

Moksh upadhyay

B 50

2315001393

Assignment 2 answers

MongoDB Schema Design Challenge - Mongoose Schemas

1. E-Commerce Store – Product & Orders

User Schema

```
const userSchema = new mongoose.Schema({  
  name: { type: String, required: true },  
  email: { type: String, required: true, unique: true }  
  password: { type: String, required: true ,minlength:6}  
});
```

Product Schema

```
const productSchema = new mongoose.Schema({  
  title: { type: String, required: true },  
  description{ typr : String,required :true}  
  price: { type: Number, required: true, min: 0 },  
  category: { type: String, required: true,enum[ 'electronics' , 'Clothing',  
'Books' , 'other'],  
  stock: { type: Number, required: true, min: 0 }  
});
```

Order Schema

```
const orderSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
  items: [{
    productId: { type: mongoose.Schema.Types.ObjectId, ref: 'Product' },
    quantity: { type: Number, required: true, min: 1 }
  }],
  totalAmount: { type: Number, required: true, min: 0 },
  orderDate: { type: Date, default: Date.now }
});
```

Review Schema

```
const reviewSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
  productId: { type: mongoose.Schema.Types.ObjectId, ref: 'Product',
required: true },
  rating: { type: Number, required: true, min: 1, max: 5 },
  comment: String
});
```

2. Online Course Platform – Instructors & Students

User Schema

```
const userSchema = new mongoose.Schema({
  name: {type :String, required : true}
  email: { type: String, required: true, unique: true,}

  role: { type: String, enum: ['student', 'instructor'], required: true }
});
```

Course Schema

```
const courseSchema = new mongoose.Schema({
```

```
title: { type: String, required: true },
category: String,
price: { type: Number, required: true, min: 0 },
createdAt: { type: Date, default: Date.now },
});
```

Enrollment Schema

```
const enrollmentSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  courseId: { type: mongoose.Schema.Types.ObjectId, ref: 'Course' },
  enrolledAt: { type: Date, default: Date.now }
});
```

3. Event Booking System – Organizers & Attendees

User Schema

```
const userSchema = new mongoose.Schema({
  name: {type: String, required : true}
  email: { type: String, required: true, unique: true}

  role: { type: String, enum: ['organizer', 'attendee'], required: true }
});
```

Event Schema

```
const eventSchema = new mongoose.Schema({
  title: { type: String, required: true },
  organizerId: { type: mongoose.Schema.Types.ObjectId, ref: 'User',
required: true },
  location: String,
  startTime: { type: Date, required: true },
  endTime: { type: Date, required: true },
  capacity: { type: Number, required: true, min: 1 }
});
```

Booking Schema

```
const bookingSchema = new mongoose.Schema({
  eventId: { type: mongoose.Schema.Types.ObjectId, ref: 'Event', required:
true },
  attendeeId: { type: mongoose.Schema.Types.ObjectId, ref: 'User',
required: true },
  bookingDate: { type: Date, default: Date.now }
});
```

4. Blogging Platform – Authors & Articles

Author Schema

```
const authorSchema = new mongoose.Schema({
  name: String,
  email: { type: String, required: true, unique: true },
  bio: String
});
```

Article Schema

```
const articleSchema = new mongoose.Schema({
  title: { type: String, required: true },
  content: { type: String, required: true },
  authorId: { type: mongoose.Schema.Types.ObjectId, ref: 'Author',
required: true },
  tags: [String],
  published: { type: Boolean, required: true },
  createdAt: { type: Date, default: Date.now }
});
```

Comment Schema

```
const commentSchema = new mongoose.Schema({
  articleId: { type: mongoose.Schema.Types.ObjectId, ref: 'Article',
required: true },
```

```
  userName: { type: String, required: true },
  commentText: { type: String, required: true },
  postedAt: { type: Date, default: Date.now }
});
```

5. Subscription App – Users & Plans

User Schema

```
const userSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  name: String,
  signupDate: { type: Date, default: Date.now }
});
```

Plan Schema

```
const planSchema = new mongoose.Schema({
  name: { type: String, required: true },
  price: { type: Number, required: true, min: 0 },
  features: [String],
  billingCycle: { type: String, enum: ['monthly', 'yearly'], required: true }
});
```

Subscription Schema

```
const subscriptionSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
  planId: { type: mongoose.Schema.Types.ObjectId, ref: 'Plan', required:
true },
  startDate: { type: Date, default: Date.now },
  isActive: { type: Boolean, default: true }
});
```