Complete Prisma Setup Guide

1. Install Prisma

For new project:

```bash
npm init -y
npm install prisma @prisma/client
```

For existing project:

```bash
npm install prisma @prisma/client
```

Install as dev dependency (optional but recommended):

```bash
npm install prisma --save-dev
npm install @prisma/client
```

2. Initialize Prisma

```bash
npx prisma init
```

This creates:

- `prisma/` folder
- `prisma/schema.prisma` file
- `.env` file with DATABASE_URL

3. Configure Database Connection

Edit `.env` file:

```env
# For PostgreSQL
DATABASE_URL="postgresql://username:password@localhost:5432/your_database_name"

# For MySQL
DATABASE_URL="mysql://username:password@localhost:3306/your_database_name"

# For SQLite (for development)
DATABASE_URL="file:./dev.db"
```

For cloud databases:

```env
# Supabase
DATABASE_URL="postgresql://postgres:[PASSWORD]@[HOST]:5432/postgres"

# PlanetScale
DATABASE_URL="mysql://[USERNAME]:[PASSWORD]@[HOST]/[DATABASE_NAME]?sslaccept=strict"

# Railway
DATABASE_URL="postgresql://[USERNAME]:[PASSWORD]@[HOST]:[PORT]/[DATABASE_NAME]"
```

4. Create Your Schema

Edit `prisma/schema.prisma`:

```prisma
// This is your Prisma schema file,
// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql" // or "mysql" or "sqlite"
  url      = env("DATABASE_URL")
}

enum Role {
  USER
  ADMIN
}

model User {
  id        String   @id @default(cuid())
  email     String   @unique
  name      String
  password  String
  avatar    String   @default("/images/user-avatar.png")
  role      Role     @default(USER)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@map("users")
}
```

5. Create and Run Migration

Create migration:

```bash
npx prisma migrate dev --name init
```

This will:

- Create the migration file
- Apply it to your database
- Generate the Prisma client

For existing database (pull schema):

```bash
npx prisma db pull
npx prisma generate
```

6. Generate Prisma Client

```bash
npx prisma generate
```

7. Setup Prisma Client in Your Code

Create `lib/prisma.js` (or `lib/prisma.ts`):

For JavaScript:

```javascript
import { PrismaClient } from '@prisma/client'

const globalForPrisma = globalThis

export const prisma = globalForPrisma.prisma || new PrismaClient()

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma
```

For TypeScript:

```typescript
import { PrismaClient } from '@prisma/client'

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined
}

export const prisma = globalForPrisma.prisma ?? new PrismaClient()

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma
```

8. Using Prisma in Your Application

Basic CRUD operations:

```
javascript
```

```javascript
import { prisma } from './lib/prisma'

// Create user
const createUser = async () => {
  const user = await prisma.user.create({
    data: {
      email: 'john@example.com',
      name: 'John Doe',
      password: 'hashedPassword123',
      role: 'USER'
    }
  })
  return user
}

// Find user
const getUser = async (email) => {
  const user = await prisma.user.findUnique({
    where: { email }
  })
  return user
}

// Update user
const updateUser = async (id, data) => {
  const user = await prisma.user.update({
    where: { id },
    data
  })
  return user
}

// Delete user
const deleteUser = async (id) => {
  const user = await prisma.user.delete({
    where: { id }
  })
  return user
}

// Get all users
const getAllUsers = async () => {
  const users = await prisma.user.findMany()
  return users
}
```

9. Common Commands

```bash
# View your data in Prisma Studio
npx prisma studio

# Reset database (careful!)
npx prisma migrate reset

# Deploy migrations to production
npx prisma migrate deploy

# Format schema file
npx prisma format

# Validate schema
npx prisma validate

# Push schema changes without migration (for prototyping)
npx prisma db push
```

10. Package.json Scripts (Optional)

Add these to your `package.json`:

```json
{
  "scripts": {
    "db:migrate": "npx prisma migrate dev",
    "db:generate": "npx prisma generate",
    "db:studio": "npx prisma studio",
    "db:push": "npx prisma db push",
    "db:reset": "npx prisma migrate reset"
  }
}
```

11. Environment Variables for Different Environments

Development (.env.local):

```env
DATABASE_URL="postgresql://localhost:5432/myapp_dev"
```

Production:

Set DATABASE_URL in your hosting platform (Vercel, Netlify, etc.)

12. Troubleshooting

Common issues:

1. **"Client not generated"**:

   ```bash
   npx prisma generate
   ```

2. **Migration issues**:

   ```bash
   npx prisma migrate reset
   npx prisma migrate dev
   ```

3. **Connection issues**:
   - Check DATABASE_URL format
   - Ensure database server is running
   - Verify credentials

4. **Schema changes not reflected**:

   ```bash
   npx prisma db push  # For development
   # OR
   npx prisma migrate dev --name your_change_name
   ```

Next Steps

1. Set up your database (PostgreSQL, MySQL, etc.)
2. Configure your DATABASE_URL
3. Run the migrations
4. Start building your application with Prisma!

Remember to:

- Always backup your database before major changes
- Use migrations for production deployments
- Keep your schema.prisma file in version control
- Never commit your .env file with real credentials