Blissboard — Technical Architecture & Implementation Plan

This document lays out a clear, shareable technical architecture and implementation plan for Blissboard — a Next.js app where users choose event website templates, pay, submit event data, and receive a live link once the site is manually built by the Blissboard team within 72 hours.

---

1. Overview

Blissboard lets customers browse templates (wedding, anniversary, birthday...), purchase a template, submit event content, and receive a live hosted site. The build process is **manual**, meaning the Blissboard team matches the submitted user data with the chosen template and completes the site deployment within 72 hours.

---

2. High-level Components

- **Frontend (Next.js + React)**
    - Public pages: Homepage, Template Gallery, Template Detail, Checkout, Auth (Sign up / Login)
    - Protected pages: Dashboard, Orders, Event Details Form, Status Tracking
    - Libraries: React Hook Form, Zod/Yup, TailwindCSS (or Chakra), Framer Motion for animations
- **Backend (Next.js API routes or separate Node/Nest service)**
    - Handle business logic: orders, payments (webhooks), admin endpoints
    - Use Prisma (Postgres) or another ORM
- **Database**

    - PostgreSQL (primary), or managed DB (Neon, Supabase, PlanetScale for MySQL)
    - Use Prisma for schema & migrations
- **File Storage / CDN**
    - Cloudinary or AWS S3 for images/videos/assets
    - Store only URLs in DB
- **Payments**
    - Stripe (global) + Paystack (for Nigeria)
    - Webhooks to confirm payments & update order status
- **Hosting / Deployment**
    - **Frontend**: Vercel (recommended for Next.js)
    - **Live Event Sites**: Manually deployed to hosting/CDN after the 72-hour build
- **Authentication**
    - NextAuth.js or Clerk for account and session management
- **Email / Notifications**
    - SendGrid / Resend for transactional emails (payment confirmation, build completion, link delivery)
- **Admin Dashboard**
    - Template management (create/edit/delete), order management, manual site deployment tracking

---

3. Sequence (User Flow)

1. User visits site → browses gallery
2. User signs up / logs in to continue

3. User selects template → clicks Purchase
4. Checkout collects payment → payment provider returns success via webhook
5. On webhook: create `order` record (status: `paid`)
6. User completes event details form
7. Admin receives notification, manually builds site and deploys within 72 hours
8. Admin marks order as `live` and sends email + dashboard notification with link

---

4. Data Models (essential fields)

- **users**: id, email, name, passwordHash (or provider info), role, createdAt
- **templates**: id, title, slug, category, previewImages (array URLs), price, demoUrl, configJson
- **orders**: id, userId, templateId, amount, currency, status (pending|paid|processing|live|cancelled), paymentProviderId, createdAt, updatedAt
- **event_data**: id, orderId, formJson, mediaUrls (array), status
- **deploys**: id, orderId, deployUrl, provider, notes, status, createdAt

---

5. API Endpoints (examples)

- `GET /api/templates` — list templates
- `GET /api/templates/:slug` — template detail
- `POST /api/auth/signup` — sign up (or handled by NextAuth)
- `POST /api/checkout/create-session` — create Stripe/Paystack session
- `POST /api/webhooks/payment` — payment webhook

- `POST /api/orders/:id/event-data` — save event form data
- `GET /api/orders/:id/status` — order status
- `POST /api/admin/deploy/:orderId` — mark order as live and add deployment details

---

6. Build Strategy

Since builds are manual:

- Admin downloads event data, prepares the customized site locally or through internal tools
- Once ready, deploys to hosting (Vercel, Netlify, or static hosting bucket)
- Updates order status and sends the customer their live link

---

7. Security & Best Practices

- Use HTTPS everywhere
- Store secrets in environment variables
- Validate and sanitize uploads
- Use signed URLs for private media if needed
- Use payment provider hosted flows to avoid handling card data

---

8. MVP Feature Set

1. Template gallery + preview

2. Authentication (email + social)
3. Checkout with Stripe/Paystack
4. Payment webhook → create order
5. Event data form (post-payment)
6. Manual admin build workflow
7. Dashboard showing orders + live link
8. Admin panel for template & order management

---

9. Suggested Tech Stack (concise)

- Frontend: Next.js, React, TailwindCSS
- Auth: NextAuth.js (or Clerk)
- DB/ORM: PostgreSQL + Prisma
- Payments: Stripe (+ Paystack optional)
- Storage: Cloudinary or S3
- Hosting: Vercel (frontend), Vercel/Netlify/CDN for live sites
- Email: SendGrid / Resend
- Monitoring: Sentry

---

10. Next Steps

- Draw visual architecture diagram showing manual build step
- Prepare Trello/Jira task breakdown

- Create starter Next.js repo scaffold