Wedding Website Project Development Requirements (PDR)

[Your Name/Username]

September 16, 2025

Document Version: 1.2

1 Project Overview

1.1 Goal

Create a responsive website for a February 13, 2026, wedding at Tasi Ballroom, Hotel Nikko Guam (https://nikkoguam.com/en/meetingsevents/tasi-ballroom/). Features: RSVP, guest media uploads/comments, notifications/updates, AI chatbot for Q&A, Abstract Global Wallet (AGW) integration for $GRIND token gifting ($5 USD equivalent per guest). Share via WhatsApp link/QR code on invites. Launch MVP by end of September 2025 for invites.

1.2 Key Requirements

- **User Experience:** Simple but sexy—clean, intuitive for all ages/devices (iPhone/Android). Normie-friendly AGW onboarding for crypto newbies.
- **Accessibility:** High-contrast mode, ARIA labels. Add "Quick RSVP" mode (email-only, no blockchain/media) for simplicity.
- **Privacy/Security:** Basic compliance (e.g., no sharing data without consent). Store minimal data; use HTTPS. Not GDPR-level, as it's a personal wedding site.
- **Scale:** 200–250 guests. Media uploads: Industry standard limits (e.g., 10MB/image, 50MB/video, like Instagram/Twitter). Total storage: Free-tier caps (e.g., Supabase 500MB).
- **Budget:** Free tiers (Replit, Vercel hobby, GitHub free, Flowise self-hosted). You have basic subs for Vercel/GitHub/Replit/Twitter/ChatGPT—leverage those. API caps for AI (e.g., 100 queries/day).
- **Timeline:** MVP live by Sep 30, 2025 (invites out). Full features by Dec 2025. 2–4 weeks total dev.
- **Assumptions:** You're beginner-level (read code, use AI to explain/write). Use free tools; no paid APIs beyond basics. Wedding theme: Romantic, fun (cannabis ties—subtle nods?).

1.3 Design Preferences

- **Color Scheme:** Fuchsia primary (#FF00FF accents). Bridesmaids red (#FF0000), groomsmen black (#000000) suits with white (#FFFFFF) shirts. Ties: Cannabis trichomes macro (subtle green/crystal gradients, e.g., #00FF00 with white specks—use CSS gradients or image backgrounds).
- **Vibe:** Minimalist-romantic-modern. Inspiration: Clean like a wedding invite app (e.g., Joy or WithJoy sites). Landing page: Hero image of Tasi Ballroom (pull from site), explicit AGW login prompt: "Log in with AGW to RSVP & Get Your Gift! (No wallet? We'll set one up—easy!)"
- **Branding:** No logo yet—use couple's initials/monogram if provided later. Fonts: Elegant sans-serif (e.g., Montserrat via Google Fonts).

1.4 Venue Details (for AI FAQs/Site Info)

- **Tasi Ballroom:** 2nd floor, 396m^2, 4.8m height, seats 260. Transforms into 3 smaller rooms. Lavish dinners (260), cocktails (500). (From provided doc/link—embed map/schedule on /info page.)
- **FAQs Base:** Seating arrangements, menu (TBD—add placeholder), open bar rules (e.g., top-shelf till 11 PM), time/place (Feb 13, 2026, ceremony 4 PM, reception 6 PM), upload instructions.

2 Features

2.1 MVP (Core—Launch by Sep 30)

- **Landing/Home:** Welcome, venue photo, explicit AGW login button ("Log in to RSVP & Explore"). Quick RSVP button (email-only mode).
- **RSVP Form:** Name, email, guest count, attending (yes/no/maybe), dietary notes. Submit to DB. Unique ID for tracking.
- **Info Section:** Schedule, menu, bar rules, venue map (embed Google Maps), dress code (red/fuchsia theme, cannabis ties subtle).
- **Updates/Notifications:** Feed of announcements (you post via admin).
- **QR Code:** Generate site link QR for invites (downloadable).
- **AGW Integration:** Smooth login (AGW handles wallet gen/signup). Post-login: Collect wallet address, link to RSVP. Claim $GRIND? Make it opt-in (button: "Claim Your $5 GRIND Gift"—sends post-event to save tokens).
- **Deployment:** Live on custom domain (e.g., ourwedding2026.com via Vercel) or free Vercel subdomain.

2.2 Stretch (Add by Dec)

- **Guest Gallery:** Upload pics/videos/comments (pre/during/after). Limits: 10MB image, 50MB video. Threaded comments.
- **AI Chatbot:** Open-ended Q&A (e.g., "Can I bring my dog?" → "Check with

couple!"). Powered by Flowise (Grok/OpenAI—cap usage at 100/day). Embed as floating bubble.

- **Admin Dashboard:** For you—post updates, view RSVPs, batch-send $GRIND (script in Replit).
- **Quick Mode:** Email-only RSVP (no AGW/media). Button on landing: "Just RSVP Quick? Use Email."

3 Tech Stack

- **Frontend:** Next.js (React-based, Vercel-optimized). Tailwind CSS for styling (responsive, theme colors).
- **Backend/DB/Storage:** Supabase (free tier: Postgres DB for RSVPs, storage for media). Alternative: Firebase if preferred (easier real-time).
- **AGW/Crypto:** agw-react package (`https://www.npmjs.com/package/agw-react`). Abstract chain SDK for $GRIND sends (batch script in Replit—post-event).
- **AI:** Flowise (self-hosted in Replit or cloud free). LLM: Grok (via xAI API) or OpenAI (cap spending via API keys).
- **Other Libs:** qrcode.react (QR gen), react-chatbot-kit (AI embed), react-dropzone (uploads).
- **Tools Breakdown:**
  - **GitHub:** Repo management.
  - **Replit:** Coding/dev/testing (Next.js template).
  - **Vercel:** Deployment/hosting (auto-deploys from GitHub).
  - **Flowise:** AI flow builder (embed via API/iframe).

3.1 Dependencies & Env Vars

- **Packages (install in Replit):** `next react react-dom tailwindcss @supabase/supabase-j` `agw-react qrcode.react react-chatbot-kit react-dropzone`.
- **Env Vars (set in Vercel/Replit):** SUPABASE_URL, SUPABASE_KEY, AGW_API_KEY, FLOWISE_API_URL, GROK_API_KEY (for AI caps).

3.2 Domain Recommendations

You do not strictly need to buy a domain to make everything work—Vercel provides a free subdomain (e.g., wedding-site-2026.vercel.app) that's fully functional for HTTPS, sharing via WhatsApp, and QR codes. This keeps costs at $0 and avoids any setup delays. However, I recommend purchasing a custom domain for professionalism: It looks more polished on invites (e.g., ourwedding2026.com vs. a long Vercel URL), is easier to remember/share, and builds trust (no "vercel.app" branding). Average cost is $10–20/year based on current 2025 pricing from registrars like GoDaddy, Namecheap, or Vercel itself (which handles DNS/SSL for free). Buy via Vercel dashboard for seamless integration—no extra config needed. If budget is tight, start with free subdomain and add domain later (Vercel allows easy upgrades without redeploying).

4 Tasks Breakdown by Tool (Ordered for One-Shot Implementation)

Prioritize MVP. Each task includes beginner-friendly notes (e.g., "Copy-paste this code; AI explain if needed"). Expanded with more in-depth instructions for Flowise (agent setup) and Vercel (frontend/deployment) to avoid confusion/loops—use as direct prompts for ChatGPT/Replit AI (e.g., "Implement this exact Flowise flow").

4.1 GitHub (Start Here—Day 1)

1. Create repo: "wedding-site-2026" (private). Add .gitignore for Next.js.
2. Branches: main (prod), develop (work).
3. Add README: Paste this PDR.
4. Secrets: Add env vars (e.g., Supabase keys).
5. Integrate: Link to Replit (clone) and Vercel (import repo).

4.2 Replit (Core Dev—Days 1–7)

1. Setup: New Replit, select Next.js template. Clone GitHub repo. Run `npm install` + listed packages.
2. Basic Structure (Day 1–2):
   - pages/index.js: Landing with hero (venue image), AGW login button, Quick RSVP button.

     ```
     // Example: AGW Login
     import { useAGW } from 'agw-react';
     export default function Home() {
       const { connect, address, isConnected } = useAGW();
       return (
         <div className="bg-fuchsia-100 text-black"> {/* Tailwind classes
           <h1>Welcome to Our Wedding!</h1>
           {!isConnected ? <button onClick={connect}>Log in with AGW (Eas
           <button>Quick RSVP (Email Only)</button>
         </div>
       );
     }
     ```

   - Style with Tailwind: Add config for colors (fuschia: '#FF00FF', red: '#FF0000', etc.).
3. RSVP (Day 3): /rsvp page. Form submits to Supabase.

   ```
   // Use Supabase client
   import { createClient } from '@supabase/supabase-js';
   const supabase = createClient(process.env.SUPABASE_URL, process.env.SUPABASE_
   // On submit: supabase.from('rsvps').insert({ name, email, ... });
   ```

4. Info/Updates (Day 4): Static /info (venue details), dynamic /updates (pull from Supabase).
5. Gallery (Stretch, Day 5): Upload form with react-dropzone → Supabase storage.
6. AGW Full (Day 6): Post-login, store address in DB. Opt-in claim button (logs for now; batch-send later).
7. AI Prep (Day 7): Placeholder chat. Fetch from Flowise API.
8. QR (Day 7): /admin page with qrcode.react.
9. Test: Run `npm run dev`. Mobile preview in Replit.

4.3 Vercel (Deployment—Days 2–8; In-Depth to Avoid Credit Burn/Loops)

Vercel hobby tier is free (unlimited deploys, no "credits" burn for basic use—only pro features cost). To one-shot: Focus on auto-deploys from GitHub pushes; test previews before main. Detailed steps:

1. Dashboard Setup: Sign in, "New Project" → Import GitHub repo. Select Next.js preset (auto-detects). No custom build commands needed (defaults: `next build`).
2. Env Vars: Add via dashboard (e.g., SUPABASE_URL from Supabase project). Test: Redeploy button—watch logs for errors (e.g., "Missing env var?"). If button fails (e.g., AGW connect), check console: "Network tab for API calls; debug with console.log in code."
3. Domain: If buying ($∼10–20/year), add in "Domains" tab—Vercel handles purchase/DNS/SSL (auto-redirects). Else, use free .vercel.app. Prompt tip: "If deploy fails, push small changes (e.g., add console.log) to develop branch for preview URLs—fix locally before main."
4. Frontend Optimization: For buttons (e.g., AGW/RSVP): Use Vercel Speed Insights (free) post-deploy to check load times. If slow, add Next.js Image for venue pics. Error handling: Wrap buttons in try-catch; log to Vercel dashboard (view in "Logs" tab).
5. Deploy: Set "develop" branch for previews (test buttons live without burning anything). Merge to main for prod. Analytics: Enable free tier—monitor for issues.
6. Troubleshooting Prompt: If stuck (e.g., button not working), copy error from Vercel logs into ChatGPT: "Fix this Vercel deploy error: [paste log]. Code: [paste snippet]."

4.4 Flowise (AI—Days 7–10; In-Depth Agent Instructions)

Self-host in Replit (free) or use cloud trial. One-shot agent build: Drag-drop nodes for low-code. Detailed flow:

1. Setup: New Flowise project. Add LLM node (Grok/OpenAI—set API key with usage cap: e.g., OpenAI dashboard limit $5/month).
2. Build Agent:
   - Start: "Chat Input" node (user query).
   - Knowledge: "Vector Store" node—upload JSON file of FAQs (e.g., {"seat-

ing": "Arranged by table—check updates.", "menu": "TBD, dietary notes in RSVP.", "bar": "Open till 11 PM.", "venue": "Tasi Ballroom, 2nd floor, seats 260.", "upload": "Drag-drop on gallery, 10MB max."}). Use "Upsert" to add data.

- Logic: "Agent" node (for open-ended)—connect to LLM. Prompt: "You are a helpful wedding assistant. Use FAQs first. For open-ended (e.g., 'Can I bring my dog?'), respond politely: 'Best to check with the couple!' Limit to 100 chars if possible. If unknown, say 'Ask the hosts!"'
- Tools: Add "API Tool" for dynamic (e.g., query Supabase for RSVP count: "GET /api/rsvps").
- Output: "Chat Output" node. Caps: Set LLM rate limit in node (e.g., 100/day via API wrapper).
- Test: Run flow in Flowise UI—query "Open bar?" → Expected: "Yes, top-shelf till 11 PM."

3. Embed: Export as API (get endpoint URL). In Replit/Vercel: Use fetch in chat component (react-chatbot-kit). Snippet:

```
// Chat fetch
async function handleQuery(query) {
  const res = await fetch(process.env.FLOWISE_API_URL, { method: 'POST', bod
  return res.json().text;
}
```

4. Prompt Tip: Paste this section into Flowise chat: "Build this exact agent flow: [copy steps]. Debug: If no response, check LLM key in logs."

5 Implementation Checklist

Use this checklist to track progress step-by-step. Mark as [ ] Incomplete or [x] Complete. Share updates with AI (e.g., "Confirm step 1 done") for verification and to proceed orderly. Grouped by tool/timeline for clarity.

5.1 GitHub Setup (Day 1)

Create repo "wedding-site-2026" (private) with .gitignore for Next.js.

Set up branches: main (prod), develop (work).

Add README with this PDR pasted in.

Add secrets for env vars (e.g., Supabase keys).

Integrate with Replit (clone repo) and Vercel (import repo).

5.2 Replit Development (Days 1–7)

Create new Replit project with Next.js template; clone GitHub repo; run `npm install` + all packages.

Build basic structure: Add landing page (index.js) with hero image, AGW login button, and Quick RSVP button (use provided code snippet).

Configure Tailwind CSS with custom colors (fuschia, red, etc.).

Implement RSVP form on /rsvp page; connect to Supabase for data submission (use provided code).

Add static /info page with venue details and dynamic /updates page pulling from Supabase.

(Stretch) Add guest gallery with upload form using react-dropzone and Supabase storage.

Integrate full AGW: Store wallet address post-login; add opt-in claim button for $GRIND.

Prepare AI chatbot placeholder; add fetch logic for Flowise API (use provided snippet).

Add QR code generation on /admin page using qrcode.react.

Test locally: Run `npm run dev` and check mobile preview in Replit.

5.3 Vercel Deployment (Days 2–8)

Set up dashboard: Import GitHub repo, select Next.js preset.

Add env vars in dashboard (match Replit/GitHub).

(Optional) Add/purchase domain in "Domains" tab if desired.

Optimize frontend: Use Speed Insights; add error handling to buttons (try-catch).

Deploy previews: Push to develop branch; test buttons/logs.

Deploy production: Merge to main; enable analytics.

5.4 Flowise AI Integration (Days 7–10)

Create new Flowise project; add LLM node with API key and usage caps.

Build agent flow: Add Chat Input, Vector Store with FAQs JSON, Agent node with prompt, API Tool for dynamics, Chat Output.

Test flow in Flowise UI with sample queries.

Export as API; embed in Replit/Vercel site using fetch snippet.

5.5 Overall Testing & Launch (Days 8–14)

Cross-device testing: iPhone/Android simulation for responsiveness.

Privacy check: Add basic notice to site footer.

MVP launch: Confirm live URL/QR by Sep 30; share for invites.

(Post-Launch) Add stretch features: Gallery, full AI, admin dashboard, $GRIND batch script.

5.6 Admin & Iteration (Ongoing)

Set up admin dashboard for updates/RSVP views.

Monitor API usage and storage limits.

Iterate based on feedback (e.g., add monogram if provided).

6 Timeline & Milestones

- **Week 1 (Sep 16–22):** GitHub/Replit setup, MVP pages (landing, RSVP, info). Deploy preview to Vercel.
- **Week 2 (Sep 23–30):** AGW, updates, QR. AI basics. Full test (iPhone/Android). Launch MVP.
- **Post-Launch (Oct+):** Gallery, full AI, $GRIND script. Iterate based on feedback.
- **Daily Routine:** Commit to GitHub, push for Vercel previews. Use ChatGPT to explain code.

7 Risks & Mitigations

- **Beginner Coding:** Use templates (search GitHub for "Next.js wedding site"). AI-debug errors.
- **AGW Onboarding:** Test smoothness—fallback to email if issues.
- **API Costs:** Monitor Flowise/Grok usage; switch to free LLM if over.
- **Storage Limits:** Warn users on uploads; upgrade if needed.
- **Compliance:** Add basic privacy notice (e.g., "Data for wedding only—not shared.") inspired by hotel policy.
- **Testing:** Cross-device (use browser dev tools). Guest simulation.

8 Deliverables

- Live Site: URL + QR by Sep 30.
- Repo: Fully documented code.
- Admin Guide: How to post updates/send tokens.
- Follow-Up: Share repo link for reviews/snippets.