MongoDB Data Modelling: Real-Time Quiz & Hackathon Platform

Data Modelling:

Data modelling is the process of organizing and structuring data so it can be stored, retrieved, and managed efficiently in a database. It defines what data is stored, how different entities relate to each other, and the rules for handling them.

Project context: Real-Time Quiz & Hackathon Platform

We designed 9 collections with clear relationships to support quizzes, hackathons, teams, submissions, and rewards:

SNO	COLLECTIONS	PURPOSES
01	**Users**	Stores user info, roles (Admin/Participant/Judge), profiles, badges
02	**Quizzes**	Each quiz has title, description, questions (with options, answers, difficulty), schedule, and creator
03	**Quiz Attempts**	Links a user to a quiz, storing responses, scores, and completion time
04	**Hackathons**	Contains event details, problem statements, organizers, schedule, and team rules
05	**Teams**	Represents groups of users participating in a hackathon
06	**Submissions**	Stores team submissions (repo links, presentations, feedback, scores)
07	**Leaderboard**	Tracks rankings and scores for quizzes and hackathons
08	**Notifications**	Sends reminders, updates, and leaderboard announcements to users
09	**Rewards**	Assigns badges, certificates, or prizes to users for achievements

Define Relationships:

SNO	Relationship	Type	Description	Field Reference	Modelling Approach	Justification
01	Users → Quizzes	One-to-Many	A user (admin) can create many quizzes.	Quizzes.createdBy → Users._id	Reference	Keeps quizzes lightweight; avoids duplicating user details
02	Users → QuizAttempts	One-to-Many	A user can attempt multiple quizzes.	QuizAttempts.userId → Users._id	Reference	Efficient tracking of user attempts without bloating the quiz data
03	Quizzes → QuizAttempts	One-to-Many	Each quiz can have multiple attempts	QuizAttempts.quizId → Quizzes._id	Reference	Prevents duplication of quiz content for each attempt
04	Users → Hackathons	One-to-Many	A user (organizer) can create hackathons.	Hackathons.organizerId → Users._id	Reference	Keeps hackathons linked to their creators without embedding
05	Hackathons → Teams	One-to-Many	Each hackathon can have multiple teams.	Teams.hackathonId → Hackathons._id	Reference	Makes team management scalable for large hackathons
06	Teams → Submissions	One-to-One	Each team can submit one main project	Submissions.teamId → Teams._id	Reference	Avoids embedding heavy submission data in team docs
07	Hackathons → Submissions	One-to-Many	A hackathon can have many submissions	Submissions.hackathonId → Hackathons._id	Reference	Easy retrieval of all submissions for an event
08	Users → Leaderboard	One-to-Many	A user can appear in multiple leaderboards	Leaderboard.userId → Users._id	Reference	Keeps leaderboard entries lean with just user references
09	Events → Leaderboard	One-to-Many	Each quiz or hackathon has a leaderboard	Leaderboard.eventId → Quizzes/Hackathons._id	Reference	Allows a single leaderboard structure for different event types

10	Users → Notifications	One-to-Many	A user can have multiple notifications	Notifications.userId → Users._id	Reference	Keeps notifications separate for quick updates and queries
11	Users → Rewards	One-to-Many	A user can earn multiple rewards	Rewards.userId → Users._id	Reference	Avoids embedding many rewards inside user docs
12	Events → Rewards	One-to-Many	Rewards are tied to quizzes or hackathons	Rewards.eventId → Quizzes/Hackathons._id	Reference	Enables flexible linking of rewards to any event type

Sample schema:

1.users

```
{
  _id: ObjectId,
  name: String,
  email: String,
  role: { type: String, enum: ["Admin", "Participant", "Judge", "Moderator"] },
  profile: { bio: String, skills: [String] },
  badges: [String],
  createdAt: Date,
  updatedAt: Date
}
```

2. Quizzes

```
{
  _id: ObjectId,
  title: String,
  description: String,
  questions: [
```

```
    {
      questionText: String,
      options: [String],
      answer: String,
      type: { type: String, enum: ["MCQ", "ShortAnswer"] },
      difficulty: { type: String, enum: ["Easy", "Medium", "Hard"] }
    }
  ],
  timeLimit: Number,
  startTime: Date,
  endTime: Date,
  createdBy: ObjectId,
  createdAt: Date,
  updatedAt: Date
}
```

3. QuizAttempts

```
{
  _id: ObjectId,
  quizId: ObjectId,
  userId: ObjectId,
  responses: [
    { questionId: ObjectId, selectedOption: String, isCorrect: Boolean, timeTaken: Number }
  ],
  score: Number,
  completedAt: Date,
  createdAt: Date,
  updatedAt: Date
}
```

4. Hackathons

```
{
  _id: ObjectId,
  title: String,
  description: String,
  problems: [
    { title: String, description: String, releaseTime: Date }
  ],
  startTime: Date,
  endTime: Date,
  organizerId: ObjectId,
  teamsAllowed: Boolean,
  teamSize: Number,
  createdAt: Date,
  updatedAt: Date
}
```

5. Teams

```
{
  _id: ObjectId,
  hackathonId: ObjectId, \
  teamName: String,
  members: [ObjectId],
  createdAt: Date,
  updatedAt: Date
}
```

6. Submissions

```
{
```

```
  _id: ObjectId,
  hackathonId: ObjectId,
  teamId: ObjectId,
  submittedBy: ObjectId,
  repositoryLink: String,
  documents: [String],
  presentation: String,
  submittedAt: Date,
  score: Number,
  feedback: String,
  createdAt: Date,
  updatedAt: Date
}
```

7. Leaderboard

```
{
  _id: ObjectId,
  eventType: {type: String, enum: ["Quiz", "Hackathon"]},
  eventId: ObjectId,
  userId: ObjectId,
  score: Number,
  rank: Number,
  updatedAt: Date,
  createdAt: Date
}
```

8. Notifications

```
{
  _id: ObjectId,
  userId: ObjectId,
```

```
  message: String,
  type: { type: String, enum: ["Reminder", "LeaderboardUpdate", "EventUpdate"] },
  isRead: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

9. Rewards

```
{
  _id: ObjectId,
  userId: ObjectId,
  eventId: ObjectId,
  eventType: { type: String, enum: ["Quiz", "Hackathon"] },
  rewardType: { type: String, enum: ["Badge", "Certificate", "Monetary"] },
  rewardDetails: String,
  issuedAt: Date,
  createdAt: Date,
  updatedAt: Date
}
```

5 common queries your app will use

1. Get upcoming quizzes for participants

```
db.Quizzes.find(
  { startTime: { $gt: new Date() } },
  { title: 1, description: 1, startTime: 1, endTime: 1 }
)
```

2. Get a user's quiz attempts with scores

```
db.QuizAttempts.find(
```

```
  { userId: ObjectId("66b1f1a00000000000000001") },
  { quizId: 1, score: 1, completedAt: 1 }
)
```

3. Get leaderboard for a hackathon

```
db.Leaderboard.find(
  { eventId: ObjectId("66b1f2d00000000000000001"), eventType: "Hackathon" }
).sort({ rank: 1 }).limit(10)
```

4. Get teams for a hackathon with member count

```
db.Teams.aggregate([
  { $match: { hackathonId: ObjectId("66b1f2d00000000000000001") } },
  { $project: { teamName: 1, memberCount: { $size: "$members" } } }
])
```

5. Get unread notifications for a user

```
db.Notifications.find(
  { userId: ObjectId("66b1f1a00000000000000001"), isRead: false }
).sort({ createdAt: -1 })
```