**MongoDB Data Modelling: Skill Exchange Platform**

**What Is Data Modelling?**

The Skill Exchange Platform is a community-driven web application designed to connect individuals who want to offer and learn skills without monetary transactions. Users can list their skills, find others with matching interests, exchange services, and provide feedback. The platform’s backend is powered by MongoDB due to its flexibility in managing diverse, semi-structured user data and its powerful querying and aggregation capabilities.

**1. Identify Collections**

|  |  |
| --- | --- |
| Collection | Purpose |
| users | Individuals who register on the platform to offer and learn skills |
| skills | |  | | --- | |  |  |  | | --- | | Catalog of all unique skills users can offer or learn | |
| skillRequests | Records of skill exchange requests sent from one user to another |
| matches | Confirmed mutual agreements between two users to exchange skills |
| feedback | User feedback and ratings after skill exchange sessions |
| messages | Direct messages exchanged between matched users |
| notifications | Alerts related to request status, matches, and new feedback |

**2. Define Relationships**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Relationship | Type | Description | Field Reference | Modeling Approach | Justification |
| 1 | User ⟷ Skill Requests | One-to-Many | A user can send many skill requests to other users. | skillRequests.fromUser → users.\_id | 🔗 Reference | Skill requests are transactional and user-specific; better stored separately. |
| 2 | User ⟷ Skills (Offered/Needed) | One-to-Many | Each user can offer or request many skills. | users.skillsOffered / skillsNeeded | 🧩 Embed (inside user doc) | Accessed frequently with user profile; embedding ensures quick access. |
| 3 | User ⟷ Matches | Many-to-Many | Two users can match based on shared skills. | matches.userA / userB → users.\_id | 🔗 Reference | Efficient and scalable for managing mutual matches between users. |
| 4 | User ⟷ Feedback | One-to-Many | A user can write feedback for others after exchange. | feedback.fromUser → users.\_id | 🔗 Reference | Feedback grows over time; should be independently queryable. |
| 5 | Skill Request ⟷ Skill | Many-to-One | Each skill request is associated with a single skill. | skillRequests.skill (string or skillId) | 🔗 Reference or Flat String | Use reference if skill catalog is centralized; else flat string for speed. |
| 6 | Match ⟷ Skills Exchanged | One-to-Many | A match record holds all exchanged skills. | matches.skillsExchanged (array) | 🧩 Embed (inside match doc) | Embedding reflects a complete transaction snapshot. |
| 7 | Feedback ⟷ Skill | One-to-One | Feedback is specific to one skill exchanged. | feedback.skill (string or reference) | 🧩 Embed/Flat string | Skill is part of session context; embedding or flat field is sufficient. |
| 8 | Match ⟷ Messages | One-to-Many | Users in a match can send messages to each other. | messages.matchId → matches.\_id | 🔗 Reference | Keeps messages separate for scalability; supports conversation threading. |

**3. Embed vs Reference**

|  |  |
| --- | --- |
| Use Case | Recommended Modeling |
| User profile with offered and needed skills | Embed skillsOffered[] and skillsNeeded[] |
| Skill request between users | Reference fromUser and toUser (userId) |
| Feedback made by a user for another user | Reference fromUser → userId |
| Messages in a match conversation | Reference matchId |
| Skills exchanged in a match session | Embed skillsExchanged[] |

**4. Sample Schemas**

**a.users**

{  
\_id,  
username,  
email,  
password,  
bio,  
skillsOffered: [String],  
skillsNeeded: [String],  
location,  
createdAt  
}

**b. skills**

{

\_id,

name,

category

}

**c. skillRequest**

{

\_id,

fromUser: ObjectId → users.\_id,

toUser: ObjectId → users.\_id,

skill,

status,

message,

requestedAt

}

**d. matches**

{

\_id,

userA: ObjectId → users.\_id,

userB: ObjectId → users.\_id,

skillsExchanged: [String],

matchedAt

}

**e.feedback**

{

\_id,

fromUser: ObjectId → users.\_id,

toUser: ObjectId → users.\_id,

skill,

rating,

comment,

createdAt

}

**5. Query Scenarios to Think About**

|  |  |
| --- | --- |
| Scenario | Fields to Use |
| Show all skill requests sent by a user | Filter by fromUser in skillRequests |
| Show all users who offer a specific skill | Filter users by skillsOffered |
| Show all feedback for a given user | Filter feedback by toUser |
| Get all matches involving a specific user | Filter matches where userA or userB equals userId |

### **6. Practice Task**

Your task for building the Skill Exchange Platform data model includes the following:

✅ Design the collections with relationships  
→ Collections: users, skills, skillRequests, matches, feedback, messages, notifications  
→ Relationships defined using referencing and embedding as needed

✅ Include createdAt and updatedAt in every schema  
→ All collections should include timestamps for tracking record creation and modifications.

✅ Seed sample documents in MongoDB  
→ At least 10 documents should be added to each collection with realistic values.