

WATCH FULL VIDEO
ON **YOUTUBE**



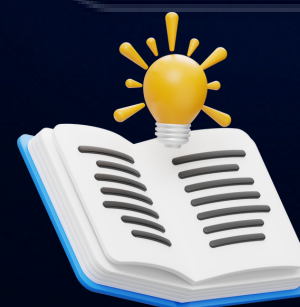
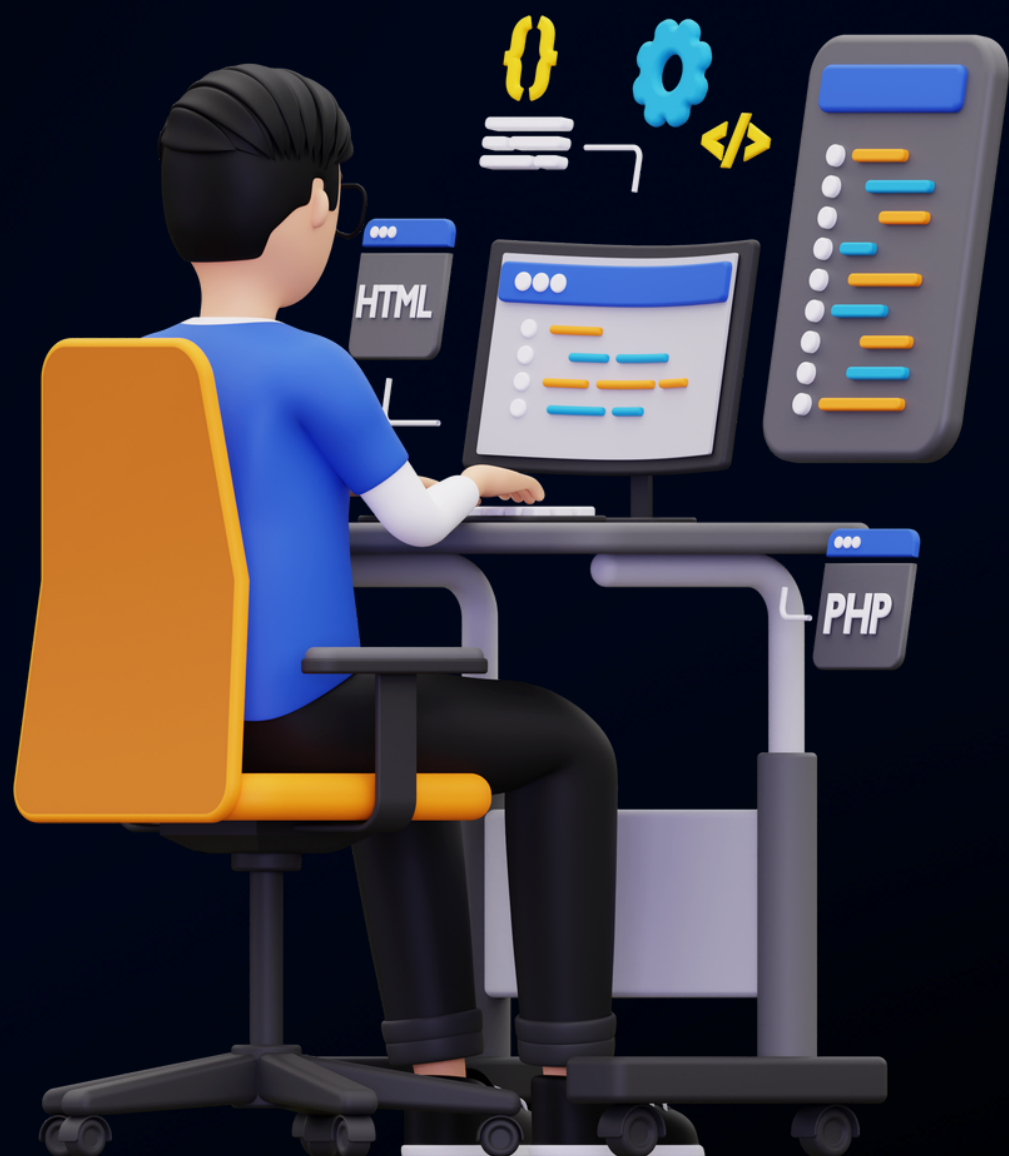
SUBSCRIBE



Next js 13 CRUD with PRISMA , Mongodb and Cloudinary

NEXT_{.JS}

13



Developers' Prisma Handbook Manual

By JB WEB DEVELOPER




Swipe




What is PRISMA?



Powered By Desishub Coding School



Prisma is a database toolkit that helps you connect to and manage your databases using a unified API. It simplifies CRUD operations by providing a type-safe and idiomatic API to interact with your database.



It acts as an intermediary layer between the application and the database, providing an intuitive API for performing CRUD operations and handling data modeling.



Step 1: Install Next js 13 project



Powered By Desishub Coding School

```
npx create-next-app@latest prisma-crud
```



Step2 - Set up Prisma in Next js 13



Powered By Desishub Coding School

//Install prisma CLI

1

npm install prisma --save-dev

//Install prisma Client

2

npm install @prisma/client

//Initialise prisma with mongodb

3

npx prisma init --datasource-provider mongodb



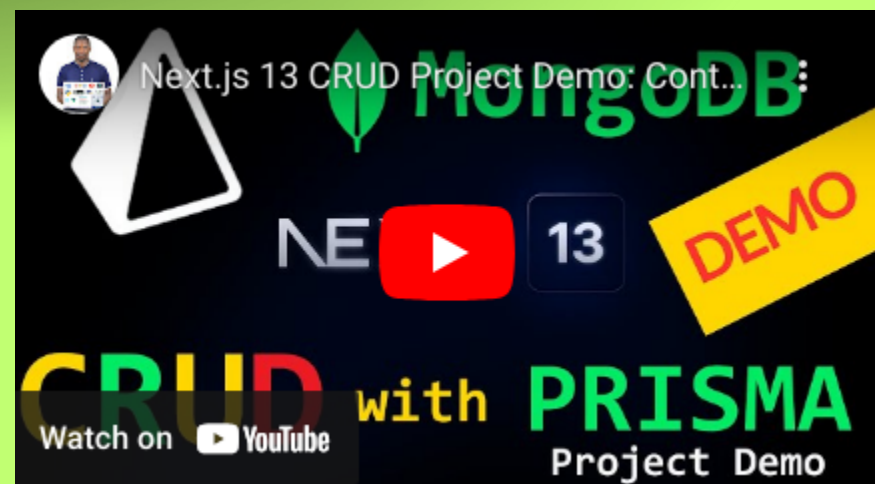
Watch the Full Video on **youtube**



Powered By Desishub Coding School

You can also Watch Full Video on
Youtube, Link Above

[Click to Watch](#)



WATCH FULL VIDEO

Youtube - Link in the description above





Step2 - Set up Prisma in Next js 13

Continued



Powered By Desishub Coding School

//Get Database URL from mongodb
and update it in .env file

4

```
DATABASE_URL="mongodb+srv://test:test@cluster0.ns1yp  
.mongodb.net/myFirstDatabase"
```

//Model your data in prisma schema

5

```
model Contact{  
  id String @id @default(auto()) @map("_id") @db.ObjectId  
  name String  
  phone String  
  image String  
  email String  
}
```




Step2 - Set up Prisma in Next js 13

Continued



Powered By Desishub Coding School

//Push the Models to the database

6

npx prisma db push

//Run prisma studio to see your db in browser

7

npx prisma studio

//Create a prisma client Instance

8

npx prisma generate



Step2 - Set up Prisma in Next js 13

Continued



Powered By Desishub Coding School

//Avoid creating multiple prisma instances
create a global instance that we will use
throughout the app.
Create a file called db.js in libs folder

9

```
import { PrismaClient } from "@prisma/client";  
const db = globalThis.prisma || new PrismaClient();
```

```
if (process.env.NODE_ENV !== "production")  
  globalThis.prisma = db;
```

```
export default db;
```




Step 3. Create the api folder



Powered By Desishub Coding School

1. In the api folder:

- create: `/contacts/route.js`
- this will handle get-all-contacts
- and also create-new-contact endpoints

2. In the contacts folder in api folder:

- create: `[id]/route.js`
- this will handle get-single-contact
update-contact and delete-contact endpoints



Step 4. Create the api endpoints



Powered By Desishub Coding School

C => POST Endpoint

R => GET Endpoint (**Get all** and **Get Single**)

U => PATCH Endpoint

D => DELETE Endpoint



Watch the Full Video on **youtube**



Powered By Desishub Coding School

You can also Watch Full Video on
Youtube, Link Above

[Click to Watch](#)





Prisma POST Endpoint

This endpoint creates a new record in base

Dont forget to import db from @utils/db

```
//Creating a new Contact
export async function POST(request) {
  try {
    const { name, phone, email, profile } = await request.json();
    const newContact = await db.contact.create({
      data: { name, phone, email, profile, },
    });

    return NextResponse.json(newContact, { status: 201, }); }
  catch (error) {
    console.log("Error while creating contact", error);
    return NextResponse.json( {
      message: "Failed to create contact",
      error: error.message, }, { status: 500, } );
  } }
```



Prisma GET All Endpoint

This endpoint gets all the records from db

Dont forget to import db from @utils/db

```
//Getting all records
export async function GET() {
  try {
    const contacts = await db.contact.findMany();
    return NextResponse.json(contacts); }
  catch (error) {
    console.log("Failed to fetch Contacts", error);
    return NextResponse.json( {
      message: "Failed to fetch contacts",
      error: error.message, }, { status: 500,
    } );
  } }
```



Prisma GET Single Endpoint

This endpoint fetches single record from database

Dont forget to import db from @utils/db



```
//Getting a single Record
export async function GET(request, { params: { id } }) {
  try { const contact = await db.contact.findUnique({
    where: { id }, });
    if (!contact) {
      return NextResponse.json( {
        message: "Contact NOT FOUND", },
        { status: 404, } ); }

    return NextResponse.json(contact); }
  catch (error) {
    console.log("Failed to fetch Contact", error);
    return NextResponse.json( {
      message: "Failed to fetch contact",
      error: error.message, }, { status: 500, } );
  } }
```




Prisma PATCH Endpoint

This endpoint updates single record in database
Dont forget to import db from @utils/db

```
export async function PATCH(request, { params: { id } }) {
  try {
    const { name, phone, email, profile } = await request.json();
    const updatedContact = await db.contact.update({
      where: { id },
      data: { name, phone, email, profile, }, });
    if (!updatedContact) {
      return NextResponse.json( {
        message: "Contact NOT FOUND", },
        { status: 404, } ); }
    return NextResponse.json(updatedContact); }
  catch (error) { console.log("Failed to update Contact", error);
    return NextResponse.json( {
      message: "Failed to update contact",
      error: error.message, }, { status: 500, } );
  } }
```



Prisma DELETE Endpoint

This endpoint deletes a single record from db
Dont forget to import db from @utils/db

```
//Deleting a single Record
export async function DELETE(request, { params: { id } }) {
  try {
    await db.contact.delete({
      where: { id },
    });
    return NextResponse.json({
      message: "Contact deleted successfully",
    });
  } catch (error) {
    console.log("Failed to delete Contact", error);
    return NextResponse.json({
      message: "Failed to Delete contact",
      error: error.message,
    }, { status: 500 });
  }
}
```



FOR Cloudinary Image Upload

Watch the Video or Check in my Recent Posts



Powered By Desishub Coding School

You can also Watch Full Video on
Youtube, Link Above

[Click to Watch](#)



 **WATCH FULL VIDEO**
Yotube - Link in the description above





Follow for More



JB WEB DEVELOPER



WATCH FULL VIDEO

Youtube - Link in the description above

