**✅ Recommended baseline: TypeScript + Next.js (App Router) monolith on PostgreSQL**

**Why this as the default**

* **One codebase** for public marketing site, public campaign pages (SSR/SEO), portal, admin, and API routes—fewer moving parts.
* **First‑class Stripe, Twilio, Cloudinary, Mux, Typesense, Clerk** SDKs and examples.
* **Edge + serverless** friendly (rate limits, webhooks, OG images).
* Matures into services later if you need, but you won’t need microservices to hit MVP.

**Core components**

* **Frontend & server**: Next.js (App Router), React, TypeScript, Tailwind (with your design tokens).
* **Auth**: Clerk (email/phone/MFA/social + server helpers) + Turnstile for anti‑bot.
* **DB & ORM**: PostgreSQL (Neon or Supabase) + Prisma (schema & migrations).
* **Queue/Jobs**: Trigger.dev or Inngest for nudges, payout timers, digests.
* **Search**: Typesense Cloud (campaigns, KB, discovery facets).
* **Realtime**: Pusher or Ably (DM, live progress, team leaderboard).
* **Media**: Cloudinary (images) + Mux (video); Sightengine for moderation.
* **Payments**: Stripe Payments + Connect Express + Identity (KYC/SSN last‑4).
* **Email/SMS/Voice**: Postmark (or SendGrid), Twilio.
* **Feature flags**: GrowthBook (or Flagsmith).
* **Analytics & error monitoring**: PostHog + Sentry.
* **Admin**: react‑admin + CASL (RBAC) as a sub‑app in /admin.
* **CMS/KB**: Docusaurus or GitBook for KB; your marketing site is CMS‑driven per Section 9.

**Hosting**: Vercel (app) + Neon/Supabase (DB) + Typesense Cloud + Cloudinary + Mux + Pusher/Ably + Postmark + Twilio.  
**Cost**: All have free/low starter tiers; with early traffic you’ll be comfortably **≤ $150/mo** combined.

**How it maps to your 25 sections (at a glance)**

| **Area** | **Stack & service** |
| --- | --- |
| Registration, security, device/session limits | Clerk (auth), Turnstile (bot), Twilio Verify (OTP) |
| Verification (DL/ID, SSN last‑4, EIN) | Stripe Identity + Connect onboarding |
| Campaign creation wizard (rich text, gallery) | Next.js forms + Cloudinary (images) + Mux (video) + Sightengine (moderation) |
| Rewards & fulfillment (tiers, nudges) | Prisma models + Trigger.dev jobs + Postmark/Twilio notifications |
| Donations/checkout (tips, cover fees, AoN banners) | Stripe Elements/Checkout + webhooks; OG images via @vercel/og |
| Team fundraising (roles, invites, leaderboard) | Prisma (team tables) + Pusher/Ably realtime |
| DM & inbox | Pusher/Ably channels + Postgres + Next.js server actions |
| Search & discovery | Typesense (facets, ranking) + SSR lists |
| Notifications matrix | Postmark + Twilio + in‑app bell; quiet hours via Trigger.dev scheduling |
| Wallet & payouts | Stripe Connect Express + scheduler (Trigger.dev) + finance ledger tables |
| Admin console (Trust & Safety, Finance Ops) | react‑admin + CASL + GrowthBook flags + Sentry |
| CMS/Marketing/KB | Next.js pages + Docusaurus/GitBook; OG images via @vercel/og |
| Support & helpdesk | Freshdesk or Help Scout + portal “My tickets” + DM→ticket bridge |
| Analytics | PostHog product analytics + Plausible (optional) + Sentry errors |

**Two good alternatives (if your team prefers)**

**Option B — Laravel + Livewire + Blade (PHP)**

* **Stripe Cashier** for payments; **Laravel Scout + Meilisearch/Typesense** for search; **Horizon + Redis** for queues; **Filament** or **Laravel Nova** for admin; **Pusher** for realtime.
* **Pros**: Super‑productive backend, excellent first‑party queue/mail/auth, Filament admin is a joy.
* **Cons**: Most third‑party examples and SDK snippets in this project skew JS; you’ll hand‑roll some integrations.

**Option C — Django + HTMX/React (Python)**

* **Django admin** (very capable out of the box), **DRF** for APIs, **Celery + Redis** for jobs, **Wagtail** for CMS, **Channels** for realtime.
* **Pros**: Batteries included, robust admin, great for complex back‑office rules.
* **Cons**: You’ll still wire Stripe Connect/Identity and realtime SDKs; fewer off‑the‑shelf UI kits than React.

If you’re undecided, choose **Next.js**—it aligns best with the modules you want and keeps onboarding wide‑talent friendly.

**Decision matrix (quick)**

| **Criterion** | **Next.js + TS** | **Laravel** | **Django** |
| --- | --- | --- | --- |
| Drop‑in SDK coverage (Stripe/Twilio/Cloudinary/Mux/Typesense/Clerk) | **Excellent** | Good | Good |
| SSR/SEO & OG images for campaigns | **Excellent** | Good | Good |
| Admin scaffolding (low code) | Good (react‑admin) | **Excellent (Filament)** | **Excellent (Django admin)** |
| Real‑time DM & live counters | **Excellent** | Excellent | Good (Channels) |
| Hiring pool (full‑stack) | **Very wide** | Wide | Wide |
| Time to MVP (your scope) | **Fastest** | Fast | Fast |
| Path to services later | **Easy** | Easy | Easy |

**Repo layout (recommended)**

/apps

/web # Next.js app (marketing + public campaign + portal + /admin)

/jobs # Trigger.dev/Inngest workflows (cron, webhooks, nudges)

/cms # (optional) Docusaurus/GitBook KB source

/packages

/ui # Shared UI components (shadcn/tailwind + a11y wrappers)

/db # Prisma schema + migrations + seed scripts

/config # Design tokens, ESLint, TS configs

**Day‑one wiring plan (you can execute this sequentially)**

**1) Bootstrap infra**

* Create repos, CI/CD (Vercel), Postgres (Neon or Supabase), Typesense Cloud, Cloudinary, Mux, Clerk, Stripe, Postmark, Twilio.
* Add env management (.env.local + Vercel env).
* Install Tailwind + your design tokens.

**2) Auth & session**

* Wire **Clerk** (email/phone, passkeys, social).
* Enforce single active session logic on middleware (store session fingerprints in DB).

**3) Database & ORM**

* Implement Prisma models for **User**, **Organizer**, **Campaign**, **Team**, **Donation**, **RewardTier**, **RewardClaim**, **Payout**, **Ledger**, **DM** (as specced).
* Seed a few categories/tags.

**4) Media**

* Drop **Cloudinary** upload widget + server signature endpoints.
* Add **Sightengine** async moderation on upload → flag/unpublish on hit.
* Wire **Mux** direct uploads for campaign videos.

**5) Payments**

* Stripe Elements for on‑page donate (or Checkout session if you prefer).
* Server: Payment intent creation, recording donations, webhook handlers (payment succeeded/refund/dispute).
* **Connect Express** onboarding for organizers; **Identity** verification hook to unlock publish.

**6) Search**

* ETL to **Typesense** on publish/unpublish/donation events; implement /search page with facets & sorts per §16.

**7) Notifications**

* Postmark transactional templates; Twilio SMS (OTP, payout returned) + voice fallback.
* In‑app bell store (Postgres) + quiet hours scheduling via Trigger.dev.

**8) DM**

* Pusher/Ably channels; Next.js server actions to send/receive; attachment pipeline → Cloudinary → AV scan.

**9) Payouts**

* Implement eligibility checks; create scheduler job at **end+hold**; **transfer** to connected account; capture payout webhooks; statements.

**10) Admin & moderation**

* /admin with react‑admin data provider; RBAC with CASL; views for Users/Campaigns/Donations/Payouts/Reports/Templates/Flags.
* Feature flags via GrowthBook (env‑toggled).

**11) CMS & KB**

* Docusaurus/GitBook for KB; link “Still need help?” → helpdesk form (Freshdesk/Help Scout).
* Marketing pages from your CMS plan in Section 9.

**12) Analytics & QA**

* PostHog events from our analytics map; Sentry error capture; Lighthouse budgets; a11y test sweep.

**A note on cost control (≤ $150/mo early)**

* **Cloudinary/Mux/Typesense/Pusher/PostHog/Sentry/Trigger.dev** all have generous free or low starter tiers; with modest early traffic, you’ll typically pay only **Postmark + minimal Twilio usage** and perhaps **one or two small “$20–$50” line items**.
* If any single tool creeps up, you can swap (e.g., Pusher ↔ Ably ↔ Supabase Realtime) without changing your overall architecture.

**What I’d lock now (to avoid rework later)**

* **Stripe Connect Express + Identity** (commit to this; it cascades into verification, payouts, taxes).
* **Typesense** for search (schema + indexing from day 1).
* **Prisma** schema aligned to the data model we already defined (so Admin & Analytics “just work”).
* **Trigger.dev/Inngest** for schedules & nudges (you have many timed workflows).