Prisma

Prisma is an open-source Next-Generation ORM (Object-Relational Mapper) that makes it easy to interact with your database in a type-safe and efficient manner. While traditionally associated with relational databases (like PostgreSQL, MySQL, etc.), Prisma now offers support for MongoDB, allowing you to leverage its benefits in a NoSQL environment.

1. Install Dependencies

   Before using it, install the package

```
npm install prisma --save-dev

npm install @prisma/client
```

2. Initialize Prisma :

   This command creates the prisma directory with a schema.prisma file and a .env file. Specify MongoDB as the provider.

```
npx prisma init
```

3. Configure your Database Connection :

- Open the newly created .env file in your project root.
- Add your MongoDB connection URL to the DATABASE_URL variable:

```
DATABASE_URL="mongodb+srv://<username>:<password>@<cluste
r- url>/<database-name>?retryWrites=true&w=majority"
```

Make sure to replace the placeholders with your actual MongoDB credentials and cluster information.

4. Define your Prisma Schema:
   - Open prisma/schema.prisma.
   - Define your models (which represent MongoDB collections) and their fields.

   Example schema.prisma:

```
```

```
generator client {
  provider = "prisma-client-js"
}
datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}
model User {
    name   String
    email String
 }
```

5. Generate Prisma Client :

Every time you modify your schema.prisma file, you need to regenerate the Prisma Client to reflect those changes in your application's code.

```
npx prisma generate
```

6. Use Prisma Client in your Application :

Now you can import and instantiate PrismaClient in your application and start querying your MongoDB database.

Example server.js

```
var express = require('express');
var app = express();
var {PrismaClient}=require('@prisma/client')
var prisma=new  PrismaClient()


prisma.$connect()
    .then(() => {
      console.log("Connected to database");
    })
    .catch((err) => {
      console.error("Error connecting to database", err);
    });

app.get('/',(req,res)=>{
 var user=prisma.user.find()
 .then((users) => {
```

```
        res.json(users);
       })
       .catch((err) => {
       console.error("Error fetching users", err);
       res.status(500).send("Error fetching users");
         }
         );
         })

       app.listen(3000, () => {
          console.log("Server started at port 3000");
       });
```

Operation	MongoDB Native Query	Prisma Query
Find All	db.users.find({})	prisma.user.findMany()
Find by Condition	db.users.find({ name: "Alice" })	prisma.user.findMany({ where: { name: "Alice" } })
Find First Match	db.users.findOne({ email: "test@example.com" })	prisma.user.findFirst({ where: { email: "test@example.com" } })
Insert One	db.users.insertOne({ name: "Bob", email: "bob@example.com" })	prisma.user.create({ data: { name: "Bob", email: "bob@example.com" } })
Update One	db.users.updateOne({ name: "Bob" }, { $set: { email: "new@example.com" } })	prisma.user.updateMany({ where: { name: "Bob" }, data: { email: "new@example.com" } })
Delete One	db.users.deleteOne({ name: "Bob" })	prisma.user.deleteMany({ where: { name: "Bob" } })
Limit + Skip	db.users.find().skip(5).limit(10)	prisma.user.findMany({ skip: 5, take: 10 })
Find by ID	db.users.findOne({ _id: ObjectId("abc123") })	prisma.user.findUnique({ where: { id: "abc123" } })
Sort	db.users.find().sort({ name: 1 })	prisma.user.findMany({ orderBy: { name: 'asc' } })