

The headless commerce engine for India

The Playbook

Everything about Bluenexus — what it is, how it works, how to sell with it. From first store to first order, headless or hosted, with Razorpay + COD.

Contents

What you'll find — and what you can skip if you're in a hurry.

01 — What is Nexus	p. 2
02 — Core concepts (Store, Product, Order, APIs)	p. 2
03 — Quick start: 90 seconds to live	p. 3
04 — Admin: every tab, every button	p. 4
05 — Headless: your site, our commerce	p. 5
06 — Payments & orders	p. 7
07 — Security & isolation	p. 8
08 — Operate: env, DB, deploy	p. 9
09 — API reference	p. 10
10 — Troubleshooting	p. 11

01 — What is Nexus

Shopify rebuilt you inside its walls. Themes, rent, slow. Nexus is the opposite: a **headless engine** for India. Your Next.js stays yours — marketing, journal, story. We add **products, variants, cart, Razorpay, COD, orders, inventory**. You call us via `fetch()` or use our hosted shop at `/shop/:subdomain`. Same data.

One engine, any vertical: apparel, jewellery, textiles, digital. Generic schema — **vendor • productType • tags • variants** — not hard-coded. Razorpay-native, COD-first, no Stripe.

ONE IDENTITY: ABSTRACT, DARK, INTENTIONAL

One form, one light, slow motion. No gradients, no stock orbs. Just a dark abstract surface that lets your brand stay front-and-center.

02 — Core concepts

Everything is a **Store**. You get one on register. It has a **subdomain** (*yourbrand.nexus.com*), **currency** (INR), **products** (isPublished), **variants** (price, stock, sku), **customers** (email per store), **orders** (PAID/PENDING/FAILED, fulfillment UNFULFILLED/SHIPPED), **discountCodes**, **images**.

Two APIs:

Storefront API

*Public • CORS * • /api/storefront/:subdomain/**

GET products, GET config, POST checkout, POST verify. No auth needed to read. Your site calls this.

Admin API

*Private • JWT cookie or x-admin-token • /api/products, /orders, /admin/**

Create products, read orders, fulfill. storeId-scoped — you never see another brand.

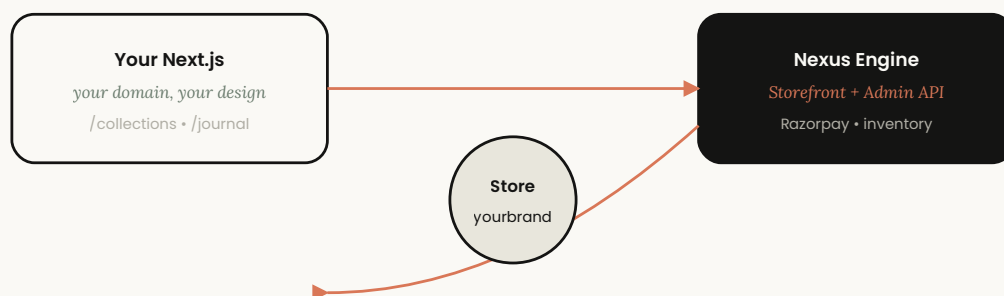


FIG. 01 — Two frontends, one Store. Headless or hosted, same products.

03 — Quick start: 90 seconds

1 Register

Go to **/register** → email + password (8+ chars, Zod). We create **Store{ subdomain: store-xxxx, currency: INR }** + OWNER user, set **JWT** httpOnly 24h.

2 Onboard

At **/onboarding** set **Store Name + subdomain** (live `GET /api/stores/check?subdomain=` 3-30 chars, a-z0-9-) + **currency**. One POST, you land in **/admin**.

3 Seed demo (optional)

GET /api/demo/seed creates **demo** store with 4 sample products if empty — 3/hour rate-limited, idempotent.

4 See it live

Hosted shop at **/shop/demo**, guided tour at **/demo** (9-step auto-play like video), integration guide at **/integrate**.

TRY — STOREFRONT READ (PUBLIC, NO AUTH)

```
curl https://bluenexus.vercel.app/api/storefront/demo/products  
→ { store: {name, subdomain, currency}, products: [{title, vendor, productType, variants:[{price}]}]}
```

04 — Admin: every tab

All under **/admin** — blocked by **middleware.ts** (jose verify, 302 to **/login** if no valid JWT). No tab leaks without login.

/admin

GET /api/admin/stats — revenue (PAID sum), orders, products

3 cards + placeholder chart. Revenue from paid orders only.

/admin/products & /admin/products/new — 200

GET /api/products/list, POST /api/products, PUT /api/products, DELETE /api/products/delete

Grid, “Add Product”, Edit. Form: title, vendor, productType, tags CSV, description (DOMPurify), options matrix → variants (price/stock/sku), images (R2 or base64 fallback), status Active/Draft. Zod 30 req/min.

/admin/orders & /admin/orders/[id]

GET /api/orders, POST /api/orders/fulfill, POST /api/orders/cancel

PAID/PENDING/FAILED, fulfillment UNFULFILLED/SHIPPED. Fulfill adds tracking, cancel restores inventory in transaction.

/admin/customers

GET /api/customers

Per-store customers, totalSpent desc, email unique per store.

/admin/marketing

/api/marketing/discounts, /api/marketing/stats

Discount codes (PERCENT/FIXED/BOGO, minSpend, expiry), abandoned carts.

/admin/channels & /admin/channels/storefront, /embed

GET+POST /api/channels/storefront

Hosted shop branding (name, themeColor, heroTitle/Subtitle). Embed gives {"
"} snippet.

/admin/settings, /admin/settings/payments, /admin/settings/api, /security

GET+POST /api/settings/payments (RazorpayKeyId + encrypted Secret, codEnabled) • GET /api/channels/storefront (storefrontToken, adminToken)

Payments: per-store Key ID/Secret (AES-GCM with ENCRYPTION_KEY) + COD toggle. **API:** Storefront (public) + Admin (secret, x-admin-token) + copy-paste fetch snippet.

NO MY-ACCOUNT LEAK

Before middleware, /admin was a client shell visible without JWT. Now edge checks existence + **jose verify** of the signature. Fake cookie → cleared. Every API also checks **storeId** or **x-admin-token** — 401 if not yours.

05 — Headless: your site, our commerce

Keep your journal, lookbook, story. Just add commerce.

FETCH — NEXT.JS COLLECTIONS PAGE

```
const { products } = await fetch("https://bluenexus.vercel.app/api/storefront/yourbrand/products?tag=featured&limit=24", { next:{revalidate:60} }).then(r=>r.json());  
// <Grid products={products} /> — your card, your lightbox, your CSS
```

BUY — HEADLESS CHECKOUT (RAZORPAY + COD)

```
const { key, razorpayOrderId, amount } = await fetch("/api/storefront/yourbrand/checkout",  
{method:"POST", body: JSON.stringify({ variantId: p.variants[0].id, quantity:1,  
method:"RAZORPAY" })).then(r=>r.json());  
new Razorpay({ key, order_id: razorpayOrderId, amount, handler: async (resp) => fetch("/api/storefront/yourbrand/verify", {method:"POST", body: JSON.stringify({...resp, variantId})}).then(()=> location.href="/shop/yourbrand/success") }).open();
```

Hosted fallback always works: **/shop/:subdomain** is the same products (hero + 3-col grid + ShopHeader cart `src/lib/shop-cart.ts` Zustand persist + `CartSheet` + `BuyButton` with Add to Cart + Buy Now + COD).

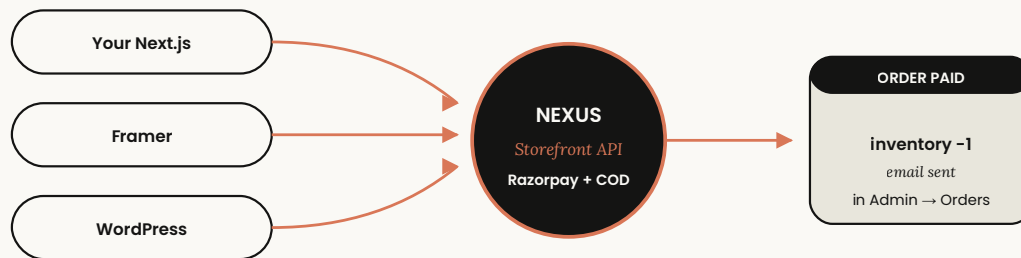


FIG. 02 – Any frontend → one Nexus engine. Same products, same checkout.

06 — Payments & orders

Razorpay-native: per-store **Key ID + Secret** (Admin → Settings → Payments), encrypted AES-GCM at rest via **ENCRYPTION_KEY**. COD is a boolean per store. Checkout creates Razorpay order (`amount paise`), verify checks `HMAC-SHA256(order_id|payment_id, secret)` idempotently (replay returns same orderId).

Inventory is **transactional**: `updateMany {inventoryCount: {gte: qty}, decrement: qty}` inside `$transaction` — two buyers hitting last piece, one gets Out of stock. Cancel (`POST /api/orders/cancel {orderId}`) restores stock.

CANCEL — RESTORES INVENTORY

`POST /api/orders/cancel { orderId } + Cookie JWT or x-admin-token → status FAILED + inventory ++ per variant`

07 — Security & isolation

Before: `/admin` was client-only — anyone saw the shell. Now `middleware.ts` (f Proxy) runs at edge: missing/invalid JWT → 302 login (and clears bad cookie). Auth APIs require **storeId** from JWT or **x-admin-token** — every **findMany** where: `{storeId}` and **delete** where: `{id, storeId}` prevents IDOR. Secrets: **JWT_SECRET** rotated from `nexus_secret_key_123` → 32B base64, **ENCRYPTION_KEY** 32B hex, **bcrypt 12**, **JWT 24h**, `httpOnly` secure lax. Input: **Zod** on every POST (title 1-200, tags 20, variants 50, email 254), **DOMPurify** on description, **DOMPurify + isomorphic-dompurify**. Rate-limit: `lib/rate-limit.ts` (5/hour register, 10/15m login, 3/hour demo seed, 20/m checkout) — swap to Upstash Redis at scale. Headers: **X-Frame DENY**, **nosniff**, **HSTS** in `next.config.ts`.

08 — Operate: env, DB, deploy

Env (Vercel → bluenexus → Settings): `DATABASE_URL` (pooler 6543 + pgbouncer), `DIRECT_URL` (5432), `JWT_SECRET` (now strong), `ENCRYPTION_KEY`, `NEXT_PUBLIC_URL`, `R2_*` (optional), `RESEND_*` (optional). **DB**: Prisma 7.6, `prisma generate && (prisma db push || true) && next build` in `package.json` handles stale DB (demo fallback to mock when ENOTFOUND). For real products, run **GET /api/demo/seed** once (needs valid DB).

Images: base64 to DB works, but set `R2_BUCKET`, `R2_ACCOUNT_ID`, `R2_ACCESS_KEY_ID`, `R2_SECRET_ACCESS_KEY`, `R2_PUBLIC_URL` to auto-upload via `lib/storage.ts` S3. **Emails**: with `RESEND_API_KEY`, `lib/email.ts` sends COD/PAID receipts; without, it logs and skips.

09 — API reference

Public (no auth) — `GET /api/storefront/:subdomain/products?productType=&tag=&vendor=&q=` • `GET /products/:slug` • `GET /config` • `POST /api/storefront/:subdomain/checkout {variantId, quantity, method, customerEmail}` • `POST /verify {razorpay_*, variantId}` • `GET /api/stores/check?subdomain=` • `GET /api/demo/seed`

Private (JWT cookie or x-admin-token) — GET/POST /api/products , DELETE /api/products/delete?id= , GET /api/products/list , GET /api/orders , POST /api/orders/fulfill , POST /api/orders/cancel , GET /api/customers , /api/settings/payments , /api/channels/storefront , /api/admin/stats .

Pages — / (marketing), /demo (9-step wizard), /integrate (headless guide), /shop:subdomain , /admin/* (middleware-guarded).

10 — Troubleshooting

500 storefront → stale Supabase pooler URL (ENOTFOUND tenant) — set fresh `DATABASE_URL` in Vercel, demo fallback keeps **demo** alive. **401 on admin API** → missing JWT or x-admin-token — login again (24h expiry). **Out of stock** → inventory `gte` check. **429** → rate limit (5/hour register, 20/min checkout). **Images 5MB** cap in `/admin/products/new` .

Do this today

- 01** Open `/demo` — watch 9 steps (Discover → Orders) like a video.
- 02** Hit **GET /api/demo/seed** once, then add a real product in `/admin/products/new` and checkout via COD on `/shop/demo`.
- 03** Copy **Storefront API** snippet from `/admin/settings/api` into your Next.js and make your marketing site shoppable.

Do those three and you've seen the whole loop — headless or hosted, same engine.

Build something people want to buy,

Manan — Blueblood Studio