Nano Banana Sticker/Avatar Generator — A → Z Launch Plan

> A complete, actionable blueprint to build a viral Sticker/Avatar generator using Nano Banana (Google AI Studio Build). Designed for a fast MVP you can ship in 2 weekends and grow into a paid product.

Table of contents

1. Vision & Value Proposition
2. User flows (3 core flows)
3. MVP feature list (must-have) + Phase 2 features
4. Tech stack (frontend, backend, infra, third-party)
5. Project folder structure (frontend + backend + infra)
6. Data models & APIs (endpoints with payload examples)
7. Integration with Nano Banana (Google AI Studio Build)
8. Image processing, export, and sticker pack generation
9. Auth, billing, and subscription flow
10. Security, privacy & legal considerations
11. Analytics, monitoring, and cost control
12. Deployment & CI/CD
13. Marketing & Viral Launch Strategy (step-by-step)
14. 2-week MVP timeline and task breakdown
15. Pricing + Monetization experiments
16. Playbook: growth hacks & community seeding
17. Sample prompts (for Nano Banana & WindSurf)
18. Appendix: Telegram/WhatsApp packaging & export details

---

1) Vision & Value Proposition

Product: Web app that transforms user photos into Nano Banana-style stickers/avatars they can download and share.

Core value: Instant, delightful, and shareable identity packs — users get expressive stickers that spread virally via chat apps.

Who pays? Casual users who want unique stickers, streamers/creators, small brands, and communities.

Success metric (MVP): 10k sticker packs shared within first month or 1k paid conversions in 3 months.

---

2) User flows

Flow A — Quick Sticker Pack (Primary viral loop)

1. User lands on homepage.
2. Upload a selfie (or drag & drop) or take with webcam.
3. Choose a pack style (Retro, Gaming, Anime, Miniature).
4. Click "Generate 10 stickers". App calls Nano Banana API for batch generation.
5. Preview results; user can reorder, delete, or regenerate specific stickers.
6. User downloads free pack (5 free) or pays to unlock full 10+ pack and HD/no-watermark.
7. User shares sticker to WhatsApp/Telegram/Instagram. Viral loop triggers.

Flow B — Create & Customize

1. User chooses advanced options: background removal/replace, color palettes, props.
2. Fine-tune prompts (or use presets).
3. Save template for later.

Flow C — Creator/Brand Upload

1. Creator uploads many faces.
2. Creates batch packs with a brand tag.
3. Bulk export & licensing (paid)

---

3) MVP vs Phase 2

MVP (ship fast; must-have)

- Upload image (webcam support optional)
- 3 styles (Retro, Miniature, Cartoon)
- Generate 5 free stickers + option to unlock 10 (paid)
- Preview & download (zip or single webp/png)
- Basic user accounts (email/social login)
- Payments (Stripe) and simple subscription
- Share buttons for WhatsApp, Telegram, Instagram
- Rate limits & basic caching
- Basic analytics (Mixpanel or Plausible) and Sentry for errors

Phase 2 (scale & differentiate)

- Advanced style packs (Gaming, Anime)
- Editable stickers (text/bubble overlays)
- Sticker store / marketplace for creators
- Team/Brand accounts & licensing
- API for third-party apps
- Mobile PWA with “install” support

- Bulk/batch generation and webhook callbacks
- Learning/Recommendation engine for style suggestions

---

4) Tech Stack

Frontend

- React (Vite or Next.js if SSR needed)
- Tailwind CSS for rapid UI
- Zustand or Redux for local state
- Firebase Auth or Auth0 (or custom JWT via backend)
- Dropzone for uploads

Backend

- Node.js + Express (or Fastify)
- TypeScript (you asked JS earlier but TS improves reliability; convert to JS if you prefer)
- Image processing: Sharp
- Queue: BullMQ + Redis for async generation
- Storage: AWS S3 or Firebase Storage
- Database: PostgreSQL (managed, e.g., Supabase) or MongoDB Atlas
- Payments: Stripe

AI & 3rd party

- Google AI Studio (Nano Banana) — invoked via their Build flow or REST endpoints (if available)
- Optional: OpenAI (LLM) to convert text options into structured prompts

Infra & DevOps

- Hosting: Vercel (frontend) + Render/Heroku/DigitalOcean App Platform (backend) or AWS ECS/EKS
- CDN: Cloudflare
- CI: GitHub Actions
- Observability: Sentry, LogRocket (frontend), PostHog/Mixpanel, Prometheus/Grafana (if needed)

---

5) Project folder structure

Monorepo approach (recommended)

```
/nano-banana-stickers
├─ apps/
│  ├─ web/               # React app (Next.js/Vite)
│  └─ api/               # Node/Express backend (server)
├─ packages/
```

```
│  ├─ ui/                # shared React components
│  ├─ lib/               # shared utilities
│  └─ types/             # shared types (if using TS)
├─ infra/                # terraform / k8s manifests / deployment scripts
├─ scripts/
├─ .github/workflows/
└─ README.md
```

Frontend (web)

```
/web
├─ public/
├─ src/
│  ├─ components/
│  ├─ hooks/
│  ├─ pages/ (or routes/)
│  ├─ styles/
│  └─ utils/
├─ package.json
└─ vite.config.js
```

Backend (api)

```
/api
├─ src/
│  ├─ controllers/
│  ├─ services/
│  ├─ jobs/            # background processing for generation
│  ├─ routes/
│  ├─ models/
│  ├─ utils/
│  └─ index.js
├─ package.json
└─ Dockerfile
```

---

6) Data models & API endpoints

Core models (simplified)

- **User**: id, name, email, stripeCustomerId, plan, createdAt
- **ImageJob**: id, userId, inputImageUrl, prompts[], status (pending/processing/done/failed), results[]
- **StickerPack**: id, userId, jobId, packName, files[], price

Important APIs

POST /api/upload - Purpose: accept image, store in S3, return private URL - Payload: multipart/form-data (file) - Response: { imageUrl }

POST /api/generate - Purpose: create a generation job - Payload: { userId, imageUrl, style, options } - Response: { jobId }

GET /api/job/:id - Purpose: job status & results - Response: { status, results: [{url, thumbUrl, meta}] }

POST /api/checkout - Purpose: create Stripe session for pack purchase - Payload: { userId, packId } - Response: { sessionUrl }

POST /webhook/ai-callback - Purpose: (if Nano Banana supports callbacks) receive generation complete notifications

---

7) Integrating with Nano Banana (Google AI Studio Build)

> NOTE: Google's UI Build flow (Nano Banana) might be consumable via API/REST — if it exposes endpoints, use them. Otherwise, you can automate using a server-side integration supported by Google Cloud.

Two integration modes

1. **Direct API calls (preferred)**
2. Create an account on Google AI Studio.
3. Create a Nano Banana model instance or project in Build.
4. Use their REST endpoint to send an image + prompt and receive an image back.
5. Respect rate limits; use an async job queue to avoid blocking HTTP responses.
6. **Indirect / Manual (if API not publicly available)**
7. Use a server-side headless workflow where you manually call an internal endpoint (not ideal). Try to get an enterprise/integration access with Google.
8. Alternatively, use a stock image-to-image model available via other providers temporarily to validate MVP.

Prompt engineering for consistency

- Use templated prompts. Example template:

```
"Transform the person in this photo into a collectible miniature figurine
in Nano Banana style: high contrast, soft plastic look, glossy eyes,
simplified features, 512x512, transparent background. Keep facial identity
and skin tone natural. Include props: <PROP>. Style variation: <STYLE>"
```

- Keep seeds/randomness stable for consistent packs. Offer a "regenerate" with new seed.

---

8) Image processing, export, and sticker pack generation

Processing pipeline

1. Upload -> store original in secure bucket
2. Generate job created -> push to queue
3. Worker pulls job -> calls Nano Banana API for N variations
4. Worker receives images -> post-process using Sharp (resize, trim, convert to webp, apply transparent bg)
5. Store outputs in S3; create ZIP and web-friendly preview.

Export formats

- WebP (animated optional) for WhatsApp/Telegram
- PNG with transparent background
- ZIP for multi-file download
- Optional: APNG/GIF for animated stickers (advanced)

Packaging for WhatsApp & Telegram

- Telegram: webp stickers with 512x512 and proper metadata. Provide guide for users to add to @stickers bot.
- WhatsApp: Use webp or image crop guidelines; for easy sharing provide an image pack and step-by-step instructions.

---

9) Auth, billing, subscription flow

Auth

- Firebase Auth (Google, Apple, email) for fast implementation.
- Use JWT from backend for API requests.

Billing options

- **Freemium**: 5 free stickers per month (or per account)
- **One-shot packs**: $1.99 per 10-pack (Stripe Checkout)
- **Monthly**: $7/month unlimited small packs (or 100/month)
- **Creator / Brand**: Custom pricing for bulk/license

Flow

1. User wants full pack -> click "Unlock"
2. Create Stripe session -> redirect to Checkout
3. On success webhook from Stripe -> mark user entitlement
4. Allow downloads

---

10) Security, privacy & legal

- **Face data** is sensitive. Add clear privacy policy: we do not store images longer than X days unless user saves them.
- Offer option to **delete images** manually. Auto-delete originals after 7–30 days if not claimed.
- Add consent checkbox: "I agree to processing my photo for sticker generation."
- GDPR: allow data export & deletion.
- Terms: do not allow illegal/NSFW content.

---

11) Analytics, monitoring & cost control

- **Analytics**: PostHog/Mixpanel for events: upload, generate, share, purchase, install sticker pack
- **Monitoring**: Sentry for errors; server metrics in Datadog or simple CloudWatch dashboards
- **Cost control**:
- Use cache for repeated generations of same image+style (hash prompt+image)
- Throttle free users aggressively
- Use image size limits and downscale before sending to AI to reduce cost

---

12) Deployment & CI/CD

- Frontend: Vercel (easy with Next.js). Use preview branches for PRs.
- Backend: Render / Heroku / AWS App Runner. Containerize with Docker.
- CI: GitHub Actions to run lint/test/build and deploy.
- Secrets: store in platform env (Vercel/Render/AWS Secrets Manager)

---

13) Marketing & Viral Launch Strategy (step-by-step)

Pre-launch (1–2 weeks before)

1. Build a single landing page with opt-in & waitlist (use Mailchimp or ConvertKit)
2. Create 10 demo packs showing diversity (Retro, Anime, Miniature)
3. Prepare a press kit: explainer video (20–30s), screenshots, FAQs, demo GIFs
4. Seed Telegram/WhatsApp groups and Discord servers with private demo invites
5. Reach out to 20 micro-influencers (1–10k followers) with free VIP codes

Launch (day 0–7)

1. Announce on Product Hunt (prepare assets, maker comments, and hunt day schedule)
2. Run a referral program: invite a friend -> both get 5 free stickers
3. Viral CTA: "Create a sticker and send it to 5 friends to unlock a free premium pack"
4. Run a hashtag challenge on Instagram/Twitter: #MyNanoBanana
5. Post short reels showing before/after and how to add stickers to WhatsApp

Post-launch (week 2–8)

1. Add creator packs / marketplace to capture creators
2. Publish tutorial content (your YouTube channels + LinkedIn threads)
3. Paid ads (small budget) targeting creators and students
4. Integrations: create Zapier or Make automation to send generated packs to cloud storage or webhook

---

14) 2-week MVP timeline (detailed)

Week 0 — setup (weekend 1)

Day 1 (Sat) - Project scaffold (monorepo), create GitHub repo - Initialize frontend (Next.js + Tailwind) and backend (Express) - Set up Stripe, S3 (or Firebase Storage), Redis local - Create basic landing page + upload UI

Day 2 (Sun) - Implement file upload -> S3 flow - Implement backend job creation + queue worker skeleton - Stub Nano Banana integration (mock) so UI can call generate - Implement preview & basic download zip

Week 1 — polish & payments

Day 3–4 (Mon–Tue) - Integrate real Nano Banana API (or fallback model) - Implement image post-processing (Sharp) - Implement auth (Firebase Auth quickstart)

Day 5–6 (Wed–Thu) - Implement Stripe Checkout + webhook - Add free/paid gating logic - Implement basic caching

Day 7 (Fri) - Analytics & error monitoring (Mixpanel, Sentry) - Add share buttons & referral token generation

Day 8–9 (Weekend 2) - Polish UI, mobile responsiveness - Create marketing assets, demo packs and GIFs - Soft launch to small audience & fix issues

Day 10–11 (Following Mon–Tue) - Product Hunt prep & schedule - Outreach to micro-influencers

---

15) Pricing & Monetization experiments

- **Experiment A (low barrier):** $0.99 per 10-pack
- **Experiment B (subscription):** $4/month unlimited basic packs
- **Experiment C (creator revenue share):** enable creators to sell packs (70/30 split)

Measure: CAC, LTV, conversion rate from free -> paid, churn.

---

16) Growth hacks & community seeding

- Pre-made viral demo GIFs for WhatsApp statuses
- Partner with Telegram sticker communities for featured placement
- Referral: give both inviter & invitee extra premium sticker
- Tweetstorm of "before/after" threads and repurpose to LinkedIn
- Sponsor small Discord servers & Twitch streamers to use exclusive packs live

---

17) Sample prompts

A. Nano Banana prompt template (image-to-image)

```
"Convert the person in this image to a collectible miniature figure in the Nano
Banana style. Maintain facial identity and natural skin tone. Make the figure
look like a high-gloss designer toy with a slightly exaggerated head, glossy
eyes, simplified clothing detail, and clean silhouettes. Render with a
transparent background, 1024x1024, high detail, no text. Style: <STYLE_VARIANT>.
Keep result suitable for stickers (clear silhouette)."
```

B. Prompt variations for "Retro" style

```
"Nano Banana Retro: Use vintage color palette, film grain, saturated highlights,
stylized toy look, 80s pop vibe. Keep transparent background."
```

C. WindSurf prompt — scaffold the project (for AI IDE)

```
"Scaffold a full-stack web project named 'nano-banana-stickers' using a
monorepo. Frontend: Next.js + Tailwind, pages: landing, upload, preview,
account, checkout. Backend: Node.js + Express with endpoints /upload, /
generate, /job/:id, /checkout, Stripe integration and Redis + BullMQ queue. Use
AWS S3 for storage. Include a worker that calls a mock Nano Banana API
(replaceable). Add Dockerfiles and GitHub Actions workflow for CI. Provide
```

```
sample .env.example and README with local dev steps. Keep code in JavaScript and
keep modules small and well-documented."
```

---

18) Appendix: Telegram & WhatsApp packaging

Telegram sticker pack steps (user-facing)

1. Generate sticker (512x512 webp, transparent)
2. Visit @stickers bot on Telegram
3. /newpack → send pack name → upload images one by one
4. Add emoji for each sticker → publish and share link

WhatsApp

- WhatsApp requires stickers in webp with certain metadata; easiest for users is to use an app like "Personal Stickers for WhatsApp" to import a ZIP of webp images.

Quick checklist to launch (copy-paste)

- [ ] Repo + monorepo scaffold
- [ ] Frontend upload & preview
- [ ] Backend job + queue + worker
- [ ] Nano Banana API integration (or mock)
- [ ] Post-processing (Sharp) + storage
- [ ] Auth (Firebase) + Stripe + webhooks
- [ ] Download ZIP & single-download flow
- [ ] Share buttons + referral
- [ ] Analytics + Sentry
- [ ] Landing page + demo GIFs
- [ ] Product Hunt assets + outreach list

Final notes

Start with a *tiny vertical* — a single solid style (e.g., Miniature Toy) and the simplest flow: upload → generate → share. Nail the UX and sharing experience; virality will follow.

Good luck — want me to generate the **WindSurf scaffold prompt adapted to the exact files & commands** (or create the starter repo code)?