GetLicit, Publisher User Guide

Entitlement-centric protected-PDF delivery. Buyer pays on Gumroad, gets a magic-link email, then views the PDF in-browser or downloads a watermarked copy. No raw file URL is ever exposed; there's nothing to crack because there's no DRM engine, just a server that checks who paid.

1. What you need before starting

- A Gumroad product (the PDF you're selling) and its **Gumroad access token** (Settings → Advanced → Applications, generate a personal access token) and **product ID** (visible in the product's edit URL).
- An SMTP account to send magic-link emails (Gmail app password, Postmark, SES, whatever you already have).
- A server to run this on (Docker-ready, see Section 5 for Helsinki-style VPS deploy).

2. First-time setup

Copy `.env.example` to `.env` and fill in:

```
PORT=3000
TOKEN_SECRET=<long random string>
PUBLISHER_SESSION_SECRET=<different long random string>
DB_PATH=./data/app.db
UPLOAD_DIR=./data/uploads
SMTP_HOST=<your smtp host>
SMTP_PORT=587
SMTP_USER=<smtp username>
SMTP_PASS=<smtp password>
MAIL_FROM=noreply@yourdomain.com
APP_BASE_URL=https://yourdomain.com
```

`TOKEN_SECRET` and `PUBLISHER_SESSION_SECRET` must be two **different** long random strings. They sign entitlement tokens and publisher login sessions respectively. Generate with `openssl rand -hex 32` or similar. Never reuse across environments.

Install and run:

```
npm install
npm run build
npm start
```

Or via Docker (see Section 5).

3. Creating your publisher account and uploading a PDF

All of this is done via HTTP calls. There's no admin UI in this MVP, just the API. Use curl, Postman, or a simple script.

Sign up:

```
curl -X POST https://yourdomain.com/dashboard/signup \
  -H "Content-Type: application/json" \
  -d '{
    "email": "you@yourdomain.com",
    "password": "a-strong-password",
    "gumroadProductId": "your_gumroad_product_id",
    "gumroadAccessToken": "your_gumroad_access_token"
  }'
```

Log in (returns a `publisher_session` cookie you'll reuse for the next calls, save cookies to a file with curl's `-c` / `-b` flags):

```
curl -X POST https://yourdomain.com/dashboard/login \
  -H "Content-Type: application/json" \
  -c cookies.txt \
  -d '{"email": "you@yourdomain.com", "password": "a-strong-password"}'
```

Upload the PDF you're selling:

```
curl -X POST https://yourdomain.com/dashboard/upload \
  -b cookies.txt \
  -F "pdf=@/path/to/your-course.pdf"
```

Response gives you `assetId`. One publisher can upload multiple assets, but the Gumroad webhook (next section) always attaches new sales to the **most recently uploaded asset** for that publisher. If you sell more than one product, run this MVP once per product, or treat asset management as a manual follow-up (not built yet, see Section 7 open items).

Check entitlements + usage for your assets any time:

```
curl https://yourdomain.com/dashboard/entitlements -b cookies.txt
```

4. Wiring up Gumroad

In your Gumroad product's **Settings** → **Advanced** → **Ping (webhook)**, set the webhook URL to:

```
https://yourdomain.com/webhooks/gumroad
```

Gumroad pings this on every sale with a `sale_id`. The server does **not** trust the webhook body for identity. It calls back to Gumroad's own API using your access token to confirm the sale is real, unrefunded, and for the right product, before creating an entitlement. This means:

- A refunded sale is rejected (`400`), no entitlement is created.
- A sale for a different product than you registered is rejected.
- Re-pinging the same `sale_id` is safe. Entitlements are idempotent, the buyer gets the same magic link, not a second one.

On a confirmed sale, the buyer's email automatically receives a magic-link email pointing at their protected view URL.

5. Buyer experience (what your customer sees)

1. Buyer purchases on Gumroad, checks their email.
2. Magic-link email arrives with a URL like `https://yourdomain.com/view/<signed-token>` .
3. Opening that link renders the PDF in-browser via pdf.js, no download, no raw file URL ever appears in the page source or network tab.
4. A "Download" button on that page fetches `https://yourdomain.com/view/<signed-token>/download` , which returns a **freshly watermarked** copy (buyer's email plus Gumroad sale ID stamped on every page), never the original file.
5. Every view and download is logged (IP, user-agent, timestamp) and visible to you via `/dashboard/entitlements` .

If a link is invalid, expired, or revoked, both routes return `404` , never a stack trace or a hint about why.

6. GetLicit's own pricing tier checkout links

`GUMROAD_STARTER_URL` , `GUMROAD_PRO_URL` , and `GUMROAD_BUSINESS_URL` point the landing page's buy buttons at the three Gumroad products for GetLicit's own Starter, Pro, and Business tiers. Create these three products in your own Gumroad account and set each URL in `.env` . This is separate from each publisher's own connected Gumroad product (`gumroadProductId` from Section 3), which is what that publisher's buyers purchase from.

7. Deploying (Docker / Helsinki VPS)

```
docker compose up -d --build
```

`docker-compose.yml` mounts `./data` for the SQLite DB and uploaded PDFs and reads secrets from `.env` . Make sure both exist next to the compose file on the server before first run. The container exposes port `3000` ; put your existing nginx/Cloudflare in front of it same as your other Helsinki services.

8. Known limits (MVP scope, not bugs)

- No admin UI. Dashboard is API-only (curl/Postman/script).

- One "current asset" per publisher for webhook attachment. No per-product routing yet if you sell multiple PDFs from one publisher account.
- No offline/grace-period license caching, no watermark field customization, no multi-file-type support (PDF only). All explicitly deferred per the original design spec.
- Entitlements never expire on their own; revocation (refund handling) is automatic, manual revocation is not yet exposed via API.