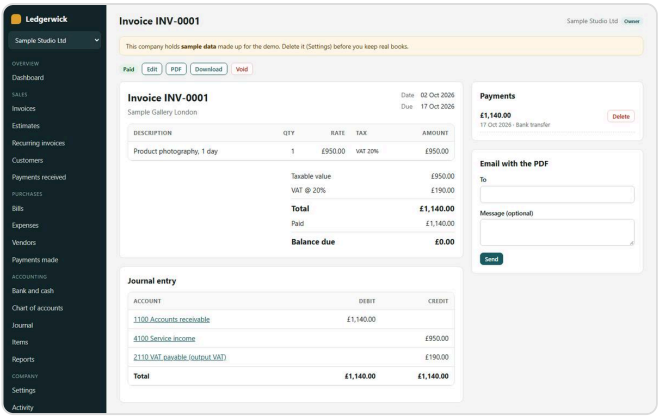
HOW TO PAY
Sample Bank · A/c 000000000000 · IFSC SMPL0000001

UPI sampletraders@bank

NOTES
Thank you for your business. This is sample data.

Computer-generated document.

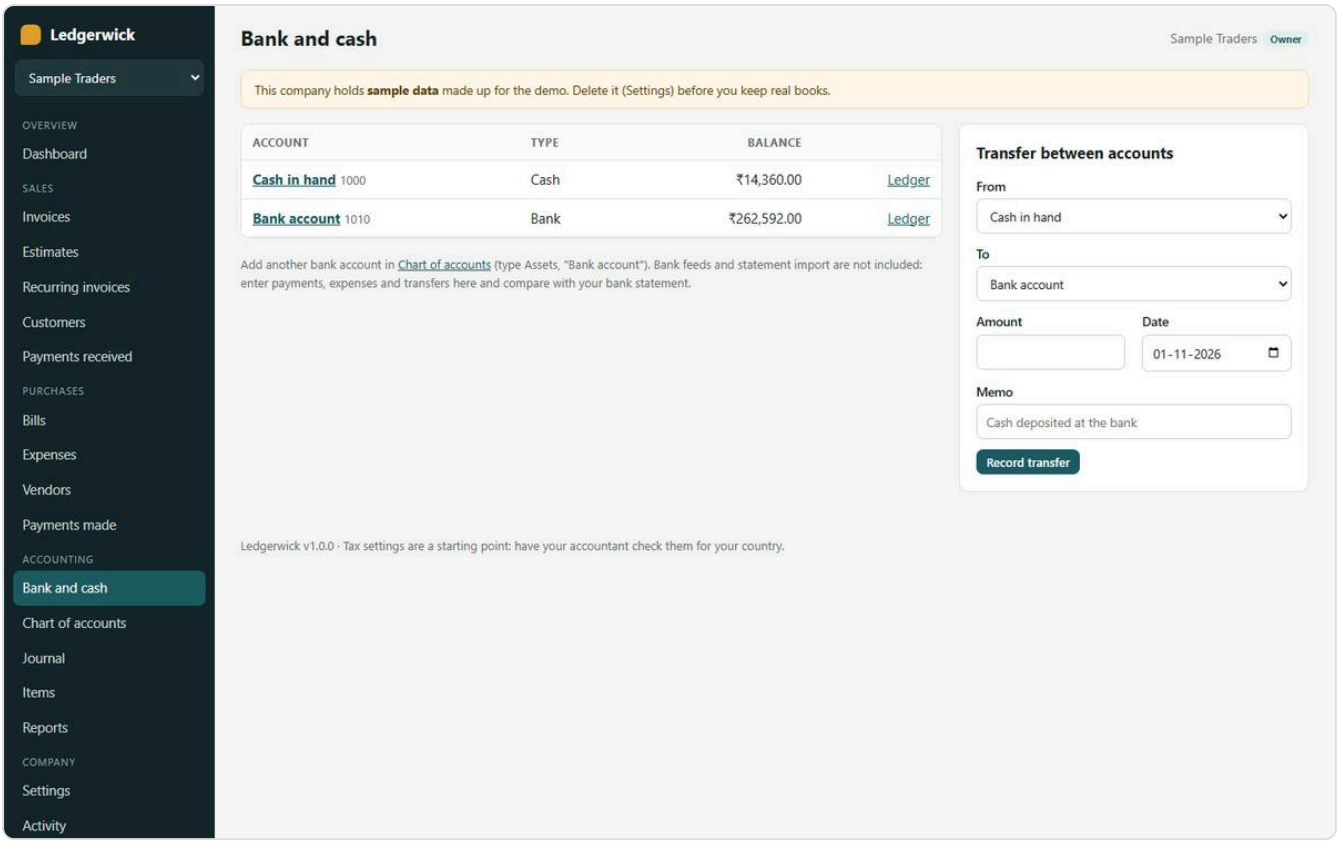
The A4 invoice PDF with GSTIN, HSN/SAC, the tax breakdown, bank details and a UPI QR.



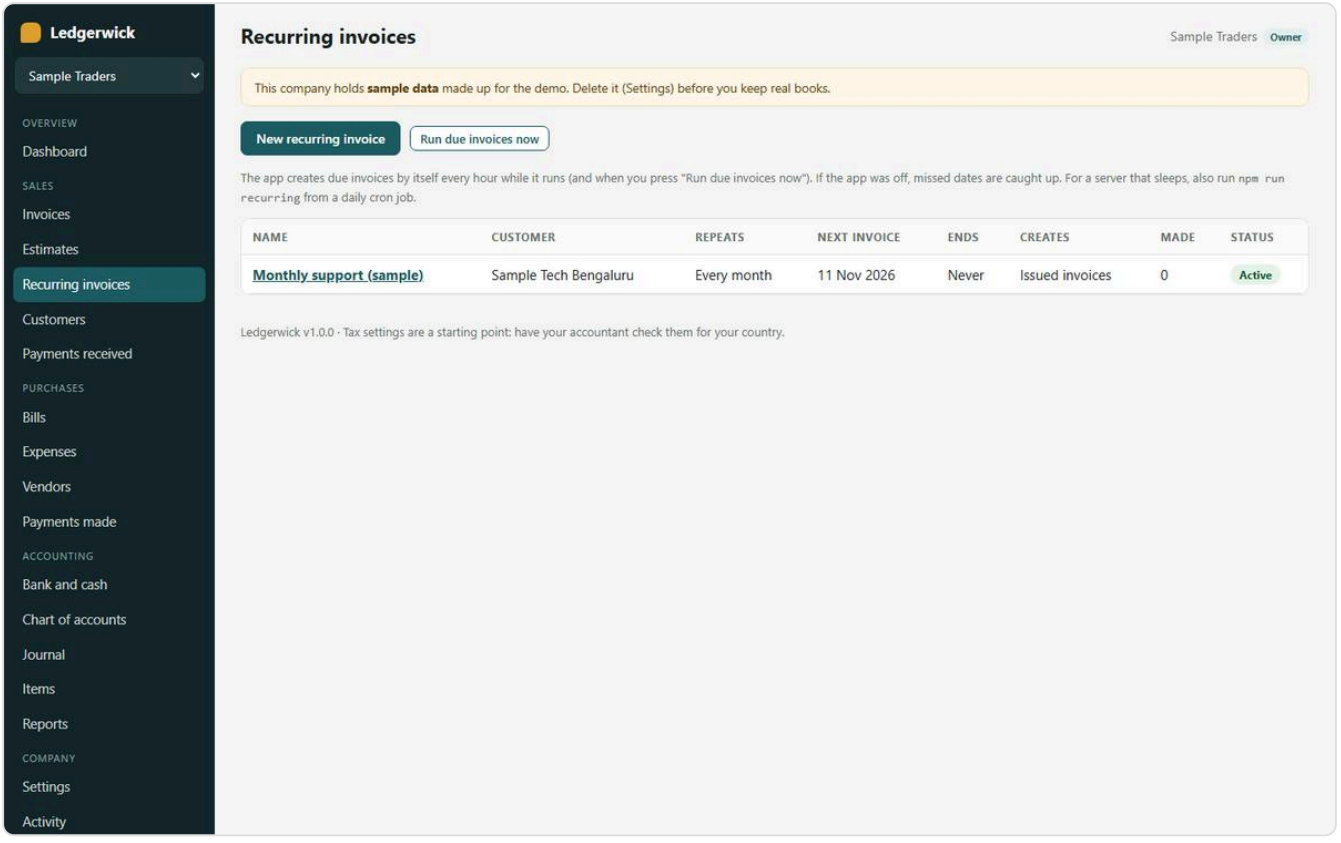
A VAT company's invoice in pounds.

> **Payments are records.** No money moves through the app. Record a payment when money reaches you (or a vendor) by cash, UPI, bank transfer, cheque or card. The UPI QR opens the payer's UPI app with the amount due, but the app cannot see or verify the transfer. There is no wallet and no payment gateway.

5. Banking, recurring invoices and reports



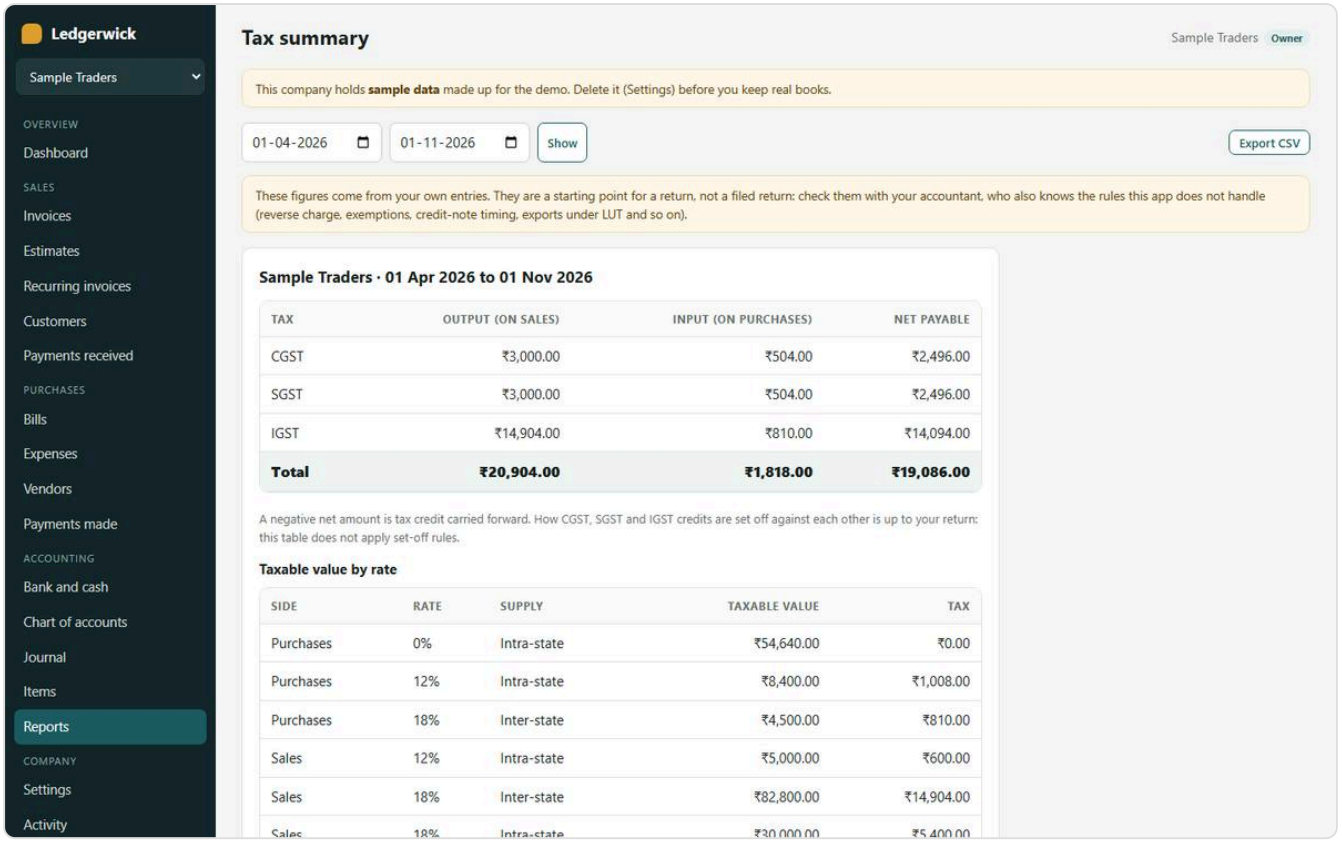
Bank and cash accounts with balances and ledgers, and a transfer between accounts. Bank feeds and statement import are not included.



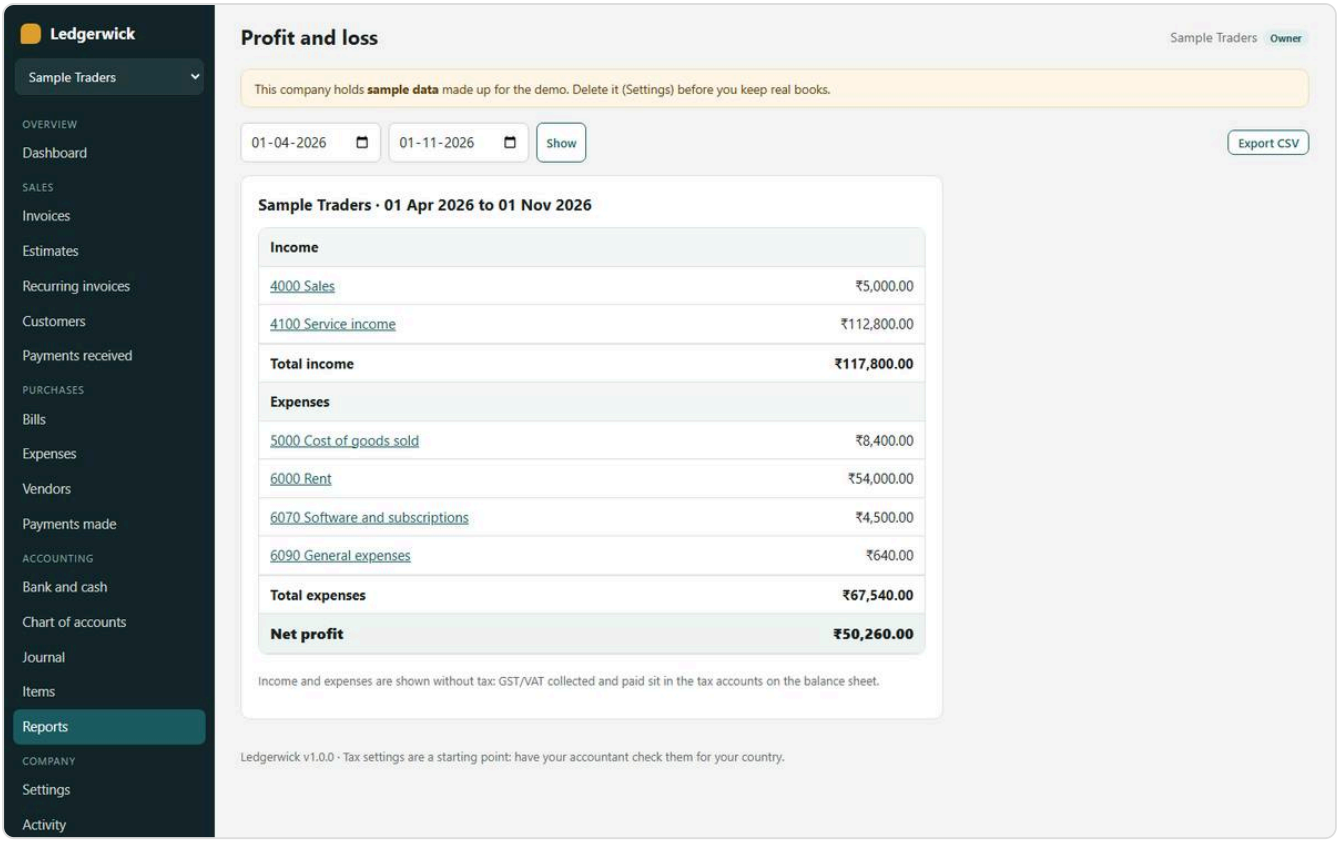
Recurring invoices: weekly, monthly, quarterly or yearly, issued or as drafts, with an end date.

The app creates due recurring invoices every hour while it runs. Month ends are kept (31 Jan → 28 Feb → 31 Mar) and missed dates are caught up. On hosting that puts idle apps to sleep, add a daily cron job:

```
cd /path/to/app && npm run recurring
```

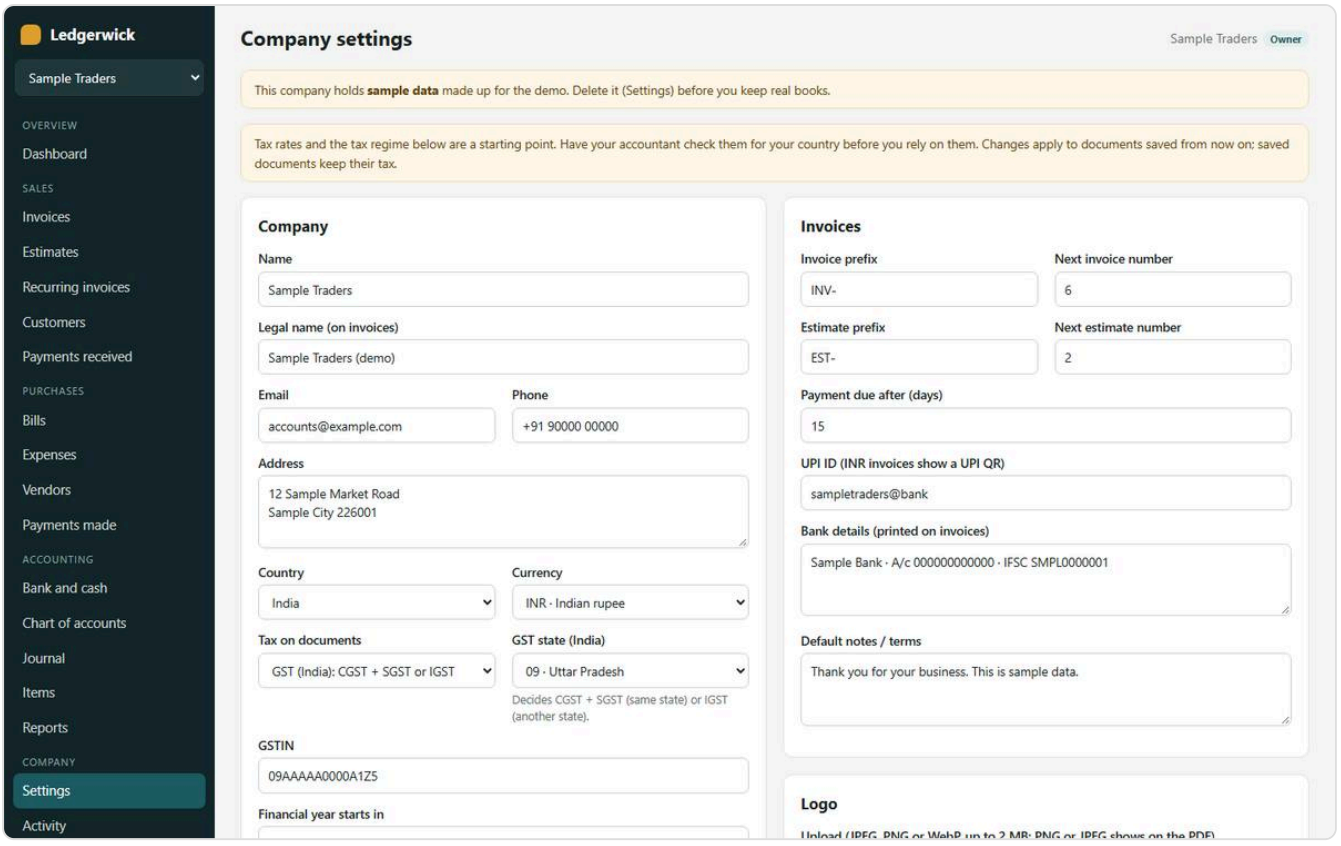


Tax summary: output and input tax per component, net payable and taxable value by rate. A starting point for your return, not a filed return.



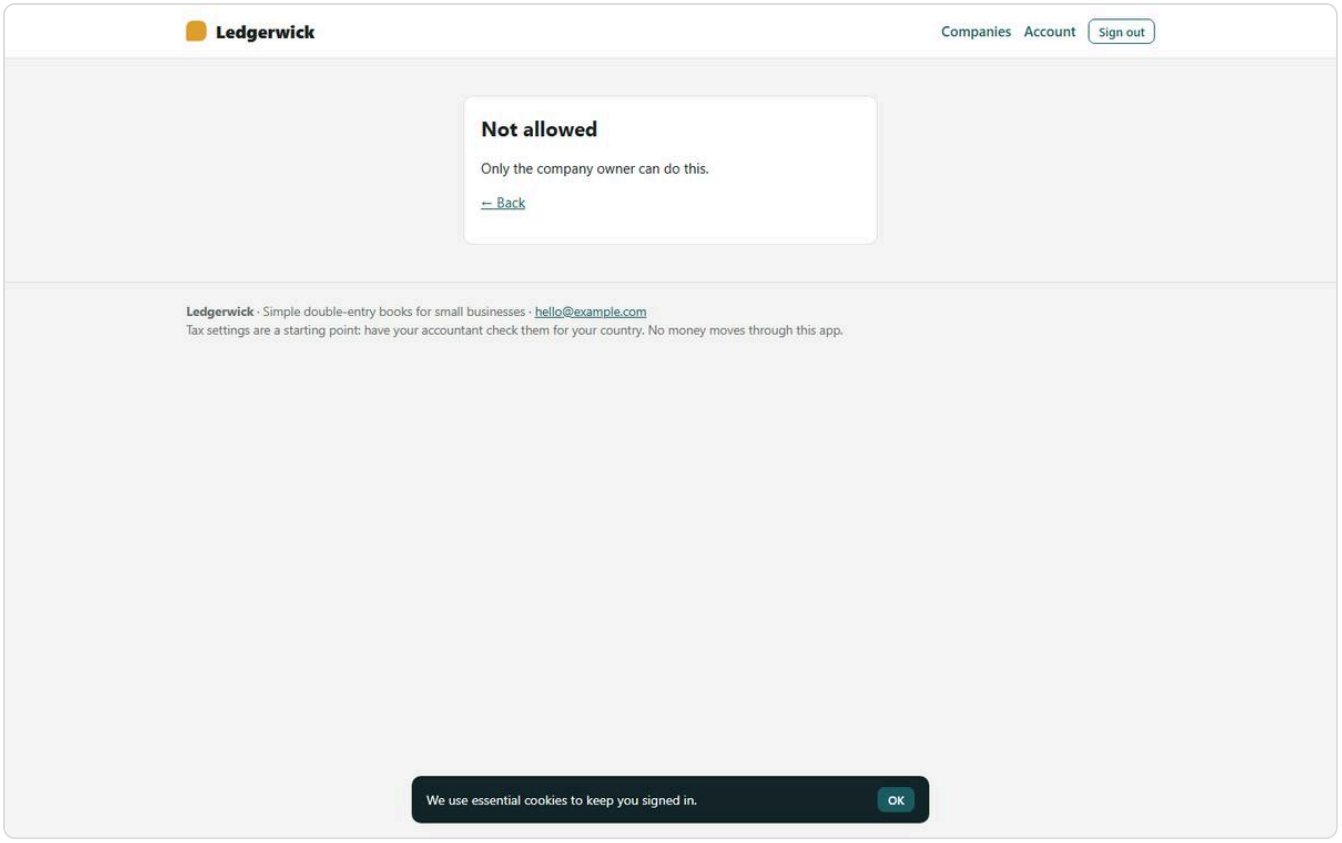
Profit and loss for any period. Every report exports to CSV.

6. Company settings and roles



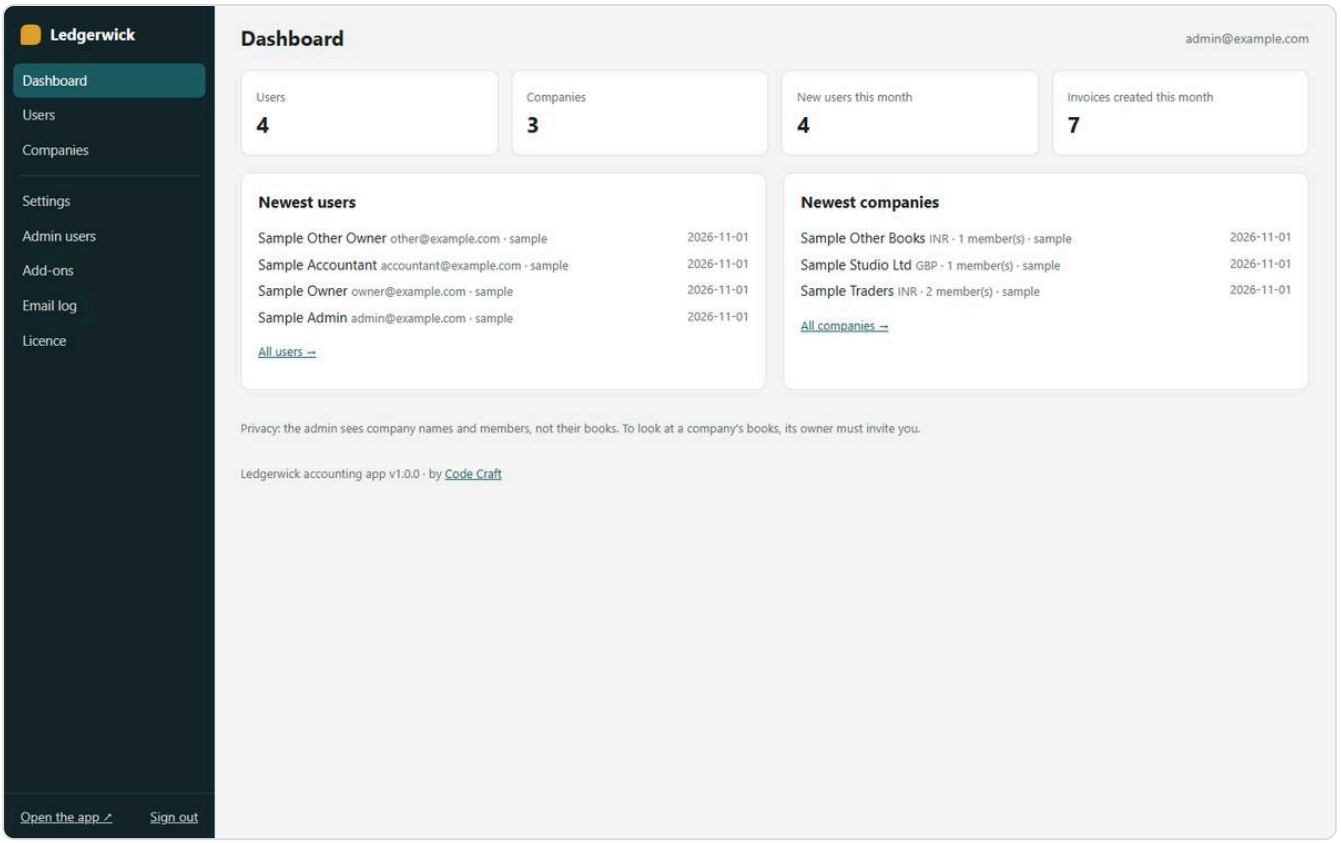
Company settings (owner only): name, legal name, country, currency, tax regime, GST state, GSTIN, financial year start, numbering, payment days, UPI ID, bank details and logo.

- **Owner:** everything, including settings, tax rates, members (invite by email, change role, remove) and deleting the company. A company always keeps at least one owner.
- **Accountant:** all bookkeeping and every report; no settings, tax rates or members.
- Someone who is not a member gets "not found" for the company, on the web and in the API.



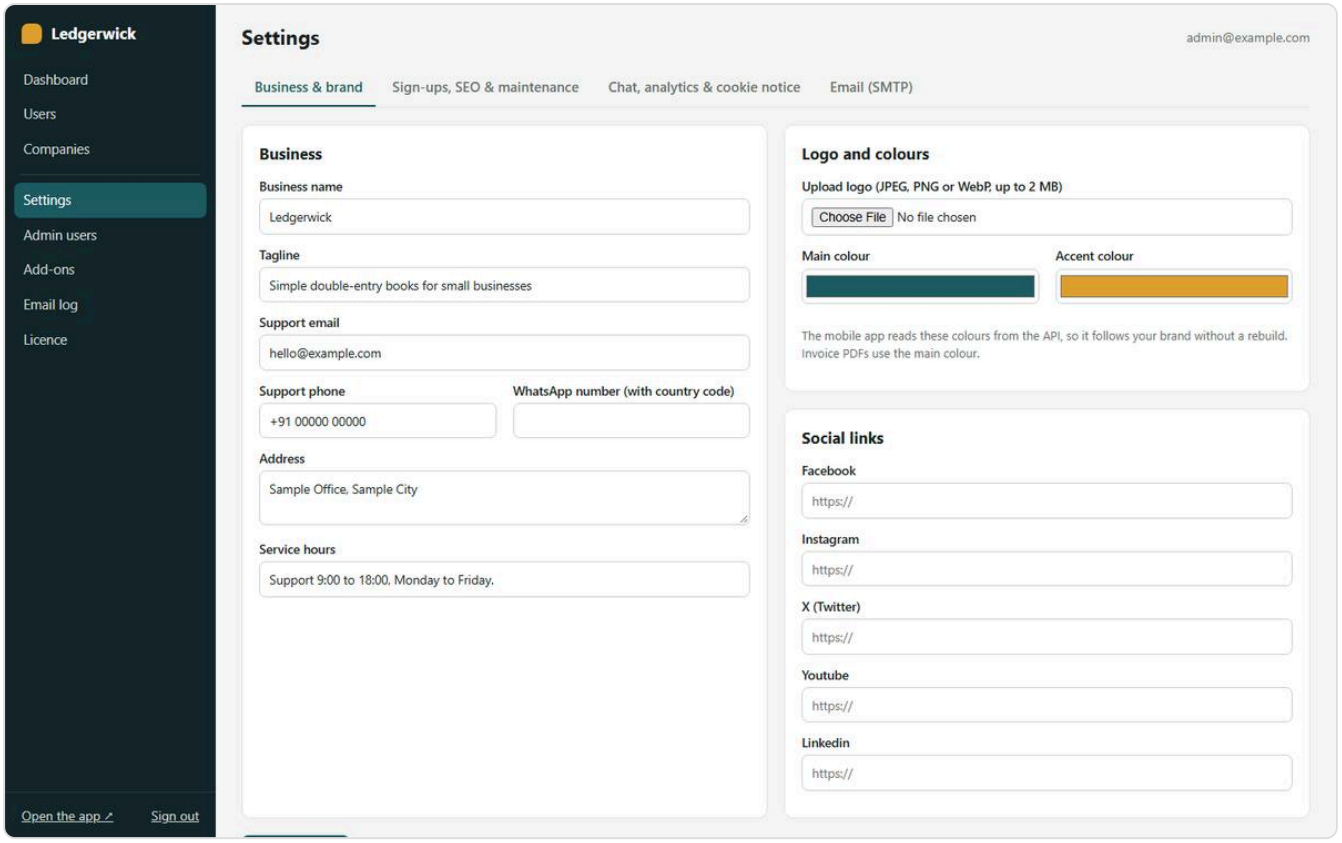
An accountant who opens company settings is refused.

7. The admin panel



Admin dashboard. The admin sees company names and members, never the books.

The admin (`/admin`) has users (switch off), companies, settings, admin users, the email log, add-ons and the licence. Settings cover the brand (name, colours, logo), sign-ups open or closed, the recurring timer, SEO, maintenance mode, live chat, analytics, the cookie notice and SMTP.



Admin settings: brand, sign-ups, SEO, maintenance mode and email.

8. Put it online

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

1. Create the Node.js app (Node 20 or 22) and upload the zip contents. The start file is `server.js`.
2. Run `npm install`, then `npm run setup` (Hostinger build command: `npm install && npm run setup`).
3. Set the environment variables below, restart the app and open `/admin`.

VPS

```
npm install && npm run setup
npm i -g pm2 && pm2 start server.js --name ledgerwick && pm2 save
# Nginx reverse proxy to 127.0.0.1:3480, then https with Certbot
```

Environment (.env)

Variable	What to set
`APP_URL`	Your https address (used in invoice and invitation links).
`TRUST_PROXY`	`true` behind https on Hostinger, cPanel or Nginx.
`APP_SECRET`	A long random string (setup creates one).
`DATABASE_URL`	SQLite by default. For MySQL `mysql://user:pass@localhost:3306/db`, for PostgreSQL `postgresql://…`, then run `npm run setup` again: it switches the Prisma provider for you.
`LICENSE_KEY`	Your Code Craft licence key (or paste it in Admin → Licence).

Licence

Until the licence is active on a live domain, new invoices cannot be created there. localhost, `*.test` and `*.local` never need a licence.

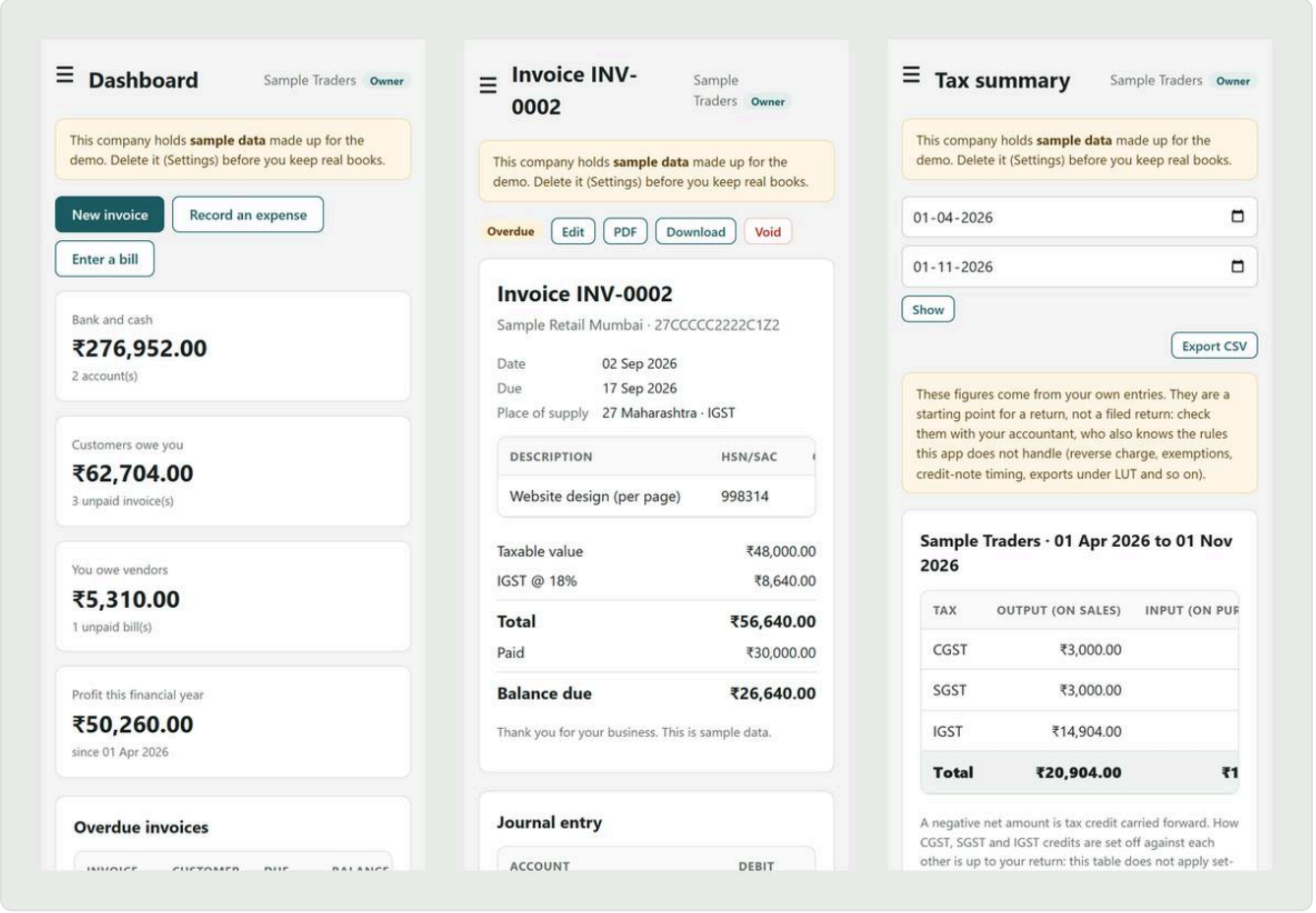
Backups

With SQLite, copy `prisma/books.db` while the app is stopped (or use `sqlite3 .backup`). With MySQL or PostgreSQL, use your host's backups or `mysqldump` / `pg_dump`. Copy `public/uploads/` too (logos).

9. REST API and the mobile app

The REST API lives at `/api/v1` and is documented endpoint by endpoint in `API.md`. Sign in with `POST /auth/login` to get a bearer token (valid 90 days, revocable from the account page). Everything about a company lives under `/companies/:cid/…`; owner-only endpoints refuse accountants. Money in requests is in minor units; totals and tax are calculated by the server.

```
curl -X POST http://localhost:3480/api/v1/auth/login \
  -H 'Content-Type: application/json' \
  -d '{"email":"owner@example.com","password":"Demo-Owner-2026!"}'
```



The web app at phone width. The Expo app in mobile/ covers the dashboard, invoices, payments received, expenses and reports.

Run the mobile app

```
cd mobile
npm install
cp .env.example .env      # set SERVER_URL to your backend
npx expo start            # press w for web, or scan the QR with Expo Go
```

On a phone, `SERVER_URL` must be reachable from the phone (your computer's LAN IP or your https domain); the Android emulator uses `http://10.0.2.2:3480`. For store builds set `APP_NAME`, `BUNDLE_ID` and `EAS_PROJECT_ID` and use EAS. **Native Android and iOS builds have not been run**: test them on your own devices before publishing.

10. Going live checklist

1. Set `APP_URL`, `TRUST_PROXY=true` and a long `APP_SECRET`.
2. Set up SMTP in Admin → Settings → Email and send a test; check the email log.
3. Delete the sample companies and users (or at least change their passwords).
4. Admin → Settings: your brand name, colours, logo, sign-ups open or closed, SEO, cookie notice.
5. Create your company with its country, currency, tax regime, GST state and financial year start.
6. Have your accountant check the tax rates and settings.
7. Enter opening balances as a manual journal entry.
8. Activate the licence in Admin → Licence.
9. On sleeping hosting, add the daily `npm run recurring` cron job. Set up backups.

Tests

```
npm test      # 19 unit tests + 11 API tests on a throwaway SQLite database
bash test/api-smoke.sh   # quick checks against a running server with the sample data
```

11. What it does not do

- Not approved by an accountant and not certified for any country's tax rules; no return filing.
- No GST e-invoicing (IRN/QR from the GST portal), e-way bills, TDS/TCS, reverse charge, composition scheme, exemptions or zero-rated exports under LUT.
- No credit notes or debit notes (void and re-issue instead).
- One currency per company: no foreign-currency invoices or exchange gains and losses.
- No inventory quantities or stock valuation; no bank feeds or statement import; no automatic year-end close (post it as a journal entry).
- Payments are only records of money that moved outside the app. No payment gateway, no wallet, no automatic payment matching.
- Native Android and iOS builds of the mobile app have not been run. Our tests ran on SQLite.

Troubleshooting

Problem	What to do
"Could not start on port 3480: the port is already in use"	Change `PORT` in `.env` or stop the other program. Hostinger and cPanel set PORT for you.
Invite links point to localhost	Set `APP_URL` to your https address and restart.
No emails arrive	Check SMTP in Admin → Settings; every email is still saved in Admin → Email log.
Cannot create invoices on the live domain	Activate the licence in Admin → Licence.

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