MongoDB Schema Design

1. E-Commerce Store – Product & Orders

User Model

```
import mongoose from 'mongoose';

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

const User = mongoose.model('User', userSchema);
```

Product Model

```
const productSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
  price: { type: Number, required: true, min: 0 },
  category: { type: String, required: true },
  stock: { type: Number, required: true, min: 0 }
});

const Product = mongoose.model('Product', productSchema);
```

Order Model

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

const Order = mongoose.model('Order', orderSchema);
```

Review Model

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

```
});

const Review = mongoose.model('Review', reviewSchema);
```

---

2. Online Course Platform – Instructors & Students

User Model

```
const courseUserSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  role: { type: String, enum: ['student', 'instructor'], required: true }
}, { timestamps: true });

const CourseUser = mongoose.model('CourseUser', courseUserSchema);
```

Course Model

```
const lessonSchema = new mongoose.Schema({
  title: { type: String, required: true },
  videoURL: { type: String, required: true },
  duration: { type: Number, required: true, min: 1 }
});

const courseSchema = new mongoose.Schema({
  title: { type: String, required: true },
  instructorId: { type: mongoose.Schema.Types.ObjectId, required: true, ref:
'CourseUser' },
  category: { type: String, required: true },
  price: { type: Number, required: true, min: 0 },
  createdAt: { type: Date, required: true },
  lessons: [lessonSchema]
});

const Course = mongoose.model('Course', courseSchema);
```

Enrollment Model

```
const enrollmentSchema = new mongoose.Schema({
  studentId: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'Co
urseUser' },
  courseId: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'Cou
rse' }
});

const Enrollment = mongoose.model('Enrollment', enrollmentSchema);
```

---

3. Event Booking System – Organizers & Attendees

User Model

```
const eventUserSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  role: { type: String, enum: ['organizer', 'attendee'], required: true }
}, { timestamps: true });

const EventUser = mongoose.model('EventUser', eventUserSchema);
```

Event Model

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

const Event = mongoose.model('Event', eventSchema);
```

Booking Model

```
const bookingSchema = new mongoose.Schema({
  eventId: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'Even
t' },
  attendeeId: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'E
ventUser' },
  bookingDate: { type: Date, required: true }
});

const Booking = mongoose.model('Booking', bookingSchema);
```

---

4. Blogging Platform – Authors & Articles

Author Model

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

const Author = mongoose.model('Author', authorSchema);
```

Article Model

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

const Article = mongoose.model('Article', articleSchema);
```

Comment Model

```
const commentSchema = new mongoose.Schema({
  articleId: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'Ar
ticle' },
  userName: { type: String, required: true },
  commentText: { type: String, required: true },
  postedAt: { type: Date, required: true }
});

const Comment = mongoose.model('Comment', commentSchema);
```

---

5. Subscription App – Users & Plans

User Model

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

const SubscriptionUser = mongoose.model('SubscriptionUser', subscriptionUserS
chema);
```

Plan Model

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

const Plan = mongoose.model('Plan', planSchema);
```

Subscription Model

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

const Subscription = mongoose.model('Subscription', subscriptionSchema);
```