Make My Marriage

System Design & Architecture — Version 1.0

Architecture: Modular Monolith using Next.js + FastAPI + MongoDB Atlas + Google Cloud Storage (GCS). The wedding is the central entity; members collaborate inside one wedding workspace, while guests use public links/QRs without accounts.

High-level flow: Users → Next.js → FastAPI → MongoDB Atlas. Media uses Browser → signed URL → GCS. External integrations include Google OAuth, Google Places/Maps, Resend and YouTube. Google Cloud CDN is optional for later scale.

1. Technology Stack

Layer	Technology	Purpose
Frontend	Next.js + TypeScript + Tailwind CSS	Dashboard, landing/demo, public wedding website and guest pages.
Backend	FastAPI + Python	REST APIs, business logic, authorization, integrations and signed URLs.
Database	MongoDB Atlas	Users, weddings, members, events, guests, RSVP, tasks, shopping, expenses, vendors and metadata.
Media	Google Cloud Storage	Photos, documents, invitation assets and other media.
Auth	Google OAuth / OpenID Connect	Member signup/login; first Google login creates the user account.
Email	Resend	Member invitations and transactional emails.
Vendor discovery	Google Maps / Places APIs	Search and location information for wedding vendors.
Live stream	YouTube Live URL	Store/display event live-stream links.
CDN	Google Cloud CDN — later	Optional faster delivery for high-volume public gallery/media.

2. FastAPI Modular Structure

Auth & Users • Wedding Management • Events • Guests & RSVP • Tasks • Shopping • Expenses & Budget • Vendors • Invitations • Gallery & Photos • Public Website • Documents • Integrations

3. MongoDB Collections

Collection	Purpose
users	Google identity, email, name and profile
weddings	Couple details, slug, settings, budget and theme
wedding_members	wedding_id + user_id + role: admin/manager/member
invitations	Email-bound invite token, role, expiry and status
events	Event details, venue, date/time, description and live URL
guests	One central guest collection per wedding
event_attendance	Guest/event mapping, RSVP and attendee counts
tasks	Assignee, event, due date, priority and status
shopping_items	Item, quantity, estimated/actual price, buyer and status
expenses	Amount, category, event, paid_by, vendor and receipt reference
vendors	Selected vendor, quote, advance, paid, remaining and documents
photos	Wedding/event, uploader, GCS path, visibility and metadata
documents	Wedding/event/vendor references and GCS path

4. Core Request Flows

Google Signup/Login: Next.js → Google OAuth → FastAPI → MongoDB → authenticated wedding workspace.

Member Invitation: Admin selects email + role → FastAPI creates invitation → Resend sends link → member opens link → Google login with invited email → invitation validated → wedding membership created.

Photo Upload: Browser requests upload permission → FastAPI validates access → signed GCS URL → browser uploads directly to GCS → FastAPI stores metadata in MongoDB.

Guest QR Photo Upload: Guest opens public QR/link → FastAPI validates public token → signed GCS URL → direct upload → metadata saved.

RSVP: Guest opens public RSVP → submits event attendance → FastAPI validates → MongoDB stores RSVP.

Vendor Discovery: Next.js → FastAPI → Google Places/Maps → result → Admin adds vendor to wedding.

Public Wedding Website: Visitor opens /w/{slug} → Next.js gets public data through FastAPI → renders wedding details, RSVP, gallery and live stream.

5. Roles & Access

Admin: wedding owner; manages members, roles and shared data. Manager: invited collaborator. Member: wedding collaborator. Guest: no account; RSVP/gallery/photo links only. Visitor: public landing/demo/website access.

6. Media & Security

GCS should remain private. FastAPI controls authorization and generates short-lived signed URLs for upload/download. MongoDB stores media metadata and GCS object paths, not image binaries. Invitation links should use high-entropy tokens, expiry and email binding for role-based member invitations.

7. Deployment Architecture

Current baseline: Google Cloud for application hosting and GCS, MongoDB Atlas as the managed database. Next.js and FastAPI remain the main application deployment; the exact GCP compute product (for example Cloud Run) will be finalized during implementation. Secrets such as OAuth credentials, Resend API keys and database credentials must use a managed secret store.

Production paths: Browser → Next.js → FastAPI → MongoDB Atlas. Media: Browser → signed GCS URL → GCS. Optional later: GCS/public media → Google Cloud CDN → Users.

8. Repository Structure

make-my-marriage/
├── frontend/ — Next.js (app, components, features)
├── backend/ — FastAPI modular monolith (auth, wedding, events, guests, tasks, shopping, expenses, vendors, invitations, gallery, website, integrations, database, core)
├── docker-compose.yml
└── README.md

9. V1 Scope & Implementation Note

V1 includes Google authentication, member invitations, wedding/events, guests/RSVP, tasks, shopping, expenses, vendor discovery/management, digital invitations, public wedding website, gallery/photo uploads, GCS storage, YouTube Live links and mocked WhatsApp/SMS. Microservices, custom video streaming, advanced media processing and CDN are not required initially. This is the current baseline; libraries, cloud products, authentication/session details, deployment choices and providers may change during development based on cost, security, performance and practical requirements.

System Design PDF with Architecture Diagram