Name-Mahesh Goyal

University Roll no.-2315001292

Sec-C

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.

1- users

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "email", "password"],
    "properties": {
      "name": {
        "bsonType": "string"
      },
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$",
        "description": "Must be a valid email address"
      },
      "password": {
        "bsonType": "string"
      }
    }
  }
}
```

2- products

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["title", "description", "price", "category", "stock"],
    "properties": {
      "title": { "bsonType": "string" },
      "description": { "bsonType": "string" },
      "price": { "bsonType": "number", "minimum": 0 },
      "category": { "bsonType": "string" },
      "stock": { "bsonType": "int", "minimum": 0 }
    }
  }
}
```

3- orders

```
{
  "$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" }
    }
  }
}
```

4-Reviews

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["userId", "productId", "rating", "comment"],
    "properties": {
      "userId": { "bsonType": "objectId" },
      "productId": { "bsonType": "objectId" },
      "rating": { "bsonType": "int", "minimum": 1, "maximum": 5 },
      "comment": { "bsonType": "string" }
    }
  }
}
```

2. 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

1. Users Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "email", "role"],
    "properties": {
      "name": { "bsonType": "string" },
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$",
        "description": "Valid email format"
      },
      "role": {
        "enum": ["student", "instructor"],
        "description": "Role must be student or instructor"
      }
    }
  }
}
```

2. Courses Collection Schema (with Embedded Lessons)

```
{
  "$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 }
          }
        }
      }
    }
  }
}
```

3. Enrollments Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["userId", "courseId", "enrolledAt"],
    "properties": {
      "userId": { "bsonType": "objectId" },
      "courseId": { "bsonType": "objectId" },
      "enrolledAt": { "bsonType": "date" }
    }
  }
}
```

3. Event Booking System – Organizers & Attendees Scenario: You are building an event management system like Eventbrite. Task: Design collections for: GLA / W3 Grads Created By: Vivek Chand • 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

Users Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "email", "role"],
    "properties": {
      "name": { "bsonType": "string" },
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$",
        "description": "Must be a valid email address"
      },
      "role": {
        "enum": ["organizer", "attendee"],
        "description": "Role must be either 'organizer' or 'attendee'"
      }
    }
  }
}
```

Events Collection Schema

```
{
  "$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,
        "description": "Must be a positive integer"
      }
    }
  }
}
```

3. Bookings Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["eventId", "attendeeId", "bookingDate"],
    "properties": {
      "eventId": { "bsonType": "objectId" },
      "attendeeId": { "bsonType": "objectId" },
      "bookingDate": { "bsonType": "date" }
    }
  }
}
```

4. 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 GLA / W3 Grads Created By: Vivek Chand • published must be a boolean • tags should be an array of strings • Use date type for timestamps

1. Authors Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "email", "bio"],
    "properties": {
      "name": { "bsonType": "string" },
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$",
        "description": "Must be a valid email"
      },
      "bio": { "bsonType": "string" }
    }
  }
}
```

2. Articles Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["title", "content", "authorId", "tags", "published", "createdAt"],
    "properties": {
      "title": { "bsonType": "string" },
      "content": { "bsonType": "string" },
      "authorId": { "bsonType": "objectId" },
      "tags": {
       "bsonType": "array",
       "items": { "bsonType": "string" }
      },
      "published": { "bsonType": "bool" },
      "createdAt": { "bsonType": "date" }
    }
  }
}
```

3. Comments Collection Schema

```
{
 "$jsonSchema": {
  "bsonType": "object",
  "required": ["articleId", "userName", "commentText", "postedAt"],
  "properties": {
   "articleId": { "bsonType": "objectId" },
   "userName": { "bsonType": "string" },
   "commentText": { "bsonType": "string" },
   "postedAt": { "bsonType": "date" }
  }
 }
}
```

5. 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

1. Users Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["email", "name", "signupDate"],
    "properties": {
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$",
        "description": "Must be a valid email"
      },
      "name": { "bsonType": "string" },
      "signupDate": { "bsonType": "date" }
    }
  }
}
```

2. Plans Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "price", "features", "billingCycle"],
    "properties": {
      "name": { "bsonType": "string" },
      "price": {
        "bsonType": "number",
        "minimum": 0,
        "description": "Price should be zero or more"
      },
      "features": {
        "bsonType": "array",
        "items": { "bsonType": "string" }
      },
      "billingCycle": {
        "enum": ["monthly", "yearly"],
        "description": "Billing cycle must be monthly or yearly"
      }
    }
  }
}
```

3. Subscriptions Collection Schema

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["userId", "planId", "startDate", "isActive"],
    "properties": {
      "userId": { "bsonType": "objectId" },
      "planId": { "bsonType": "objectId" },
      "startDate": { "bsonType": "date" },
      "isActive": { "bsonType": "bool" }
    }
  }
}
```