BACKEND CODE

CART

```
import connectDB from "@/config/db";
import User from "@/models/User";
import { getAuth } from "@clerk/nextjs/server";
import { NextResponse } from "next/server";



export async function GET(request) {
try {
const { userId } = getAuth(request);
await connectDB();
const user = await User.findById(userId);

const { cartItems } = user;

return NextResponse.json({ success: true, cartItems });

} catch (error) {
return NextResponse.json({ success: false, message: error.message});
}
}
```

UPDATE

```
import connectDB from '@/config/db';
import User from '@/models/User';
import { getAuth } from '@clerk/nextjs/server';
import { NextResponse } from 'next/server';

export async function POST(request) {
try {
const { userId } = getAuth(request);

const { cartData } = await request.json()

await connectDB()
const user = await User.findById(userId)

user.cartItems = cartData
await user.save()

return NextResponse.json({ success: true });

} catch (error) {
return NextResponse.json({ success: false, message: error.message });
}
}
```

INNGEST

```
import { serve } from "inngest/next";
import { inngest, syncUserCreation, syncUserDeletion, syncUserUpdation } from "@/config/inngest";


// Create an API that serves zero functions
export const { GET, POST, PUT } = serve({
client: inngest,
functions: [
syncUserCreation,
syncUserUpdation,
syncUserDeletion
],
});
```

PRODUCT ADD

```
import { v2 as cloudinary } from "cloudinary";
import { getAuth } from "@clerk/nextjs/server";
import authSeller from "@/lib/authSeller";
import { NextResponse } from "next/server";
import connectDB from "@/config/db";
import Product from "@/models/Product";



cloudinary.config({
cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
api_key: process.env.CLOUDINARY_API_KEY,
api_secret: process.env.CLOUDINARY_API_SECRET,
});

export async function POST(request) {
try {

const { userId } = getAuth(request);
const isSeller = await authSeller(userId);


if (!isSeller) {
return NextResponse.json({ success: false, message: "Not authorized" });
}



const formData = await request.formData();


const name = formData.get("name");
const description = formData.get("description");
const category = formData.get("category");
const price = formData.get("price");
const offerPrice = formData.get("offerPrice");
const files = formData.getAll("images");

if (!files || files.length === 0) {
return NextResponse.json({ success: false, message: "No files uploaded" });
}

//
const uploadedImages = await Promise.all(
files.map(async (file) => {
const arrayBuffer = await file.arrayBuffer();
const buffer = Buffer.from(arrayBuffer);

return new Promise((resolve, reject) => {
const stream = cloudinary.uploader.upload_stream(
{ resource_type: "auto" },
(error, result) => {
if (error) reject(error);
else resolve(result);
}
);
stream.end(buffer);
});
})
);

const imageUrls = uploadedImages.map((result) => result.secure_url);

//
await connectDB();
const newProduct = await Product.create({
userId,
name,
description,
category,
price: Number(price),
offerPrice: Number(offerPrice),
image: imageUrls,
date: Date.now(), // spelling fix: was "data"
});

return NextResponse.json({
success: true,
message: "Upload successful",
newProduct,
});
} catch (error) {
console.error("Upload Error:", error);
return NextResponse.json({ success: false, message: error.message });
}
}
```

PRODUCT LIST

```
import Product from "@/models/Product"
import connectDB from "@/config/db"
import { NextResponse } from "next/server"

export async function GET(request) {
try {

// Connect to DB
await connectDB()

// Fetch seller's products only (optional: filter by userId)
// Agar sab products chahiye to remove filter
const products = await Product.find({ })

return NextResponse.json({ success: true, products })
} catch (error) {
return NextResponse.json({ success: false, message: error.message })
}
}
```

SELLER LIST

```
import Product from "@/models/Product"
import connectDB from "@/config/db"
import authSeller from "@/lib/authSeller"
import { getAuth } from "@clerk/nextjs/server"
import { NextResponse } from "next/server"

export async function GET(request) {
try {
// Authenticate user from request
const { userId } = getAuth(request)

// Check if user is seller
const isSeller = await authSeller(userId)
if (!isSeller) {
return NextResponse.json({ success: false, message: "not authorized" })
}

// Connect to DB
await connectDB()

// Fetch seller's products only (optional: filter by userId)
// Agar sab products chahiye to remove filter
const products = await Product.find({ userId })

return NextResponse.json({ success: true, products })
} catch (error) {
```

```
return NextResponse.json({ success: false, message: error.message })
}
}
```

ADD ADDRESS

```
import connectDB from "@/config/db";
import Address from "@/models/Address";
import { getAuth } from "@clerk/nextjs/server";
import { NextResponse } from "next/server";



export async function POST(request) {
try {

const { userId } = getAuth(request);
const {address} = await request.json()

await connectDB()
const newAddress = await Address.create({...address,userId})

return NextResponse.json({ success: true, message: "Address added successfully",newAddress})
} catch (error) {
return NextResponse.json({ success: false, message: error.message});
}
}
```

DATA

```

import connectDB from "@/config/db";
import User from "@/models/User";
import { getAuth } from "@clerk/nextjs/server";
import { NextResponse } from "next/server";

export async function GET(request) {
try {
const { userId } = getAuth(request);

await connectDB();

// Clerk ID se search karo (not MongoDB _id)
const user = await User.findOne({ clerkId: userId });

if (!user) {
return NextResponse.json({ success: false, message: "User Not Found" });
}

return NextResponse.json({ success: true, user }); // user (lowercase U)
} catch (error) {
return NextResponse.json({
```

```
success: false,
message: error.message || "User Not Found",
});
}
}
```

GET ADDRESS

```
import connectDB from "@/config/db";
import Address from "@/models/Address";
import { getAuth } from "@clerk/nextjs/server";
import { NextResponse } from "next/server";

export async function GET(request) {
try {
const { userId } = getAuth(request);
await connectDB();

const addresses = await Address.find({userId})

return NextResponse.json({ success: true, addresses });

} catch (error) {
return NextResponse.json({ success: false, message: error.message });
}
}
```

ORDER

```
let orders = [];

export default function handler(req, res) {
console.log("API hit:", req.method, req.body);
if (req.method === 'POST') {
const newOrder = req.body;
orders.push(newOrder);
return res.status(201).json({ message: 'Order saved', order: newOrder });
}
if (req.method === 'GET') {
return res.status(200).json({ orders });
}
res.status(405).json({ message: 'Method not allowed' });
}
```

LAYOUT

```
import { Outfit } from "next/font/google";
import "./globals.css";
import { AppContextProvider } from "@/context/AppContext";
import { Toaster } from "react-hot-toast";
import { ClerkProvider } from "@clerk/nextjs";
```

```
const outfit = Outfit({ subsets: ['latin'], weight: ["300", "400", "500"] })


export const metadata = {
title: "QuickCart - GreatStack",
description: "E-Commerce with Next.js ",
};


export default function RootLayout({ children }) {
return (
<ClerkProvider>
<html lang="en">
<body className={`${outfit.className} antialiased text-gray-700`} >
<Toaster />
<AppContextProvider>
{children}
</AppContextProvider>
</body>
</html>
</ClerkProvider>
);
}
```

MONGO BD

```
import mongoose from "mongoose";



let cached = global.mongoose


if (!cached) {
cached = global.mongoose = { conn: null, promise: null}
}
async function connectDB() {


if (cached.conn) {
return cached.conn
}
if(!cached.promise){
const opts = {
bufferCommands:false
}
cached.promise = mongoose.connect(`${process.env.MONGODB_URI}/quickcart`,opts).then(mongoose =>{
return mongoose
})


}


cached.conn =await cached.promise
return cached.conn
}

export default connectDB
```

INNGEST

```
import { Inngest } from "inngest";
import connectDB from "./db";
import User from "@/models/User";

// Create a client to send and receive events
export const inngest = new Inngest({ id: "quickcart-next" });

// inngest function to save user data to a database

export const syncUserCreation = inngest.createFunction(
{
id:'sync-user-from-clerk'
},
{ event: 'clerk/user.created'},
async({event})=>{
const { id, first_name, last_name, email_addresses, image_url } = event.data
const userData = {
_id:id,
email: email_addresses[0].email_address,
name: first_name + ' ' + last_name,
image_url:image_url
}

await connectDB()
await User.create(userData)

}
)

//Inngest function to update user data in database
export const syncUserUpdation = inngest.createFunction(
{
id: 'update-user-from-clerk'
},
{ event: 'clerk/user.updated' },
async({event})=>{
const { id, first_name, last_name, email_addresses, image_url } = event.data
const userData = {
_id:id,
email: email_addresses[0].email_address,
name: first_name + ' ' + last_name,
image_url:image_url
}
await connectDB()
await User.findByIdAndUpdate(id,userData)
}
)

//Inngest function to delete user from database
export const syncUserDeletion = inngest.createFunction(
{
id: 'delete-user-with-clerk'
},
{ event: 'clerk/user.deleted'},
async ({event}) =>{
const {id} =event.data

await connectDB()
await User.findByIdAndDelete(id)
}
)
```

AUTH SELLER

```
import { clerkClient } from '@clerk/nextjs/server';
import { NextResponse } from 'next/server';

const authSeller = async (userId) => {
try {

const client = await clerkClient()
const user = await client.users.getUser(userId)

if (user.publicMetadata.role === 'seller') {
return true;
} else {
return false;
}
} catch (error) {
return NextResponse.json({ success: false, message: error.message });
}
}

export default authSeller;
```

ADDRESS

```
import mongoose from "mongoose";

const addressSchema = new mongoose.Schema({
userId: { type: String, required: true },
fullName: { type: String, required: true },
phoneNumber: { type: String, required: true },
pinCode: { type: String, required: true },
area: { type: String, required: true },
city: { type: String, required: true },
state: { type: String, required: true },
})
```

```
const Address = mongoose.models.Address || mongoose.model('Address', addressSchema);

export default Address;
```

PRODUCT

```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema({
userId: { type: String, required: true, ref: "user"},
name: { type: String, required: true},
description: { type: String, required: true},
price: { type: Number, required: true},
offerPrice: { type: Number, required: true},
image: { type: Array, required: true},
category: { type: String, required: true},
date : { type: Number, required: true}

})

const Product = mongoose.models.product || mongoose.model('product',productSchema)

export default Product
```

USER

```
import mongoose from "mongoose";



const userSchema = new mongoose.Schema({
_id: { type : String, required:true},
name: { type : String, required:true},
email: { type : String, required:true, unique:true},
imageUrl:{ type : String, required:true},
cartItem:{ type : Object,default:{}}
},{ minimize: false })

const User = mongoose.models.user || mongoose.model('user', userSchema)

export default User
```