

**MongoDB Schema Design Challenge**

**for Real Startup**

1. E-Commerce Store – Product & Orders

Scenario:

You are building a backend for an online e-commerce store (like Flipkart or Amazon). Customers browse products, place orders, and leave reviews.

Task:

Design schemas for the following collections:

* users
* products
* orders
* reviews

Schema Requirements:

* Each user must have name, email (unique), and password
* A product has a title, description, price, category, and stock
* An order must store userId, a list of productIds with quantities, total amount, and orderDate • A review links to both userId and productId, contains a rating (1–5), and a comment

Constraints to Apply:

* Use bsonType, required, enum, pattern, and minimum
* Enforce email uniqueness via index
* Validate that ratings are between 1–5

//user

db.createCollection("users", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "email", "password"],

properties: {

name: { bsonType: "string" },

email: {

bsonType: "string",

pattern: "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"

},

password: { bsonType: "string" }

}

}

}

});

//product

db.createCollection("products", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["title", "price", "category", "stock"],

properties: {

title: { bsonType: "string" },

description: { bsonType: "string" },

price: { bsonType: "number", minimum: 0 },

category: { bsonType: "string" },

stock: { bsonType: "int", minimum: 0 }

}

}

}

});

//orders

db.createCollection("orders", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["userId", "products", "totalAmount", "orderDate"],

properties: {

userId: { bsonType: "objectId" },

products: {

bsonType: "array",

items: {

bsonType: "object",

required: ["productId", "quantity"],

properties: {

productId: { bsonType: "objectId" },

quantity: { bsonType: "int", minimum: 1 }

}

}

},

totalAmount: { bsonType: "number", minimum: 0 },

orderDate: { bsonType: "date" }

}

}

}

});

//reviews

db.createCollection("reviews", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["userId", "productId", "rating"],

properties: {

userId: { bsonType: "objectId" },

productId: { bsonType: "objectId" },

rating: { bsonType: "int", minimum: 1, maximum: 5 },

comment: { bsonType: "string" }

}

}

}

});

* 1. Online Course Platform – Instructors & Students

Scenario:

You’re designing a database for an online learning platform like Udemy.

Task:

Design schemas for:

**Note**: role: [‘student’,’instructor’]

* users (can be students or instructors)
* courses
* enrollments
* lessons

Schema Requirements:

* Users must include name, email, role (student or instructor)
* A course must include title, instructorId, category, price, and createdAt
* Lessons are embedded in the course, and include title, videoURL, and duration (in minutes) • Students enroll in courses through the enrollments collection

Constraints to Apply:

* Role should be validated with enum
* Course price should be a number ≥ 0
* Lesson duration must be a number > 0

//user

db.createCollection("users", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "email", "role"],

properties: {

name: { bsonType: "string" },

email: {

bsonType: "string",

pattern: "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"

},

role: { bsonType: "string", enum: ["student", "instructor"] }

}

}

}

});

//course

db.createCollection("courses", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["title", "instructorId", "category", "price", "createdAt", "lessons"],

properties: {

title: { bsonType: "string" },

instructorId: { bsonType: "objectId" },

category: { bsonType: "string" },

price: { bsonType: "number", minimum: 0 },

createdAt: { bsonType: "date" },

lessons: {

bsonType: "array",

items: {

bsonType: "object",

required: ["title", "videoURL", "duration"],

properties: {

title: { bsonType: "string" },

videoURL: { bsonType: "string" },

duration: { bsonType: "number", minimum: 1 }

}

}

}

}

}

}

});

//enrolment

db.createCollection("enrollments", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["userId", "courseId"],

properties: {

userId: { bsonType: "objectId" },

courseId: { bsonType: "objectId" },

enrolledAt: { bsonType: "date" }

}

}

}

});

* 1. Event Booking System – Organizers & Attendees

Scenario:

You are building an event management system like Eventbrite.

Task:

Design collections for:

* users
* events
* bookings

Schema Requirements:

* Users have name, email, and role (organizer or attendee)
* Events include title, organizerId, location, startTime, endTime, and capacity
* Bookings store eventId, attendeeId, and bookingDate

Constraints to Apply:

* Validate that capacity is a positive integer
* Event startTime and endTime should be date types
* Email should follow a valid pattern and be unique

//user

db.createCollection("users", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "email", "role"],

properties: {

name: { bsonType: "string" },

email: {

bsonType: "string",

pattern: "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"

},

role: { bsonType: "string", enum: ["organizer", "attendee"] }

}

}

}

});

//event

db.createCollection("events", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["title", "organizerId", "location", "startTime", "endTime", "capacity"],

properties: {

title: { bsonType: "string" },

organizerId: { bsonType: "objectId" },

location: { bsonType: "string" },

startTime: { bsonType: "date" },

endTime: { bsonType: "date" },

capacity: { bsonType: "int", minimum: 1 }

}

}

}

});

//booking

db.createCollection("bookings", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["eventId", "attendeeId"],

properties: {

eventId: { bsonType: "objectId" },

attendeeId: { bsonType: "objectId" },

bookingDate: { bsonType: "date" }

}

}

}

});

* 1. Blogging Platform – Authors & Articles

Scenario:

You are creating a lightweight CMS/blogging system like Medium.

Task:

Design collections for:

* authors
* articles
* comments

Schema Requirements:

* Each author has name, email, and bio
* Articles contain title, content, authorId, tags (array of strings), published (boolean), and createdAt • Comments reference articleId and include userName, commentText, and postedAt

Constraints to Apply:

* Ensure article title and content are required
* published must be a boolean
* tags should be an array of strings
* Use date type for timestamps
* //authors
* db.createCollection("authors", {
* validator: {
* $jsonSchema: {
* bsonType: "object",
* required: ["name", "email"],
* properties: {
* name: { bsonType: "string" },
* email: {
* bsonType: "string",
* pattern: "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"
* },
* bio: { bsonType: "string" }
* }
* }
* }
* });
* //articles
* db.createCollection("articles", {
* validator: {
* $jsonSchema: {
* bsonType: "object",
* required: ["title", "content", "authorId", "published", "createdAt"],
* properties: {
* title: { bsonType: "string" },
* content: { bsonType: "string" },
* authorId: { bsonType: "objectId" },
* tags: {
* bsonType: "array",
* items: { bsonType: "string" }
* },
* published: { bsonType: "bool" },
* createdAt: { bsonType: "date" }
* }
* }
* }
* });
* //comments
* db.createCollection("comments", {
* validator: {
* $jsonSchema: {
* bsonType: "object",
* required: ["articleId", "userName", "commentText"],
* properties: {
* articleId: { bsonType: "objectId" },
* userName: { bsonType: "string" },
* commentText: { bsonType: "string" },
* postedAt: { bsonType: "date" }
* }
* }
* }
* });

* 1. Subscription App – Users & Plans

Scenario:

You’re building the backend for a SaaS app with subscription plans (like Notion or Canva).

Task:

Design schemas for:

* users
* plans
* subscriptions

Schema Requirements:

* Users should have email, name, and signupDate
* Plans include name, price, features, and billingCycle (monthly, yearly)
* Subscriptions include userId, planId, startDate, and isActive

Constraints to Apply:

* Validate that plan price is ≥ 0
* Billing cycle must use enum
* features should be an array of strings

//users

db.createCollection("users", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "email"],

properties: {

name: { bsonType: "string" },

email: {

bsonType: "string",

pattern: "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"

},

signupDate: { bsonType: "date" }

}

}

}

});

//plans

db.createCollection("plans", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "price", "billingCycle"],

properties: {

name: { bsonType: "string" },

price: { bsonType: "number", minimum: 0 },

features: {

bsonType: "array",

items: { bsonType: "string" }

},

billingCycle: {

bsonType: "string",

enum: ["monthly", "yearly"]

}

}

}

}

});

//subscription

db.createCollection("subscriptions", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["userId", "planId", "startDate"],

properties: {

userId: { bsonType: "objectId" },

planId: { bsonType: "objectId" },

startDate: { bsonType: "date" },

isActive: { bsonType: "bool" }

}

}

}

});