MongoDB Data Modeling – LifeLink Project

# 1. Identify Collections

|  |  |
| --- | --- |
| **Collection** | **Purpose** |
| users | All users (donors, recipients, admins) |
| donations | Records of each donation (blood or organ) |
| requests | Requests for blood/organs made by recipients |
| appointments | Scheduled donation appointments |
| healthRecords | Donor's health and eligibility data |
| alerts | Emergency alerts sent to nearby donors |
| hospitals | Partnered hospitals/blood banks |
| forumPosts | Community discussions and success stories |

# 2. Define Key Relationships

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Relationship | Type | Description | Modeling Approach |
| 1 | User ↔ Requests | One-to-Many | A recipient can make many requests | requests.userId → users.\_id |
| 2 | Donor ↔ Donations | One-to-Many | A donor can donate multiple times | donations.userId → users.\_id |
| 3 | User ↔ Appointments | One-to-Many | Users schedule appointments | appointments.userId → users.\_id |
| 4 | Donor ↔ Health Record | One-to-One | Each donor has one health record | healthRecords.userId → users.\_id |
| 5 | Requests ↔ Alerts | One-to-Many | Each request can trigger alerts | alerts.requestId → requests.\_id |
| 6 | ForumPost ↔ Users | One-to-Many | User can write multiple posts | forumPosts.userId → users.\_id |
| 7 | Donations ↔ Hospitals | Many-to-One | Donations may be linked to hospitals | donations.hospitalId → hospitals.\_id |

**3. Embed vs Reference**

|  |  |
| --- | --- |
| **Use Case** | **Recommended Modeling** |
| Order items with product snapshot | Embed |
| Product belongs to a seller | Reference sellerId |
| Product belongs to a category | Reference categoryId |
| Review made by a user | Reference userId |
| Cart items | Embed with product snapshot |

# 4. Sample Schemas

* Schema: users

{

"\_id": ObjectId("..."),

"name": "Janane",

"email": "janane@gmail.com",

"role": "donor", // or "recipient", "admin"

"location": {

"city": "Chennai",

"zip": "600001"

},

"bloodGroup": "O+",

"organsWillingToDonate": ["kidney", "liver"],

"createdAt": ISODate(...),

"updatedAt": ISODate(...)

}

* Schema: requests

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"type": "blood", // or "organ"

"bloodGroup": "B+",

"organ": "kidney",

"urgency": "high",

"status": "pending",

"createdAt": ISODate(...),

"updatedAt": ISODate(...)

}

* Schema: donations

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"hospitalId": ObjectId("..."),

"type": "blood",

"date": ISODate("2025-08-01T10:00:00Z"),

"status": "completed",

"createdAt": ISODate(...),

"updatedAt": ISODate(...)

}

* Schema: appointments

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"date": ISODate("2025-08-05T10:00:00Z"),

"location": "Apollo Hospital, Chennai",

"status": "scheduled",

"createdAt": ISODate(...),

"updatedAt": ISODate(...)

}

* Schema: healthRecords

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"weight": 60,

"bp": "120/80",

"eligible": true,

"lastDonationDate": ISODate("2025-06-01"),

"nextEligibleDate": ISODate("2025-08-01"),

"createdAt": ISODate(...),

"updatedAt": ISODate(...)

}

* Schema: alerts

{

"\_id": ObjectId("..."),

"requestId": ObjectId("..."),

"message": "Urgent blood needed: B+",

"nearbyUserIds": [ObjectId("..."), ObjectId("...")],

"status": "sent",

"sentAt": ISODate("2025-07-23T10:30:00Z")

}

**5. Query Scenarios to Think About (LifeLink)**

|  |  |
| --- | --- |
| **Scenario** | **Fields to Use / MongoDB Query** |
| Find eligible donors in Chennai with B+ blood group | users.find({ "location.city": "Chennai", "bloodGroup": "B+", "role": "donor", "eligibility": true }) |
| Get all pending blood or organ requests by urgency | requests.find({ "status": "pending", "urgency": "high" }) |
| Show donation history of a specific donor | donations.find({ "userId": ObjectId("...") }) |
| List upcoming appointments for a user | appointments.find({ "userId": ObjectId("..."), "date": { "$gte": new Date() } }) |
| Get all alerts sent for a specific request | alerts.find({ "requestId": ObjectId("...") }) |