Make My Marriage

Database Design Document — Version 1.0

MongoDB Atlas is the application database. The wedding is the central entity. Members authenticate with Google; guests use public links/QRs without accounts. Media files are stored in GCS and only their metadata/path is stored in MongoDB.

1. Database Principles

• Wedding-centric: private business records contain wedding_id.
• Central guest list: one guest record per wedding; event attendance is separate.
• Members vs guests: members are authenticated users; guests do not need accounts.
• References over duplication for growing/shared data.
• Images/documents stay in GCS, not MongoDB.
• FastAPI performs authorization and data access checks.

2. High-Level Relationships

User → Wedding Membership → Wedding
Wedding → Events
Wedding → Guests → Event Attendance → Events
Wedding → Tasks / Shopping / Expenses / Vendors / Invitations / Photos / Documents
Event → Tasks / Expenses / Vendors / Photos / Attendance

3. Collections

Collection	Responsibility
users	Google identity, email, name, profile
weddings	Couple details, slug, settings, budget, theme
wedding_members	User-to-wedding membership and role
member_invitations	Pending/accepted member invitations
events	Default and custom wedding events
guests	Central guest list
event_attendance	Guest-to-event RSVP/attendance
tasks	Wedding and event tasks
shopping_items	Shopping and purchase tracking
expenses	Budget and expense records
vendors	Selected/managed wedding vendors
vendor_event_links	Vendor-to-event mapping
invitation_designs	Digital invitation configuration
photos	GCS photo references and metadata
documents	GCS document references and metadata
public_links	Secure public RSVP/gallery/upload links

4. Core Collection Schemas

users

- _id: ObjectId
- google_id: string, unique
- email: string, normalized, unique
- name: string
- avatar_url: string | null
- created_at / updated_at: datetime

weddings

- _id: ObjectId
- slug: string, unique
- couple: { partner1_name, partner2_name }
- created_by: ObjectId → users
- wedding_date: datetime | null
- status: active | archived
- theme: object
- settings: object
- created_at / updated_at: datetime

wedding_members

- _id: ObjectId
- wedding_id: ObjectId
- user_id: ObjectId
- role: admin | manager | member
- status: active | removed
- joined_at / created_at / updated_at: datetime

member_invitations

- _id: ObjectId
- wedding_id: ObjectId
- email: normalized string
- role: manager | member
- token_hash: string
- invited_by: ObjectId
- status: pending | accepted | expired | revoked
- expires_at: datetime
- accepted_by / accepted_at: nullable

events

- _id: ObjectId
- wedding_id: ObjectId
- name: string
- event_type: string
- date: datetime
- end_date: datetime | null
- venue: { name, address, lat, lng }
- description / dress_code: nullable
- live_stream_url: nullable
- status: active | cancelled
- created_by: ObjectId

guests

- _id: ObjectId
- wedding_id: ObjectId
- name: string
- phone / email: nullable
- relationship: nullable
- side: bride | groom | both | other
- family_group: nullable
- adults: int
- children: int
- notes: nullable

event_attendance

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId
- guest_id: ObjectId
- rsvp_status: pending | accepted | declined
- attendee_count: int
- adults: int
- children: int
- message: nullable
- responded_at: nullable

tasks

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId | null
- title: string
- description: nullable
- assigned_to: ObjectId | null
- due_date: nullable
- priority: low | medium | high
- status: todo | in_progress | completed
- created_by: ObjectId

shopping_items

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId | null
- name: string
- category: string
- quantity: number
- estimated_price: number
- actual_price: number | null
- buyer_user_id: ObjectId | null
- status: to_buy | partially_purchased | purchased

expenses

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId | null
- vendor_id: ObjectId | null
- name: string
- category: string
- amount: number
- currency: string
- paid_by: ObjectId | null
- payment_status: unpaid | partial | paid
- expense_date: datetime
- receipt_document_id: ObjectId | null

vendors

- _id: ObjectId
- wedding_id: ObjectId
- name: string
- category: string
- phone / email / website: nullable
- address: object | null
- google_place_id: nullable
- rating: nullable
- quote_amount: nullable
- advance_amount: nullable
- paid_amount: nullable
- remaining_amount: nullable
- status: shortlisted | booked | completed | cancelled

vendor_event_links

- _id: ObjectId
- wedding_id: ObjectId
- vendor_id: ObjectId
- event_id: ObjectId
- notes: nullable

invitation_designs

- _id: ObjectId
- wedding_id: ObjectId
- theme_id: string
- title: string
- content: object
- asset_paths: [string]
- public_slug: string
- status: draft | published

photos

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId | null
- uploaded_by_user_id: ObjectId | null
- guest_upload: bool
- gcs_object_path: string
- mime_type: string
- file_size: number
- visibility: private | shared | public
- caption: nullable

documents

- _id: ObjectId
- wedding_id: ObjectId
- event_id: ObjectId | null
- vendor_id: ObjectId | null
- name: string
- document_type: contract | bill | receipt | booking | other
- gcs_object_path: string
- mime_type: string
- file_size: number
- uploaded_by: ObjectId

public_links

- _id: ObjectId
- wedding_id: ObjectId
- type: rsvp | gallery | photo_upload | invitation
- token_hash: string
- scope: object
- expires_at: nullable
- status: active | revoked | expired

5. Index Strategy

Recommended V1 indexes:
• users: unique google_id, unique normalized email.
• weddings: unique slug.
• wedding_members: unique (wedding_id, user_id), plus (user_id, status).
• member_invitations: (wedding_id, email, status) and TTL on expires_at where appropriate.
• events: (wedding_id, date).
• guests: (wedding_id, name), optionally (wedding_id, phone/email).
• event_attendance: unique (event_id, guest_id), plus (wedding_id, rsvp_status).
• tasks: (wedding_id, status), (wedding_id, assigned_to, status), (wedding_id, due_date).
• expenses: (wedding_id, event_id), (wedding_id, expense_date), (wedding_id, category).
• vendors: (wedding_id, category), optionally google_place_id.
• photos: (wedding_id, event_id, created_at), (wedding_id, visibility).
• public_links: unique token_hash, plus (wedding_id, type, status).

6. Tenant Isolation & Authorization

For every private request, FastAPI should: (1) authenticate the user, (2) resolve their wedding membership, (3) verify role/permission, and (4) query using wedding_id. The frontend must never be the only security layer. This prevents data from one wedding being accessed by another wedding.

7. Embed vs Reference

Embed: small tightly-owned data such as couple details, wedding settings, event venue and theme configuration.
Reference: users, members, guests, events, tasks, expenses, vendors, photos and documents because these grow independently or participate in multiple relationships.
Never embed: photo/document binaries or large growing lists inside the wedding document.

8. Example

Ashu & Priya wedding → Haldi / Mehndi / Sangeet / Wedding / Reception → central guest list → event_attendance maps Rahul to selected events → vendors can link to multiple events → expenses can belong to wedding or event → photos reference GCS → members are stored through wedding_members.

9. Data Lifecycle

Create wedding → create Admin membership → create default events → invite members → add guests/vendors/tasks/shopping → collect RSVP → track expenses → upload photos/documents → publish selected public content → archive wedding. Important records should generally use status/soft-delete rather than immediate hard deletion where recovery/auditability matters.

10. V1 Database Rules

1. MongoDB is the source of truth for application metadata/business records.
2. GCS is the source of truth for media files.
3. Never store image/document binaries in MongoDB.
4. Always scope private records by wedding_id.
5. Guests do not need user accounts.
6. Keep event attendance separate from the central guest record.
7. Store money as numeric values with explicit currency.
8. Add/adjust indexes according to real query patterns during development.

11. Future Changes

This is the initial database baseline. Collections and fields may change during API implementation, UI development, security review, performance testing and real-world usage. New modules should preserve the wedding-centric tenant model and FastAPI authorization boundary.