# Project Context: Two-Day Wedding Information & Interactive Guest Site

This is a single-page application (SPA) built to serve 100-150 wedding guests. It provides essential, real-time information about the two-day event and includes interactive features for personalization and entertainment.

## 1. High-Level Architecture

The architecture is designed to be serverless, low-cost (using free tiers), and highly scalable:

* **Source of Truth:** Google Sheets (4 tabs: guests, events, outfits, playlist\_requests).
* **Data API:** SheetDB, acting as a read/write REST API for the Google Sheets data.
* **Frontend (App):** Single-file React application (src/App.jsx) using Vite and Tailwind CSS.
* **Secure Backend:** Vercel Serverless Function (api/add-song.js) to securely handle sensitive API calls.
* **External Service:** YouTube Data API (v3) for playlist management.
* **Deployment:** Vercel (free tier).

## 2. Current Development Status (Alpha Ready)

All core features are implemented in the current codebase. The focus is now on final testing and adding minor utility features.

| **Feature Area** | **Component/File** | **Status** | **Notes** |
| --- | --- | --- | --- |
| **Data & Configuration** | .env, src/App.jsx | **Configuration Required** | All API keys (SheetDB, YouTube Client ID/Secret/Refresh Token) and IDs (Playlist ID) are placeholders. **These MUST be set correctly in the Vercel Environment Variables before deployment and in the local .env file for testing.** |
| **Authentication** | LoginPage, fetchGuestByEmail | **Complete** | Simple email lookup (non-OAuth) against the SheetDB guests sheet. Guest info (e.g., room\_number) is stored in sessionStorage upon successful login. |
| **Live Timeline** | TimelineSection | **Complete** | Fetches events data (which includes adjusted\_start\_time calculated by a formula in Google Sheets). Displays Current/Next status based on a 2-hour duration assumption. Polls every 10 minutes, with a manual refresh button. |
| **Outfit Guide** | OutfitSection | **Complete** | Fetches and joins events and outfits data by event\_id for chronological display. |
| **Song Requests** | PlaylistSection, api/add-song.js | **Complete (Pending API Keys)** | Frontend handles URL validation and calls the Vercel Serverless Function (/api/add-song). The Vercel function securely calls YouTube Data API to insert the video and logs the transaction to the SheetDB playlist\_requests audit sheet. |
| **Styling** | All components | **Mobile-First Aesthetic** | Uses Tailwind CSS with a pink/serif theme, optimized for mobile display. |

## 3. Next Steps & Development Goals

The immediate focus should be on rigorous testing and adding final touches.

### Goal A: End-to-End Testing (Priority 1)

1. **Local Test:** Run npm run start-api and npm run dev simultaneously.
   * Verify the Login works with a valid test email.
   * Verify manual changes to the delay\_minutes cell in Google Sheets are reflected instantly when the user clicks the refresh button on the Timeline.
   * Verify that adding a YouTube URL successfully inserts a video into the live YouTube playlist AND logs a row in the SheetDB playlist\_requests tab.
2. **Deployment Test:** Complete the Vercel deployment (Phase 3) and verify all features function on the live Vercel URL.

### Goal B: Feature Expansion (Next Iteration)

1. **Detailed Contact/FAQ Section:** Build out the placeholder 'FAQs & Contact' tab to display static information (emergency numbers, venue address, local taxi info). This data can be added to a new sheet (info) and fetched via SheetDB.
2. **Wedding Gallery Placeholder:** Add a simple, responsive gallery section using placeholder images that can be updated closer to the date.
3. **Refine Time Zone Handling:** Currently assumes the client time zone is the event time zone. Consider displaying a warning if the guest is in a different time zone.

**You now have the full context of the Wedding Information Website project. I am ready to start developing and testing features on top of this code.**