Woodpecker Beats — Step■by■Step Implementation Roadmap

A detailed guide to building a beat■selling platform with Next.js

Scope

This document provides a practical, ordered plan for delivering a producer■friendly beat marketplace with secure checkout, licensed downloads, and dashboards. It is opinionated toward Next.js + TypeScript + Stripe + S3/Cloudinary, but you can swap equivalents.

Outcomes

- Ship a production■ready MVP for selling beats.
- Support secure payments, licensed downloads, and email receipts.
- Give the producer a simple upload flow and sales dashboard.
- Lay foundations for SEO, analytics, and later growth features.

1) Goals & Non■Goals

- Goal: Fast, mobile■first storefront with discoverability (search/filters).
- Goal: Simple, reliable checkout using Stripe Checkout + webhooks.
- Goal: License■aware fulfillment (lease vs exclusive).
- Goal: Secure, expiring download links (no public asset leakage).
- Non■Goal: Complex multi■producer marketplace (start single■seller).
- Non■Goal: Full social features, messaging, or subscriptions at MVP.

Definition of Done (section): Goals are documented in repo README and reflected in issues/labels.

2) Architecture Overview

Stack: Next.js (App Router), TypeScript, Tailwind, shadcn/ui, TanStack Query (client data fetching), Prisma or Mongoose (DB), Stripe (payments), Cloudinary/S3 (assets), Resend/Postmark (email).

- Next.js Server Actions or route handlers for secure ops (uploads, webhooks).
- DB: Start with PostgreSQL + Prisma *or* MongoDB + Mongoose — pick one and stay consistent.
- File storage: S3/Cloudflare R2/Cloudinary. Store only URLs in DB.
- Audio handling: store *preview* MP3 (watermarked/shortened) and *deliverable* WAV/MP3/ZIP.
- Webhooks: Stripe → your /api/webhooks/stripe to create Orders, generate download links, send emails.

Definition of Done: Architecture diagram and decisions captured in ADRs (/docs/).

3) Project Setup

1. Create Next.js app with TypeScript and App Router. Add Tailwind and shadcn/ui.
2. Initialize repo, add commit hooks (lint■staged, eslint, prettier).
3. Choose DB (Postgres+Prisma or Mongo+Mongoose). Create a single 'Beat' model and 'Order' model.
4. Configure environment variables (.env.local.example).
5. Pick storage provider (S3/Cloudinary). Create bucket/folder structure (e.g., beats/previews, beats/masters, artwork).
6. Install Stripe SDK and create a test product in Dashboard to confirm connectivity.

Definition of Done: App boots locally, lint passes, commit hook works, sample envs exist.

4) Data Model

Start minimal; you can extend later. Suggested fields:

- Beat: id, title, description, genre, bpm, key, tags[], artworkUrl, previewUrl (mp3), masterUrl (zip/wav), price: { lease, exclusive }, isPublished, createdAt.
- Order: id, userEmail, beatId, licenseType (lease|exclusive), amount, currency, status, downloadUrl (signed), expiresAt, createdAt.

Definition of Done: Migrations applied; seed script creates 3 sample beats.

5) Authentication

1. NextAuth (email/Google). Producer = admin role (via allowlist or role field).
2. Protect upload/dashboard routes with middleware. Public store stays open.
3. Customer can checkout without account (Stripe Checkout), but create account post■purchase for library access.

Definition of Done: Admin can sign in; protected routes redirect when unauthenticated.

6) Storage & Media Processing

1. Use presigned uploads direct from browser to S3/Cloudinary; store returned URLs.
2. Keep two audio assets per beat: preview (30–60s, watermarked/low bitrate) and master (deliverable).
3. Optional watermark: layer a subtle tag every 10s in the preview. If not, truncate to 30–45s.
4. Folder strategy: artwork/, previews/, masters/. Use predictable keys (beatId-filename.ext).
5. Implement a server action to generate a signed URL for downloads with short TTL (e.g., 15–60 min).

Definition of Done: Upload form can push to storage and returns usable URLs.

7) Producer Upload Flow

1. Form fields: title, genre, bpm, key, tags, shortDesc, longDesc, artwork, preview mp3, master zip/wav, pricing (lease/exclusive), publish toggle.
2. Client■side validation (file types, sizes). Server■side validation on API.
3. On submit: create Beat in DB, then redirect to Beat detail in dashboard.
4. Add status indicators: uploading, processing, saved, error.
5. Allow draft saves before publishing.

Definition of Done: Producer can create, edit, publish/unpublish a beat.

8) Public Storefront

1. Home: hero, featured beats, genres, search bar.
2. Beats grid: pagination or infinite scroll; filters for genre, bpm range, price, tags.
3. Beat card: artwork, title, price (lease/exclusive), quick play preview, 'Buy' button.
4. Beat detail page: waveform player (e.g., Wavesurfer), metadata, license table, related beats.
5. Global audio player: persistent mini■player between routes (optional for MVP).

Definition of Done: A user can discover beats, play previews, and navigate quickly.

9) Checkout & Payments

1. Create Stripe Products/Prices programmatically per beat & license or generate dynamically at checkout.
2. From 'Buy' → call server action to create Checkout Session with line items (license type).
3. Success URL returns session_id; store a pending Order (status='processing').
4. Implement /api/webhooks/stripe to verify signature and mark Order as 'paid' on checkout.session.completed.
5. Enable tax, currency, and email collection in Checkout as needed.

Definition of Done: Test cards complete purchase; Order becomes paid in DB.

10) Fulfillment (Downloads & Licensing)

1. On paid event: generate a signed download URL for the master asset and store it on the Order with expiry.
2. Email receipt with license terms and download link (expiring).
3. Customer Library page: list past purchases with fresh signed links (re■sign on demand).
4. Block exclusive purchases if already sold as exclusive (or after exclusive sale, disable leases).
5. Optional: License PDF generator (insert buyer name, beat title, license terms).

Definition of Done: After payment, customer can securely download the master within minutes.

11) Dashboards

1. Producer Dashboard: stats (sales, revenue), recent orders, quick links to upload/edit.
2. Beat Management: table with publish toggle, price edits, delete/unlist.
3. Customer Dashboard: purchases, re■download, invoices/receipts.
4. Use TanStack Query for snappy data fetching and caching.

Definition of Done: Producer can manage catalog; customers can access purchases.

12) Admin Console

1. Feature flags (toggle experimental UI elements).
2. Order lookup & refund action (manual).
3. Content moderation (unpublish, replace files).
4. Webhook events log viewer for debugging.

Definition of Done: You can diagnose failed webhooks and fix catalog issues quickly.

13) Observability & Analytics

1. Add basic server logging for uploads, checkout creation, webhook handling.
2. Client analytics: page views, play clicks, add■to■cart/checkout clicks.
3. Funnel tracking: preview → checkout → success; track abandonment reasons.
4. Error monitoring (Sentry) and performance metrics (LCP, CLS, TTFB).

Definition of Done: You can answer: where do users drop off? what breaks most often?

14) Security & Compliance

1. Never expose master file URLs publicly. Always use signed URLs with short TTL.
2. Validate and sanitize all file uploads (MIME, size caps).
3. Verify Stripe webhook signatures and use idempotency keys.
4. Rate limit API routes (e.g., uploads, checkout creation).
5. Privacy/Terms pages; include license terms on product pages and email receipts.

Definition of Done: Basic OWASP checks pass; secrets stored via environment variables.

15) SEO & Marketing

1. Semantic pages with metadata (OG tags), clean URLs (/beats/{slug}).
2. Public sitemap.xml and robots.txt; server■side render catalog pages.
3. Collections (e.g., 'Dark Trap', 'Lo■Fi Focus'): internal linking for discovery.
4. Email capture (newsletter) and post■purchase follow■up.
5. Rich snippets: product schema (price, availability).

Definition of Done: Lighthouse SEO ≥ 90 on key pages; sitemap indexed.

16) Brand & Content

1. Name, logo, color system (ensure contrast).
2. Thumbnail template for beats (consistent artwork style).
3. Copywriting: license terms, FAQs, refund policy, DMCA notice.
4. Demo playlist embedded; social links and contact form.

Definition of Done: Brand kit in /public/brand; FAQ answers common objections.

17) Testing Strategy

1. Unit tests for utils (price calc, signing URLs).
2. Integration tests for API routes (checkout session, webhooks with fixtures).
3. E2E tests (Playwright) for upload → publish → buy → download flow.
4. Accessibility checks (axe) and keyboard navigation in player.

Definition of Done: CI runs tests on PR; critical flows covered.

18) CI/CD & Deployment

1. Vercel project set up with env vars; preview deployments on PR.
2. CI: lint, typecheck, test; block merge on failures.
3. Prod: manual promotion from main; run DB migrations via script.
4. Backup policy for DB and storage; restore drill documented.

Definition of Done: Green pipeline; rollbacks documented.

19) Launch Checklist

- Custom domain with HTTPS; 'www' redirect configured.
- Legal: Terms, Privacy, License terms; contact and DMCA email.
- Stripe in live mode; real card test purchase completed and refunded.
- 404/500 pages, rate limit in place, error monitoring connected.
- Seeded with at least 10 published beats and 3 collections.

Definition of Done: A stranger can discover, preview, buy, and download within 2 minutes.

20) Future Enhancements (Backlog)

- Coupons & limited■time bundles.
- Artist accounts / multi■seller marketplace.
- Exclusive licensing workflow with contract e■signature.
- Subscriptions (download credits per month).
- AI■powered search by vibe/BPM, similar■beat recommendations.
- Blog/tutorials for SEO and community building.

Thanks for building Woodpecker Beats. Ship small, ship often. ■