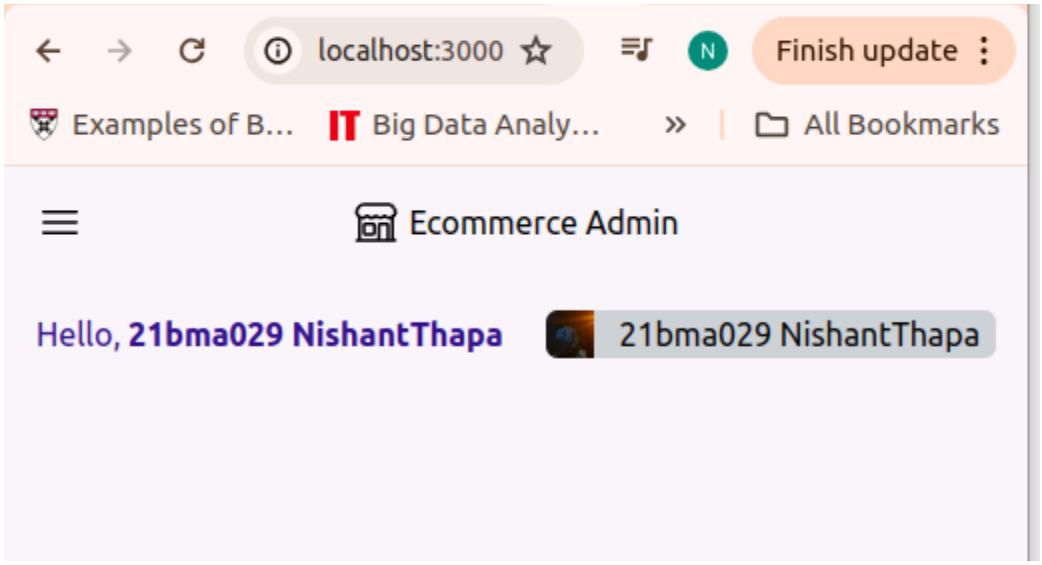
E-Commerce-Admin

To build an E-commerce website we should have a products in our Database, hence we will start with the E-commerce admin.

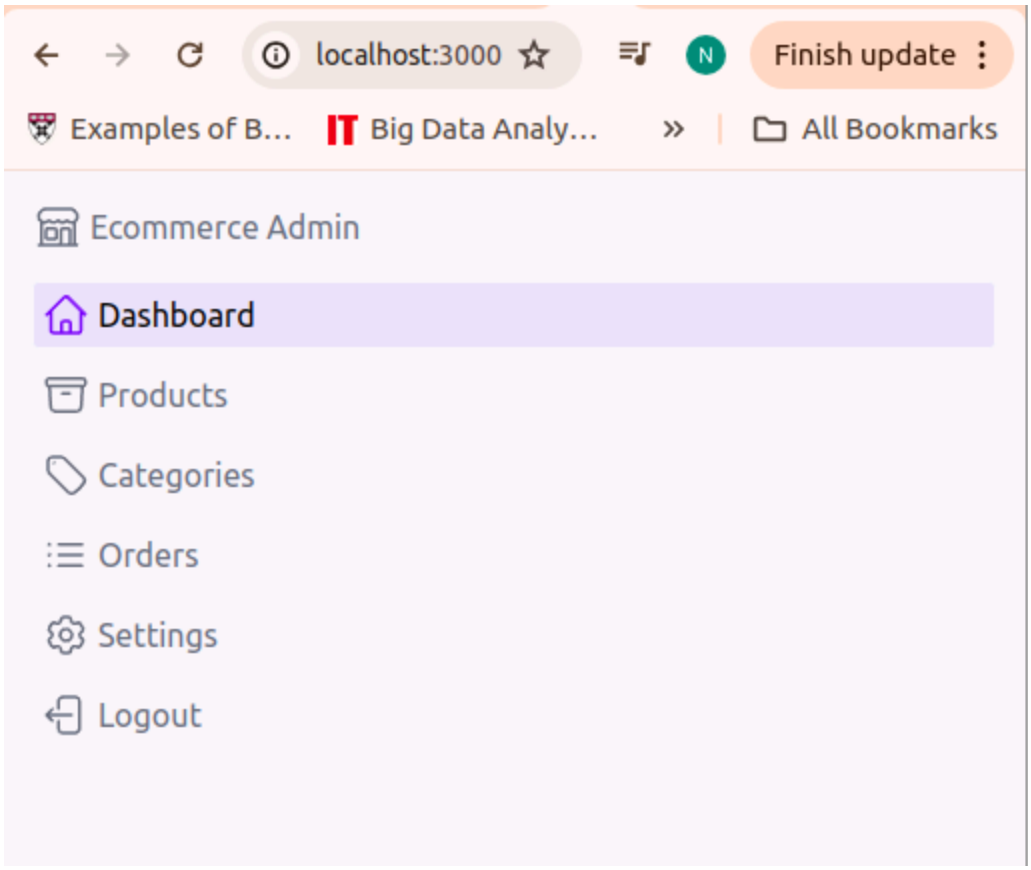
Ecommerce - Admin - UI/UX

Home page

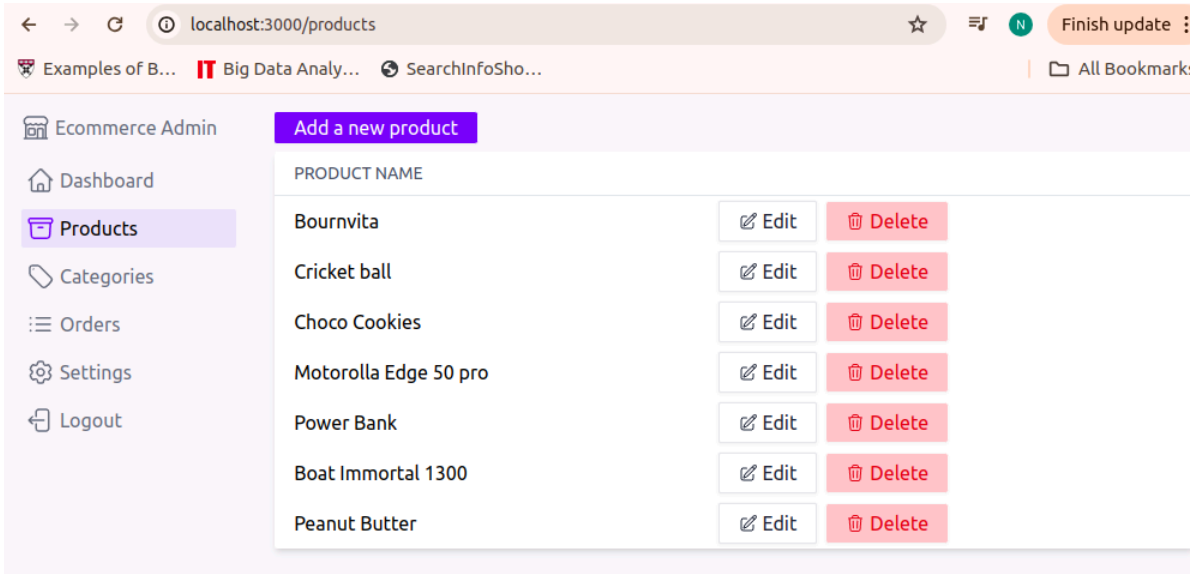
Home page - Mobile View



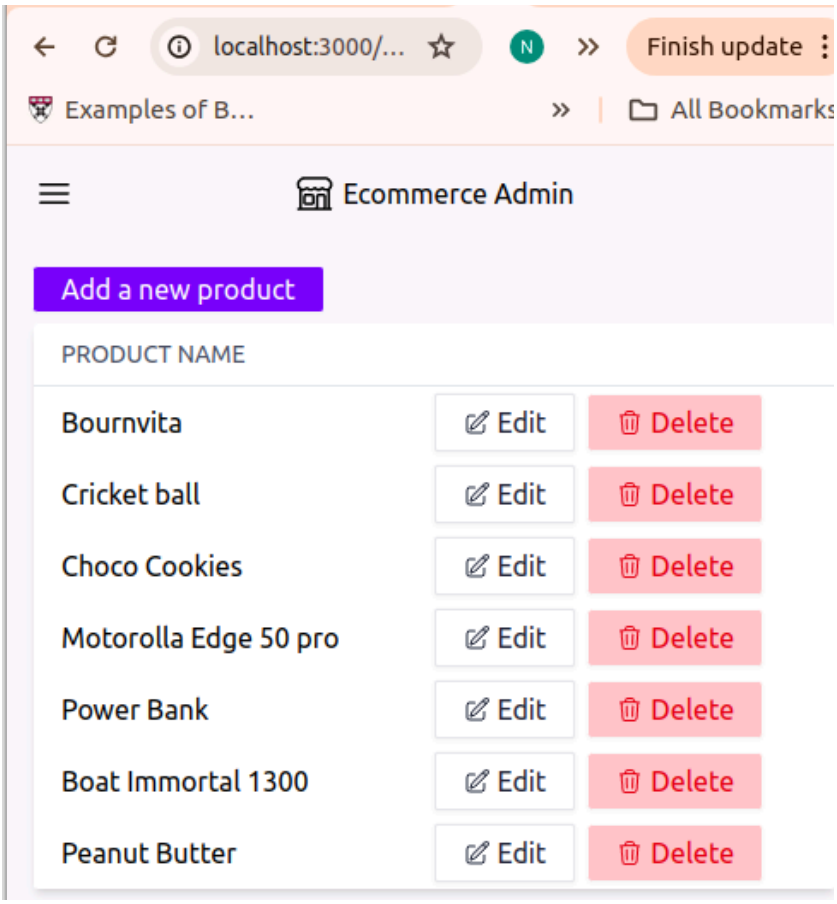
Menu



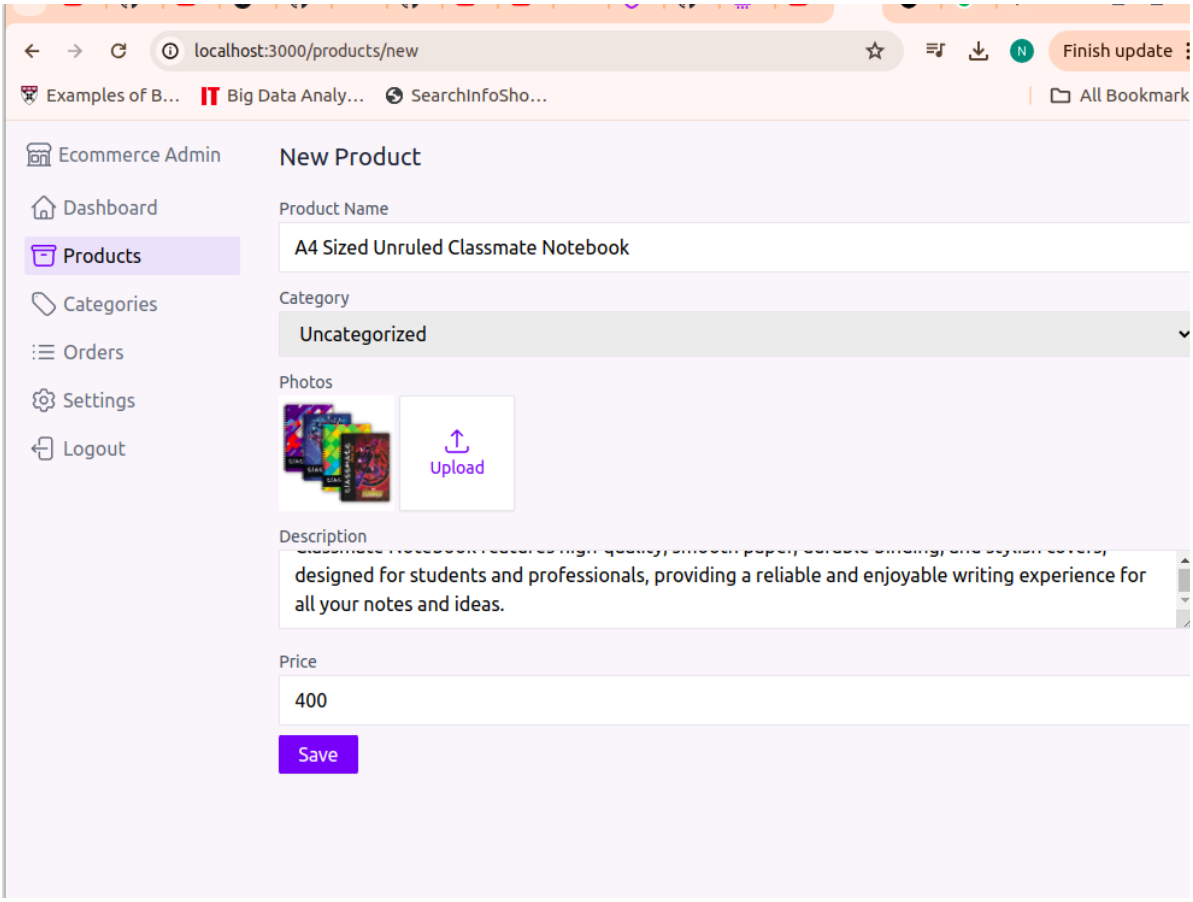
Products page



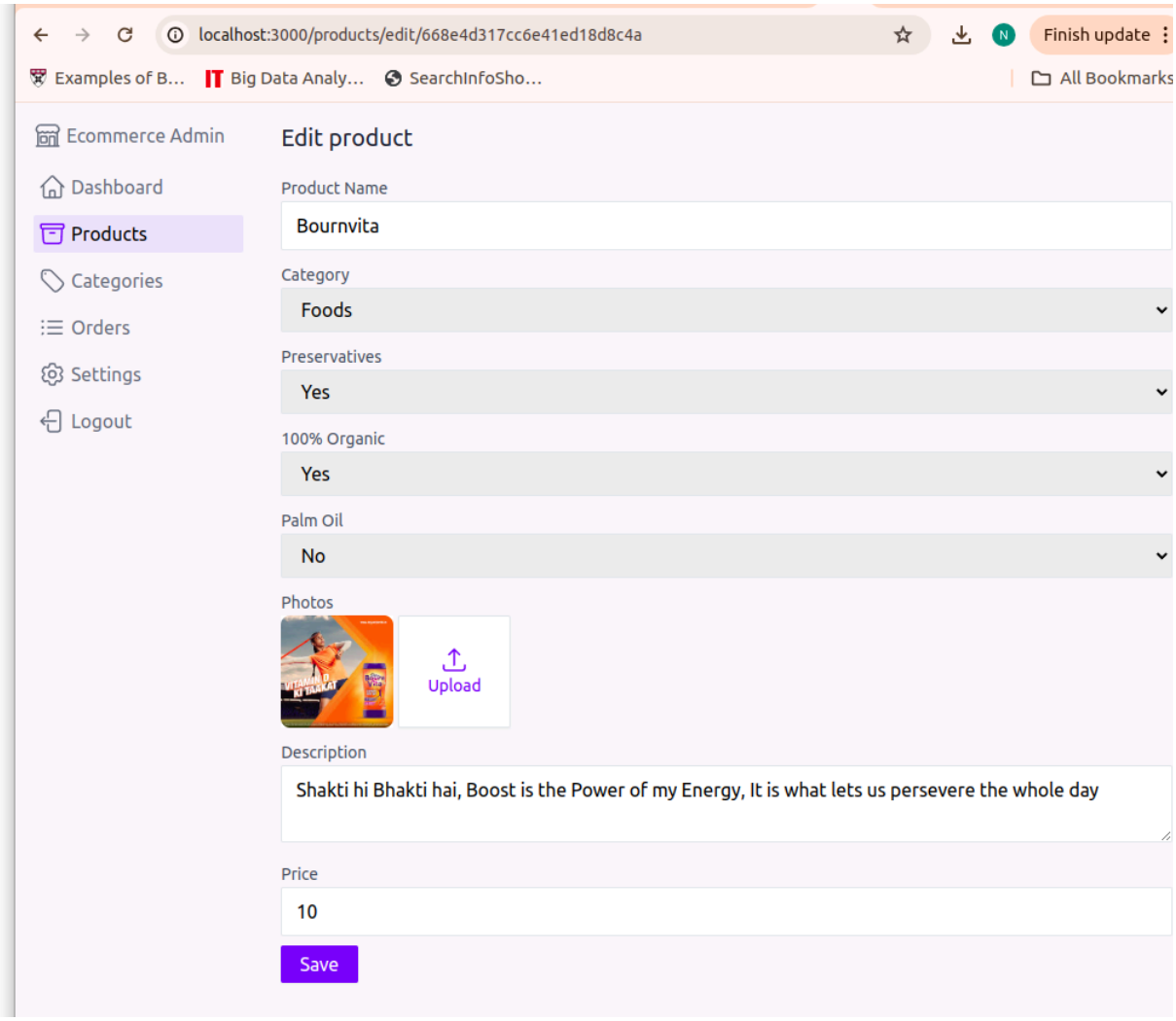
Products page - Mobile View



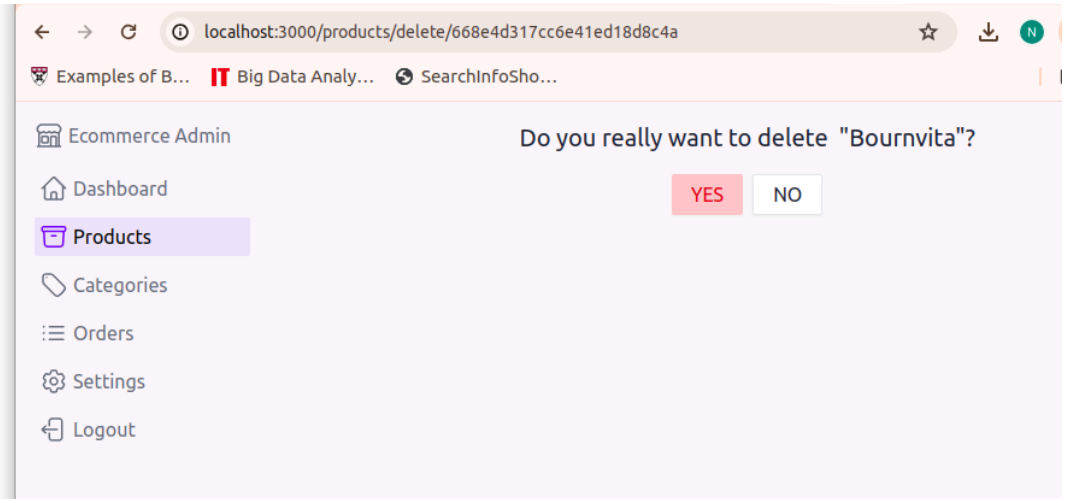
Products – Add New Product



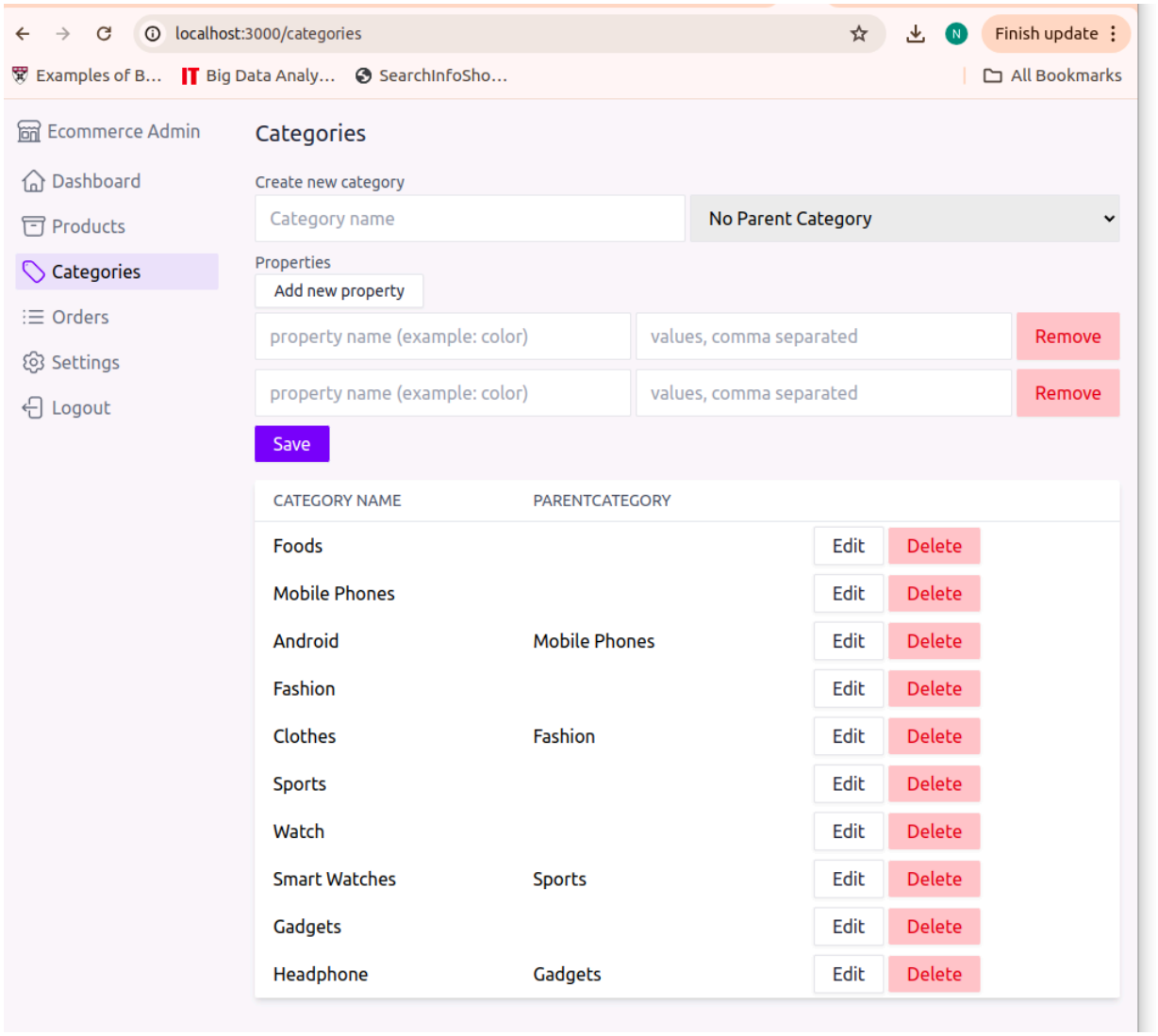
Products - Edit product



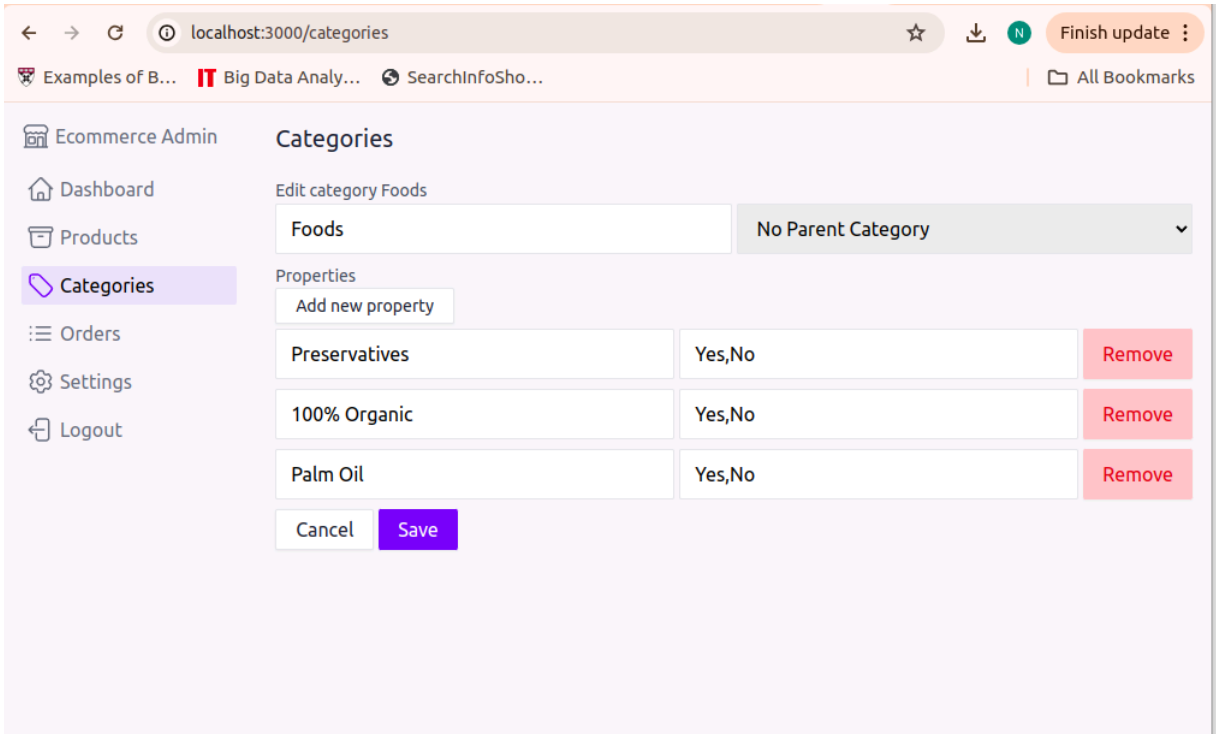
Products - Delete Page



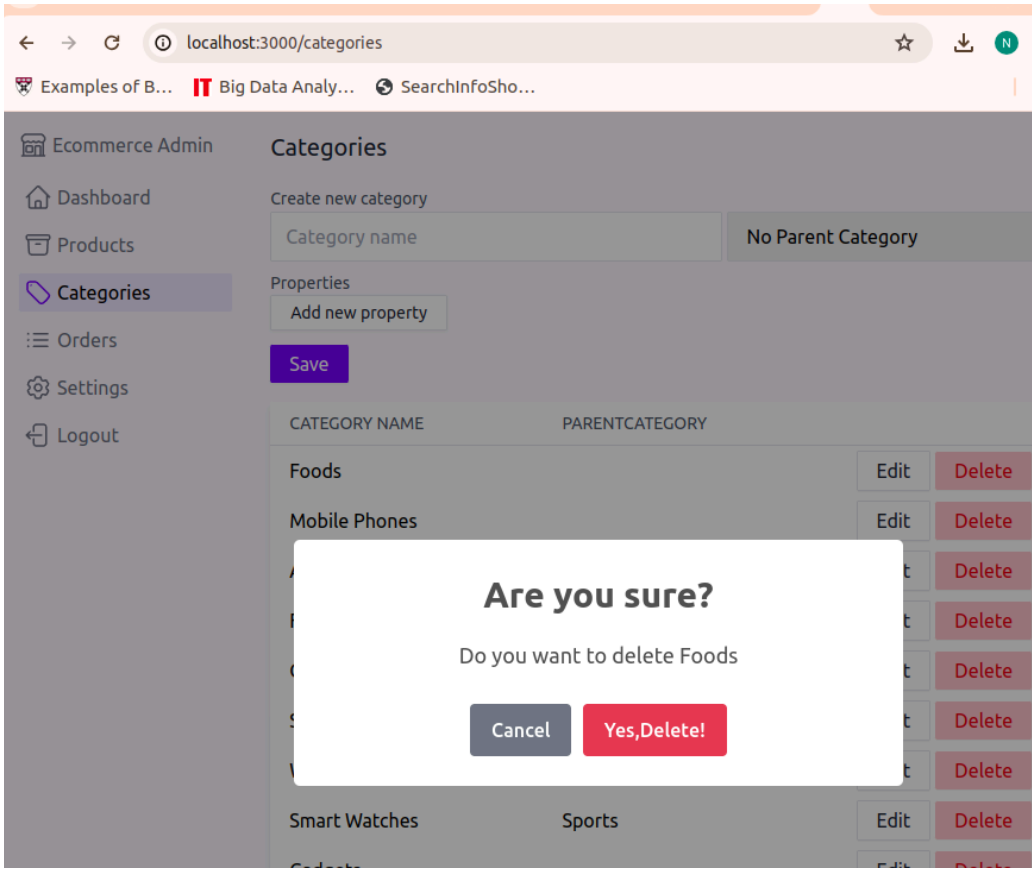
Category Page:



Category -> Edi



Category Delete



E-commerce Admin

Creating a new NextJs app:

npm create next-app@latest ./

For this project we will use Pages router

to run local server

npm run dev

Login Page

pages/index.js

```
export default function Home() {
 return (
   <div className="bg-blue-900 w-screen h-screen flex items-center">
     <div className="text-center w-full">
       <button className="bg-white p-2 rounded-lg px-4 ">Login with
Google</button>
     </div>
   </div>
 );
}
```

Now lets add authentication with NextAuth.js

NextAuth.js

Create a new route `/pages/api/auth/[...nextauth].js`

Paranthesis are such that all the request to auth will go to this file

Add next-auth library

npm install next-auth

Add Server code(backend) **from NextAuth website**

```
import NextAuth from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'


export default NextAuth({
 providers: [
   // OAuth authentication providers...
   GoogleProvider({
     clientId: process.env.GOOGLE_ID,
     // now we have to provide GOOGLE_ID and GOOGLE_SECRET by .env file
```

```
    clientSecret: process.env.GOOGLE_SECRET
  }),
 ]
})
```

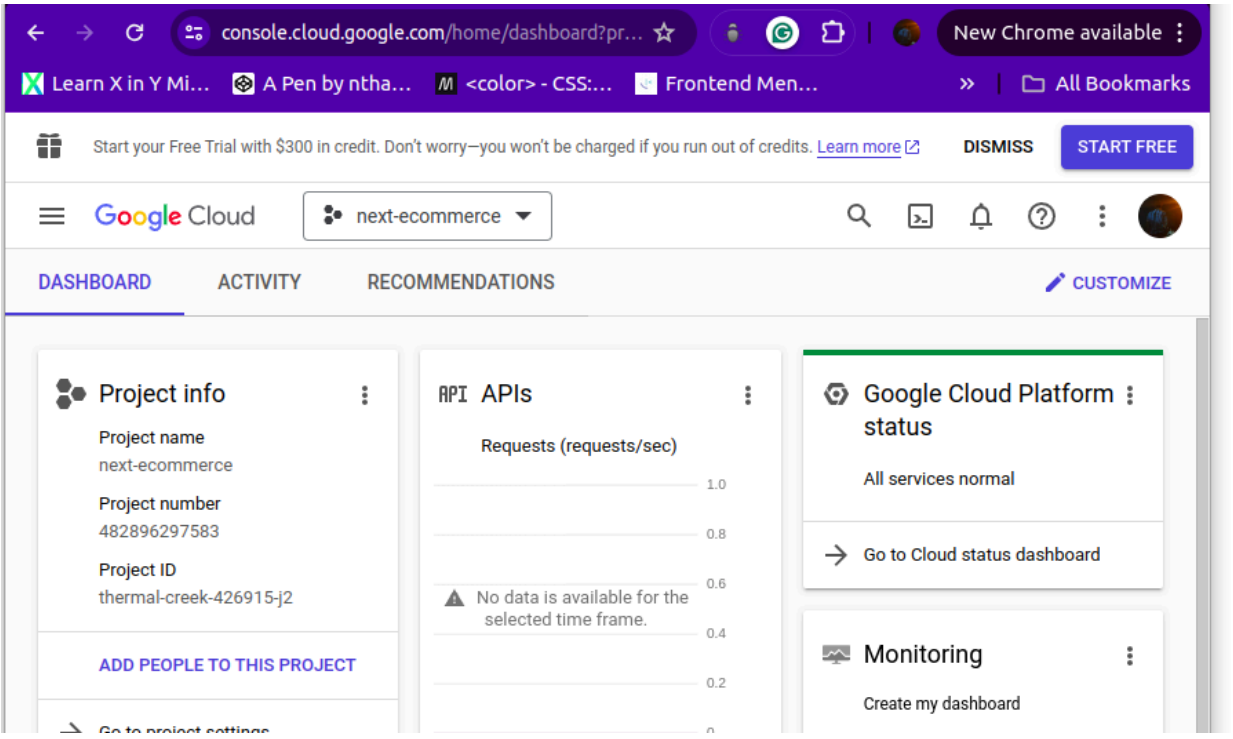
We need GOOGLE_ID and GOOGLE_SECRET

```
GOOGLE_ID=
GOOGLE_SECRET=
```

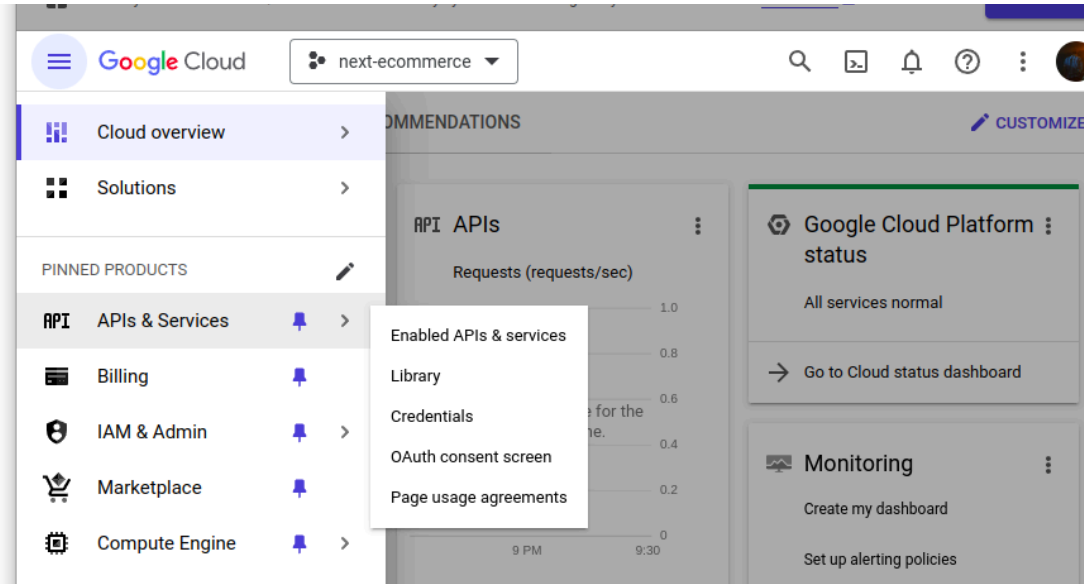
Go to google cloud console

https://console.cloud.google.com/welcome?project=thermal-creek-426915-j2

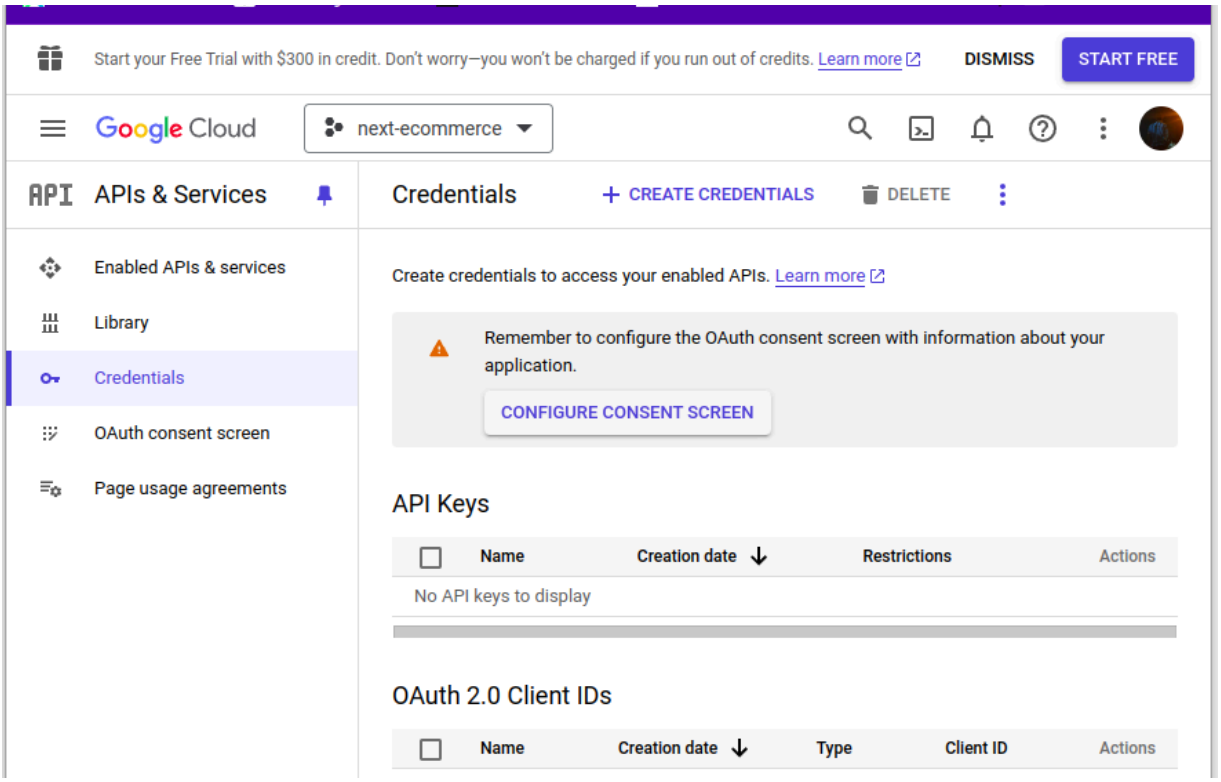
Make a new project and then select the project



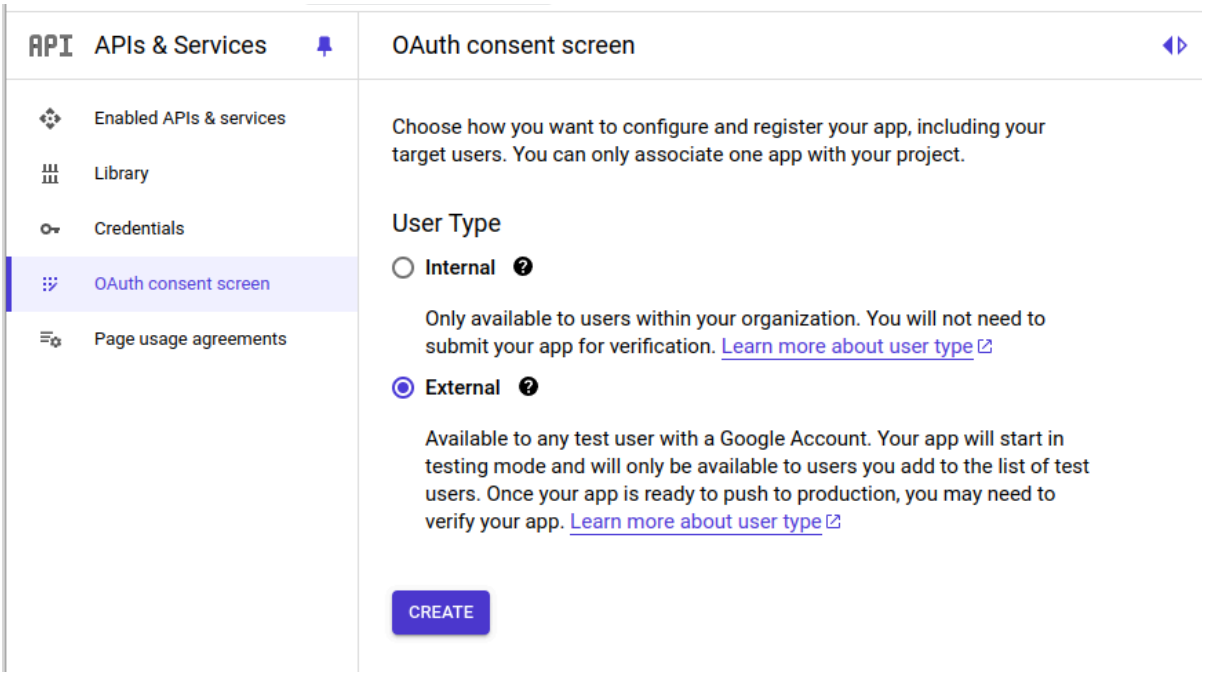
Click on Hamburger Menu



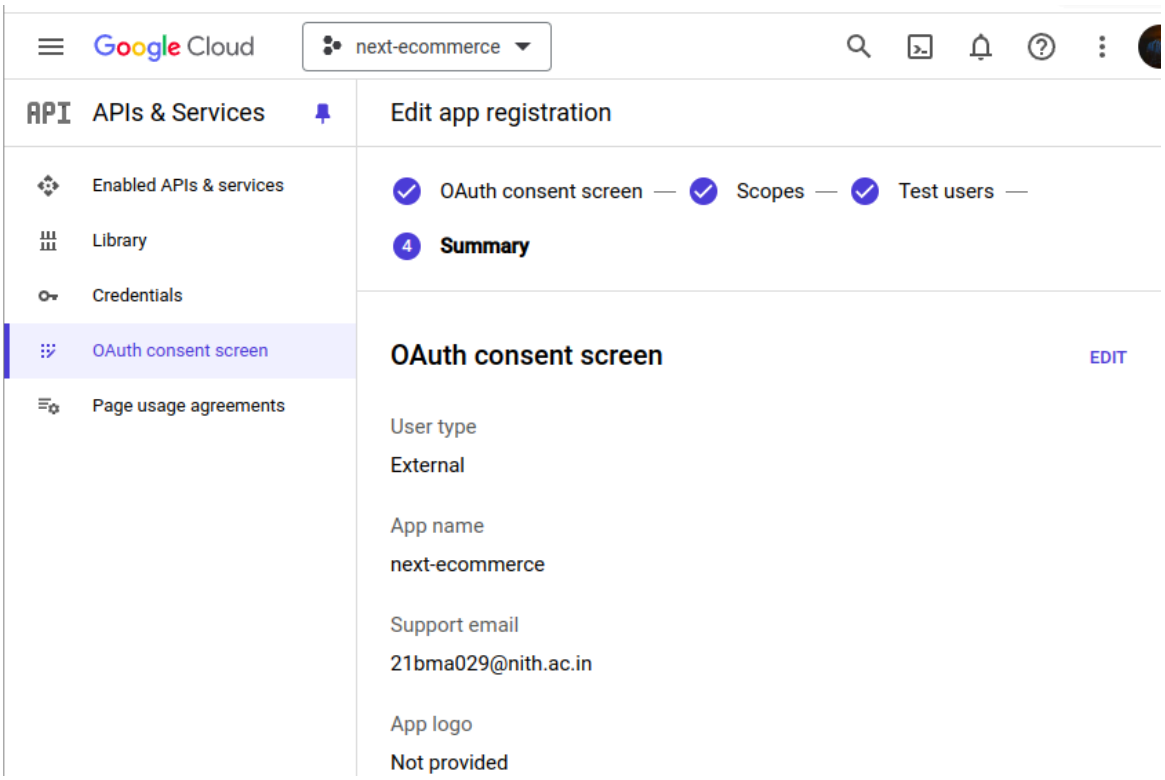
Click on Credentials and then select configure consent screen



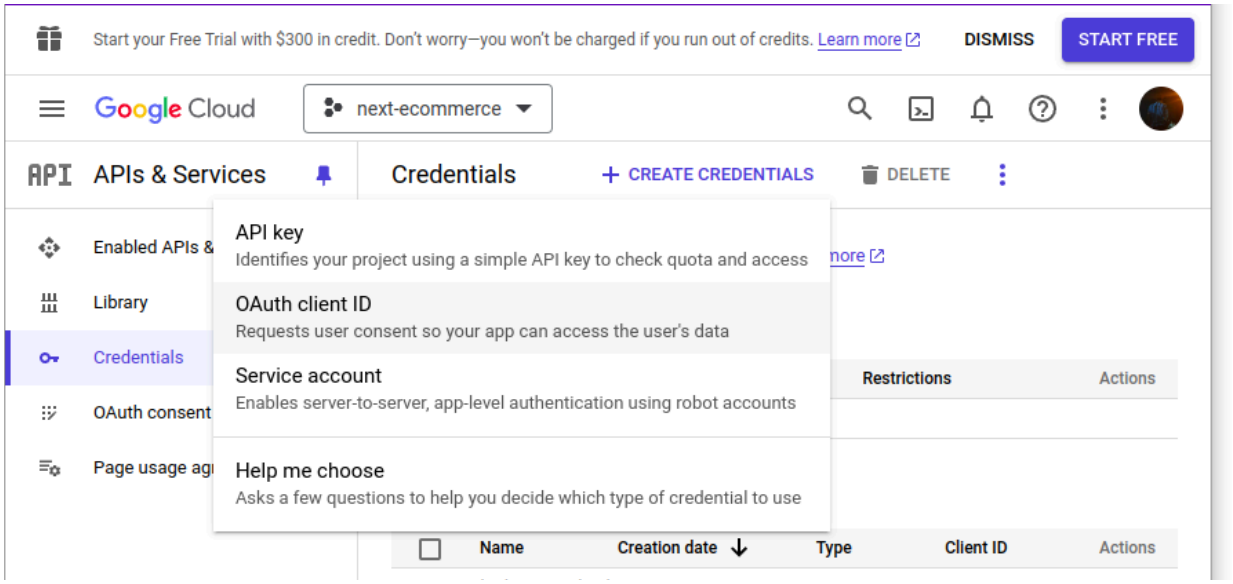
Select User type to external and then create



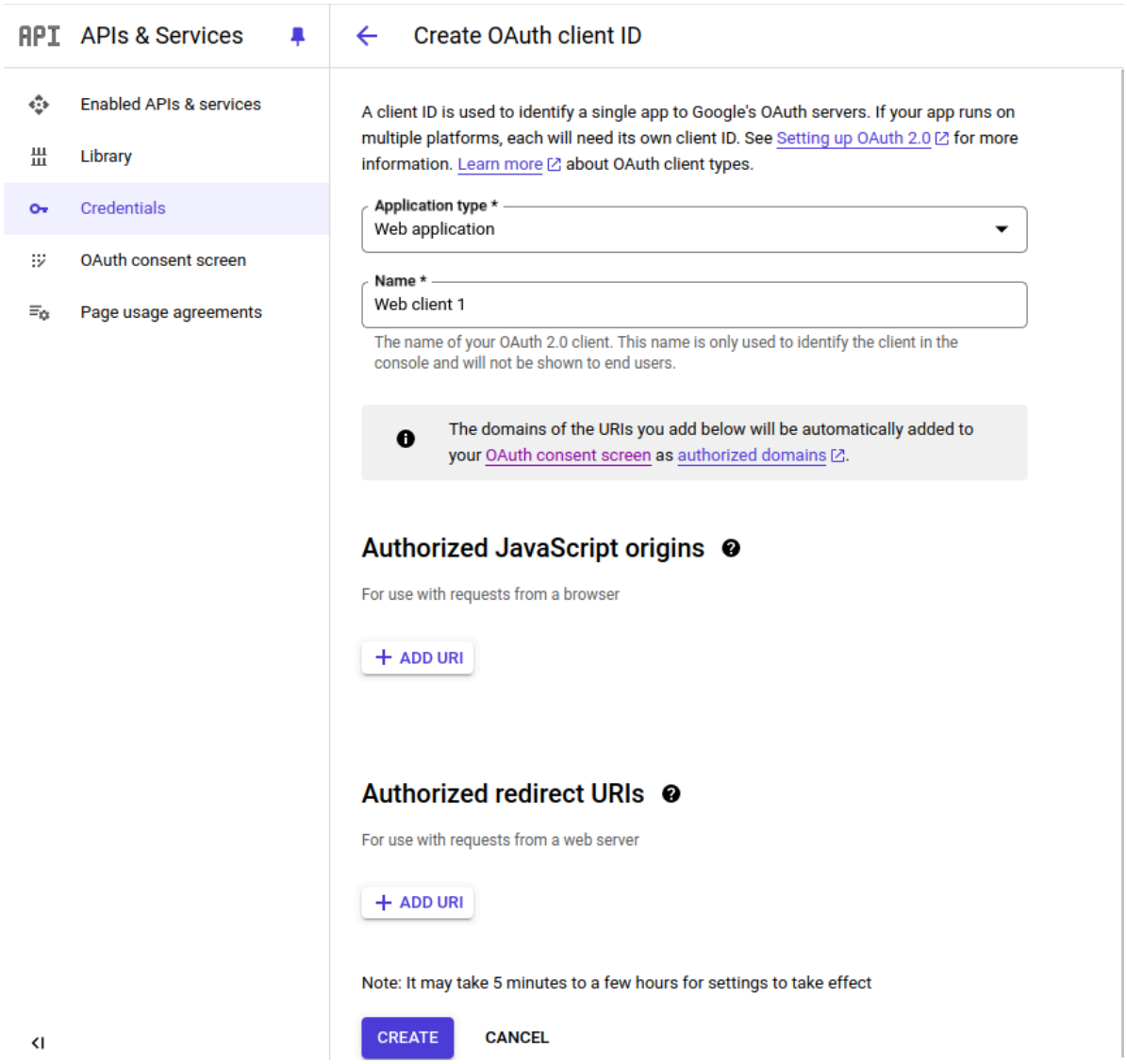
Complete the registration



Now go back to the dashboard and choose Credentials and click on the Create Credentials button and choose OAuth client id



Now complete the

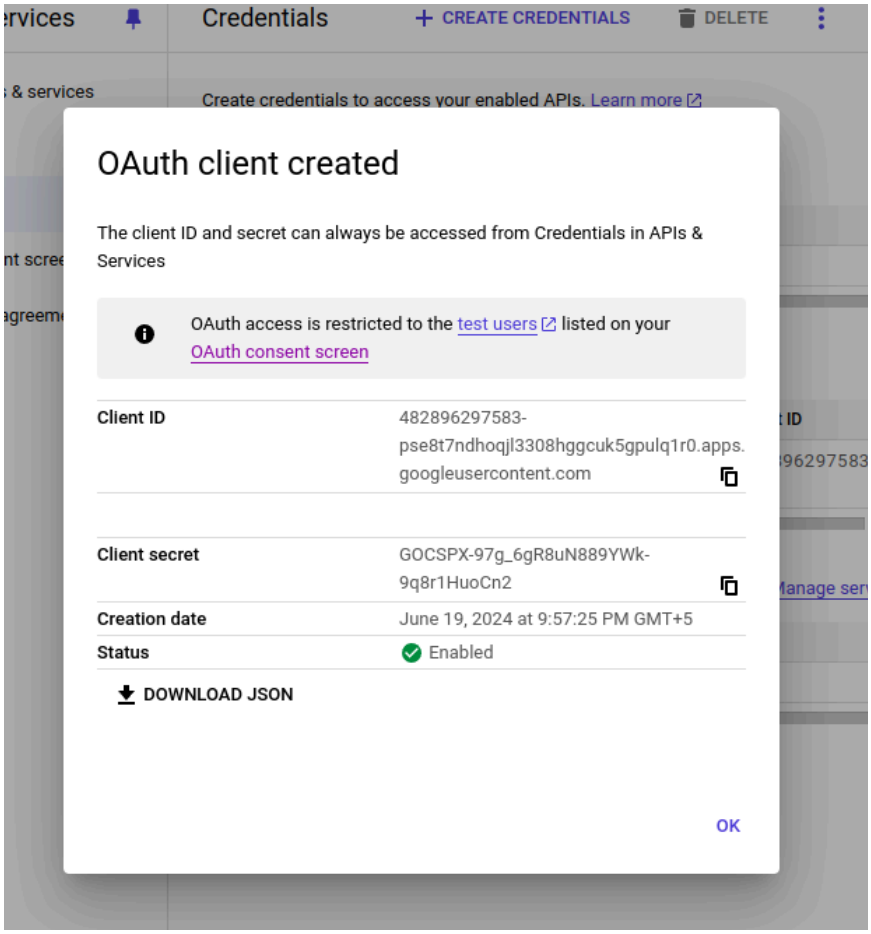


Now we have to add Authorized redirect URIs, to see this lets go through the NextAuth Documentation

[Google | NextAuth.js]

For Development

`http://localhost:3000/api/auth/callback/google` and create



Client id:

133254144567-3o27n7iscd4c1n4lbr2fih3d1ab2vrpp.apps.googleusercontent.com

Client secret:

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

Add Client(App)

pages/app.jsx

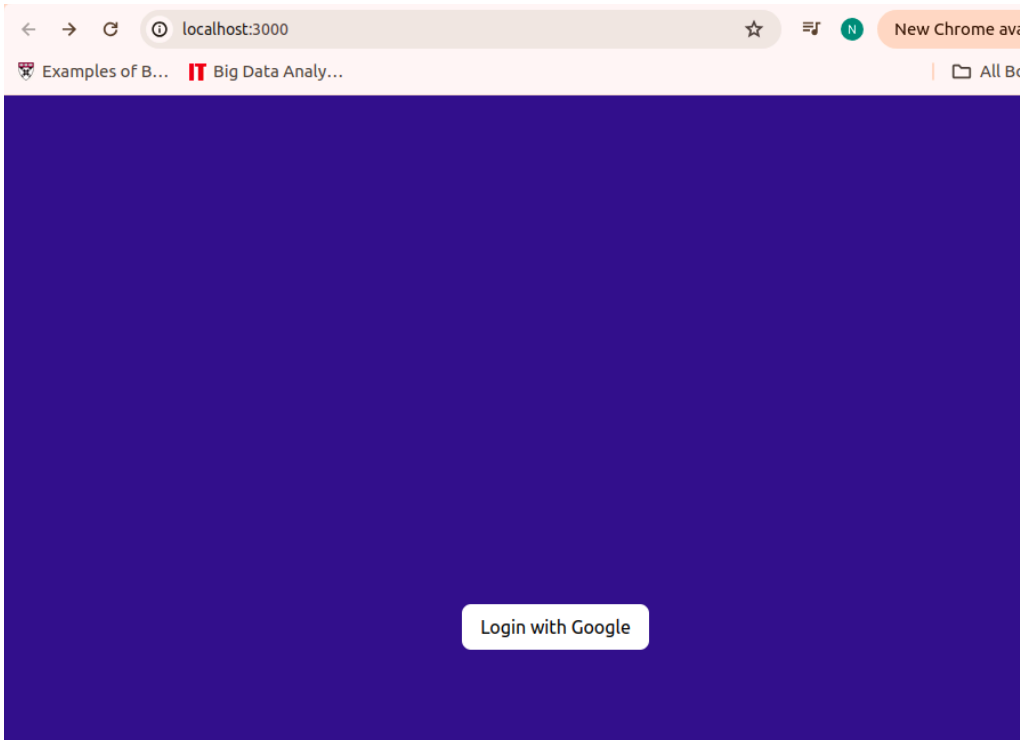
```
import "@/styles/globals.css";
import { SessionProvider } from "next-auth/react"

export default function App({
```

```
 Component, pageProps: { session, ...pageProps }
}) {
 return (
   <SessionProvider session={session}>
     <Component {...pageProps}/>
   </SessionProvider>
 )
}
```

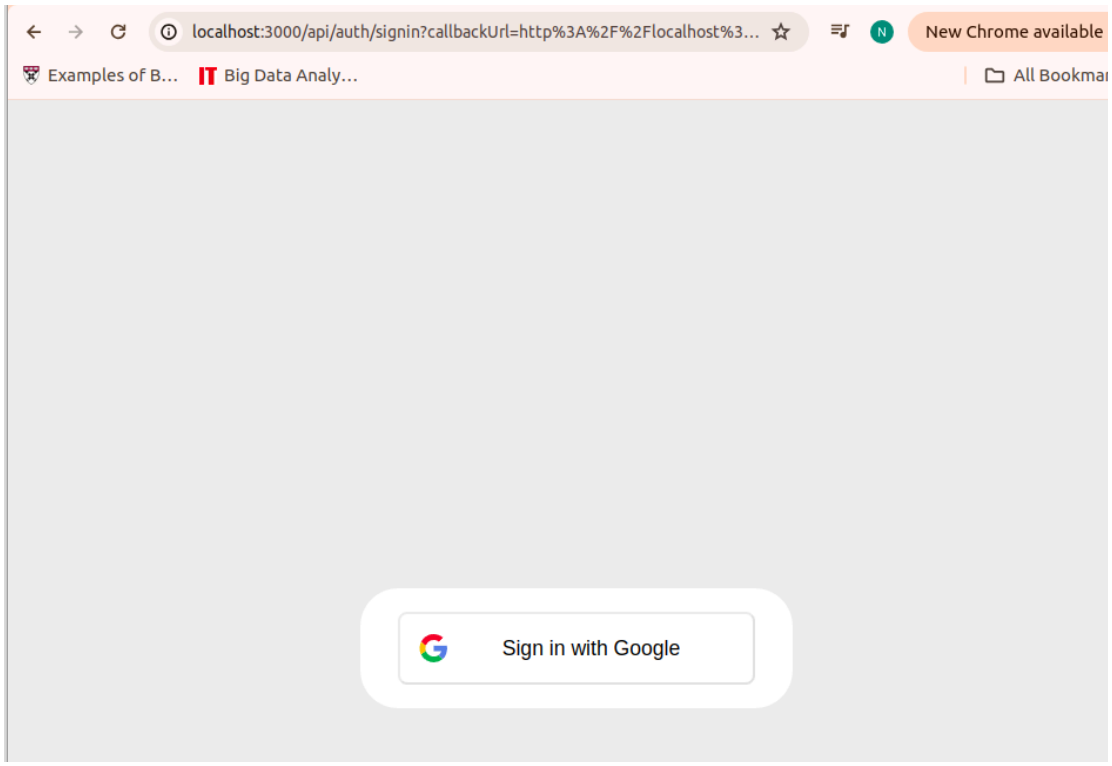
Let go to Client page

pages/index.js

```
import { useSession, signIn, signOut } from "next-auth/react";

export default function Home() {
 const { data: session } = useSession();
 //rename from session to data
 // if we have session user is logged in else not
 if (!session) {
   return (
     <div className="bg-blue-900 w-screen h-screen flex items-center">
       <div className="text-center w-full">
         <button onClick={()=> signIn()} className="bg-white p-2
rounded-lg px-4 ">
           Login with Google
         </button>
       </div>
     </div>
   );
 }

 return (
   <div>Logged In {session.user.email}</div>
 );
}
```



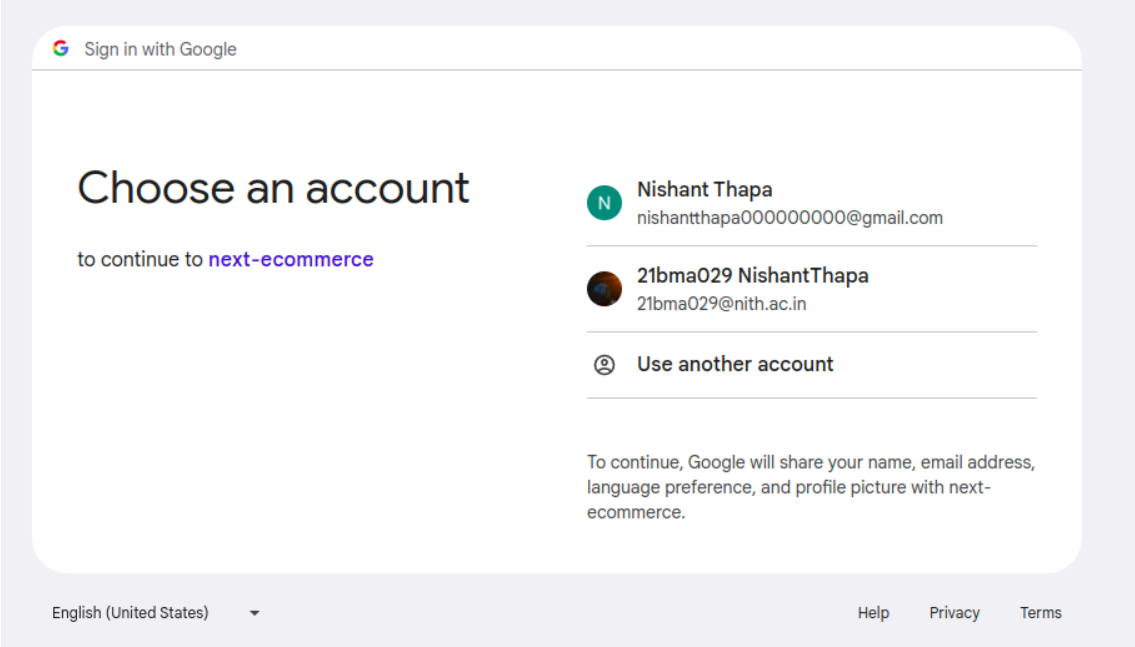
On clicking the button we are redirected to different URL but we want that google button in this page only



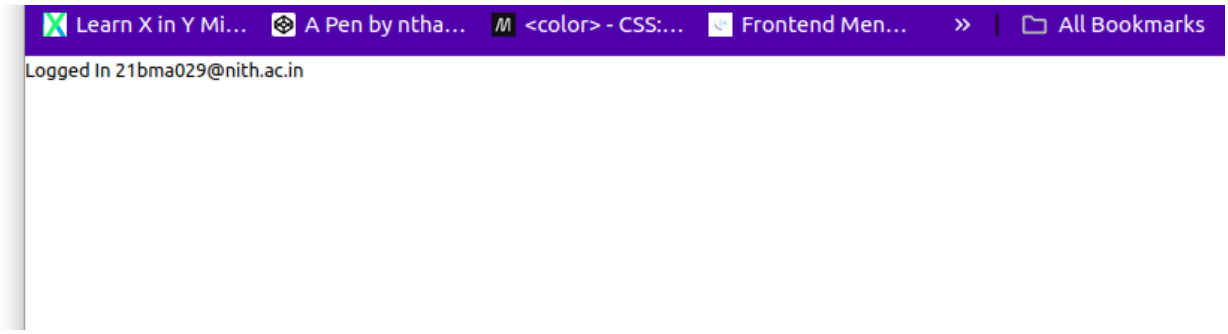
To fix this we need to add the provider id as param

```
        <button onClick={()=> signIn('google')} className="bg-white p-2
rounded-lg px-4 ">
          Login with Google
        </button>
```

Now we will be redirected to the Login page



The Name of the application Visible is what we choose on the Google cloud while making a new project, after login in



Currently we don’t have any database, whenever any user login it will go to the database we want information about the users.

Go to Adapter section in NextAuth and choose mongoDB

npm install @auth/mongodb-adapter mongodb

Add **lib/mongodb.js**

```
// This approach is taken from
https://github.com/vercel/next.js/tree/canary/examples/with-mongodb
import { MongoClient, ServerApiVersion } from "mongodb"
if (!process.env.MONGODB_URI) {
 throw new Error('Invalid/Missing environment variable: "MONGODB_URI"')
}
const uri = process.env.MONGODB_URI
const options = {
 serverApi: {
   version: ServerApiVersion.v1,
   strict: true,
   deprecationErrors: true,
 },
}
let client
let clientPromise;
if (process.env.NODE_ENV === "development") {
 // In development mode, use a global variable so that the value
 // is preserved across module reloads caused by HMR (Hot Module
Replacement).
 let globalWithMongo = global & {
   _mongoClientPromise
 }
 if (!globalWithMongo._mongoClientPromise) {
   client = new MongoClient(uri, options)
   globalWithMongo._mongoClientPromise = client.connect()
 }
 clientPromise = globalWithMongo._mongoClientPromise
} else {
 // In production mode, it's best to not use a global variable.
 client = new MongoClient(uri, options)
 clientPromise = client.connect()
```

```
}
// Export a module-scoped MongoClient promise. By doing this in a
// separate module, the client can be shared across functions.
export default clientPromise
```

This code sets up a MongoDB client connection for a Next.js application, handling both development and production environments efficiently.

In development, it uses a global variable to store the MongoClient promise. This approach preserves the MongoClient instance across module reloads due to Hot Module Replacement (HMR), avoiding multiple connections.

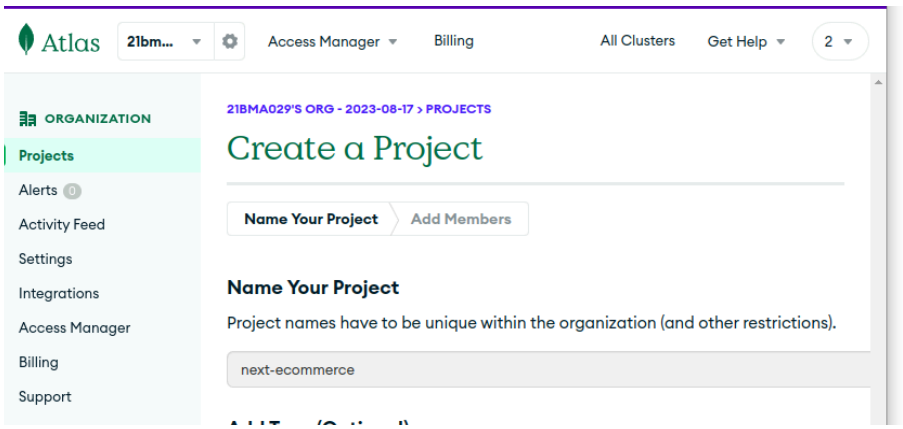
In production, it directly creates a new MongoClient instance and connects it without using a global variable. This is more suitable for a production environment to avoid potential issues with global state.

[..auth].js

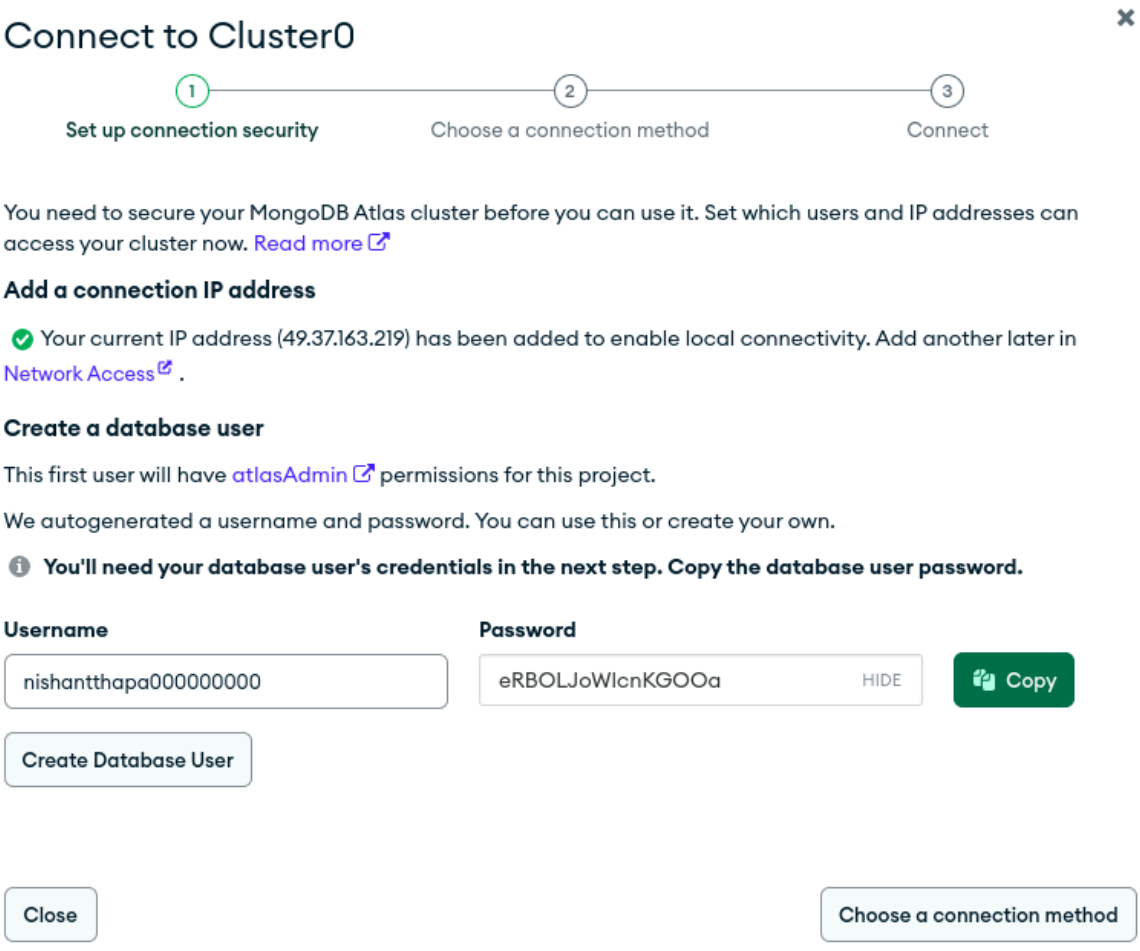
```
import clientPromise from '@/lib/mongodb'
import { MongoDBAdapter } from '@auth/mongodb-adapter'
import NextAuth from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'

export default NextAuth({
 providers: [
   // OAuth authentication providers...
   GoogleProvider({
     clientId: process.env.GOOGLE_ID,
     // now we have to provide GOOGLE_ID and GOOGLE_SECRET by .env file
     clientSecret: process.env.GOOGLE_SECRET
   }),
 ],
 adapter:MongoDBAdapter(clientPromise)
 // it will reuse an active connetion to DB if there is not such then it
will create one
})
```

Now provide MongoDB URI in the Environment file

Create a new Project



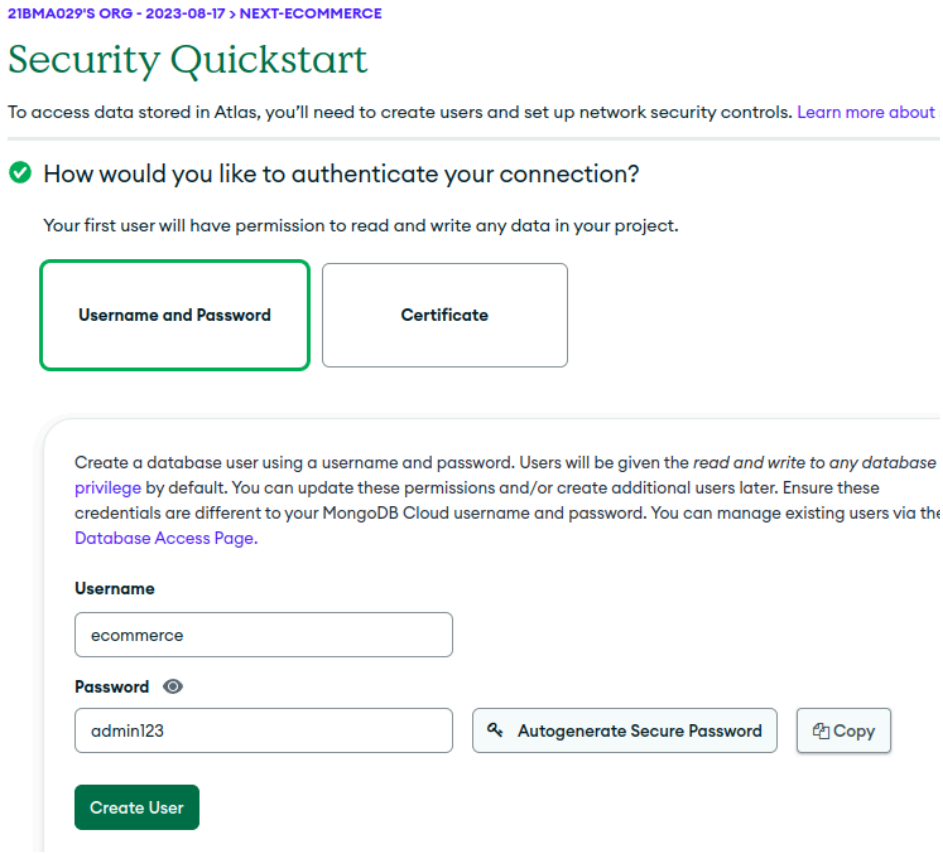
Create a new Database/Cluster



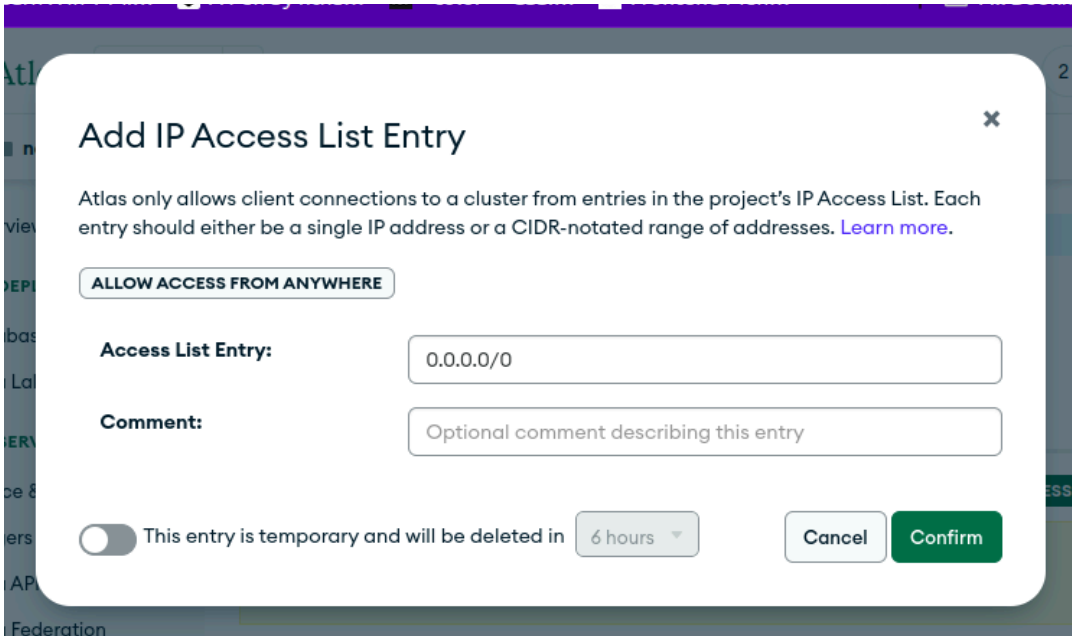
Password: eRBOLJoWIcnKGOOa

Username: ecommerce

Click on quick start on security



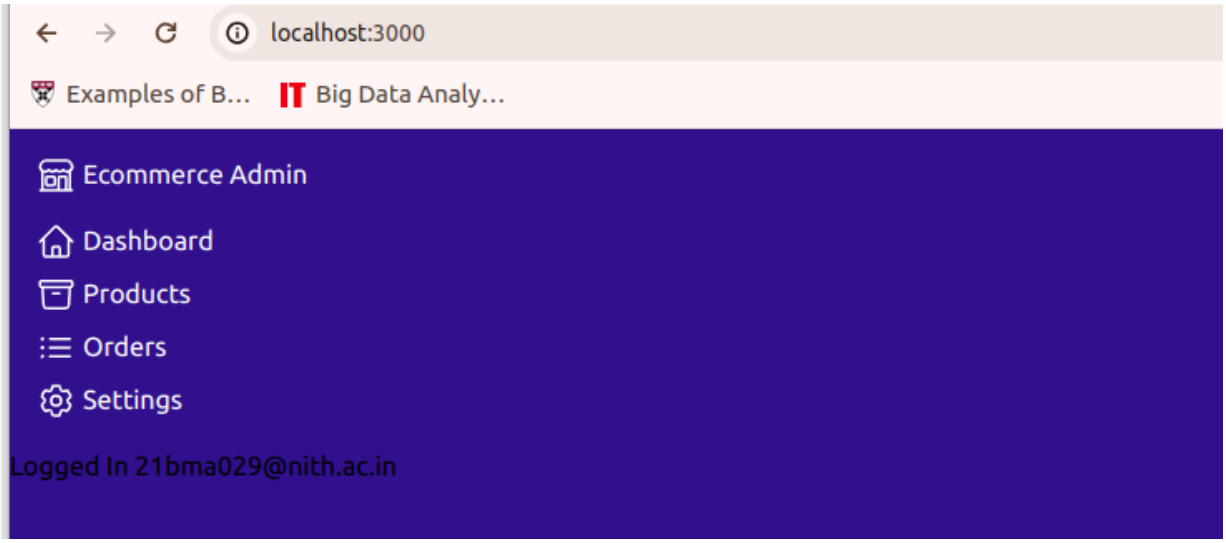
After adding users add the current Ip address, When you push for deployment make sure to allow all the IP address



Allow access from Everywhere

Now try to login

All the icons are imported as jsx from **heroicons**



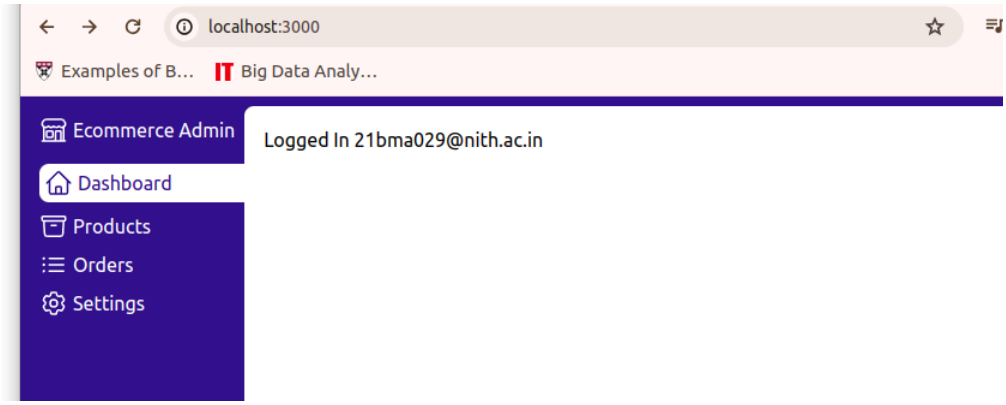
Nav.js

```jsx
import Link from "next/link";

export default function Nav() {
 const inactiveLink = 'flex gap-1 p-1';
 const activeLink = inactiveLink +' bg-white text-blue-900 rounded-l-lg';

 return (
   <aside className="text-white p-4 pr-0">
     <Link href={"/"} className="flex gap-1 mb-4 mr-2">
       {/* From heroicons */}
       <svg
         xmlns="http://www.w3.org/2000/svg"
         fill="none"
         viewBox="0 0 24 24"
         strokeWidth={1.5}
         stroke="currentColor"
         className="size-6"
       >
         <path
           strokeLinecap="round"
           strokeLinejoin="round"
           d="M13.5 21v-7.5a.75.75 0 0 1 .75-.75h3a.75.75 0 0 1
.75.75V21m-4.5 0H2.36m11.14 0H18m0 0h3.64m-1.39 0V9.349M3.75 21V9.349m0
0a3.001 3.001 0 0 0 3.75-.615A2.993 2.993 0 0 0 9.75 9.75c.896 0 1.7-.393
2.25-1.016a2.993 2.993 0 0 0 2.25 1.016c.896 0 1.7-.393 2.25-1.015a3.001
3.001 0 0 0 3.75.614m-16.5 0a3.004 3.004 0 0 1-.621-4.72l1.189-1.19A1.5
1.5 0 0 1 5.378 3h13.243a1.5 1.5 0 0 1 1.06.44l1.19 1.189a3 3 0 0 1-.621
4.72M6.75 18h3.75a.75.75 0 0 0 .75-.75V13.5a.75.75 0 0
0-.75-.75H6.75a.75.75 0 0 0-.75.75v3.75c0 .414.336.75.75.75Z"
         />
       </svg>
       <span className="">Ecommerce Admin</span>
     </Link>

     <nav className="flex flex-col gap-2">
       <Link href={"/"} className={activeLink}>
         <svg
           xmlns="http://www.w3.org/2000/svg"
           fill="none"
           viewBox="0 0 24 24"
```

```
            strokeWidth={1.5}
            stroke="currentColor"
            className="size-6"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="m2.25 12 8.954-8.955c.44-.439 1.152-.439 1.591 0L21.75
12M4.5 9.75v10.125c0 .621.504 1.125 1.125
1.125H9.75v-4.875c0-.621.504-1.125 1.125-1.125h2.25c.621 0 1.125.504 1.125
1.125V21h4.125c.621 0 1.125-.504 1.125-1.125V9.75M8.25 21h8.25"
            />
          </svg>
          Dashboard
        </Link>
        <Link href={"/products"} className="flex gap-1 ">
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            strokeWidth={1.5}
            stroke="currentColor"
            className="size-6"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="m20.25 7.5-.625 10.632a2.25 2.25 0 0 1-2.247
2.118H6.622a2.25 2.25 0 0 1-2.247-2.118L3.75 7.5M10 11.25h4M3.375
7.5h17.25c.621 0 1.125-.504
1.125-1.125v-1.5c0-.621-.504-1.125-1.125-1.125H3.375c-.621
0-1.125.504-1.125 1.125v1.5c0 .621.504 1.125 1.125 1.125Z"
            />
          </svg>
          Products
        </Link>
        <Link href={"/orders"} className="flex gap-1 ">
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
```

```
          viewBox="0 0 24 24"
          strokeWidth={1.5}
          stroke="currentColor"
          className="size-6"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M8.25 6.75h12M8.25 12h12m-12 5.25h12M3.75
6.75h.007v.008H3.75V6.75Zm.375 0a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75
0ZM3.75 12h.007v.008H3.75V12Zm.375 0a.375.375 0 1 1-.75 0 .375.375 0 0 1
.75 0Zm-.375 5.25h.007v.008H3.75v-.008Zm.375 0a.375.375 0 1 1-.75 0
.375.375 0 0 1 .75 0Z"
          />
        </svg>
        Orders
      </Link>
      <Link href={"/settings"} className="flex gap-1 ">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          strokeWidth={1.5}
          stroke="currentColor"
          className="size-6"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M9.594 3.94c.09-.542.56-.94 1.11-.94h2.593c.55 0 1.02.398
1.11.941.213
1.281c.063.374.313.686.645.87.074.04.147.083.22.127.325.196.72.257
1.075.124l1.217-.456a1.125 1.125 0 0 1 1.37.49l1.296 2.247a1.125 1.125 0 0
1-.26 1.431l-1.003.827c-.293.241-.438.613-.43.992a7.723 7.723 0 0 1 0
.255c-.008.378.137.75.43.991l1.004.827c.424.35.534.955.26 1.43l-1.298
2.247a1.125 1.125 0 0
1-1.369.491l-1.217-.456c-.355-.133-.75-.072-1.076.124a6.47 6.47 0 0
1-.22.128c-.331.183-.581.495-.644.869l-.213
1.281c-.09.543-.56.94-1.11.94h-2.594c-.55
0-1.019-.398-1.11-.94l-.213-1.281c-.062-.374-.312-.686-.644-.87a6.52 6.52
```

```
0 0 1-.22-.127c-.325-.196-.72-.257-1.076-.124l-1.217.456a1.125 1.125 0 0
1-1.369-.491-1.297-2.247a1.125 1.125 0 0 1
.26-1.431l1.004-.827c.292-.24.437-.613.43-.991a6.932 6.932 0 0 1
0-.255c.007-.38-.138-.751-.43-.992l-1.004-.827a1.125 1.125 0 0
1-.26-1.431l1.297-2.247a1.125 1.125 0 0 1
1.37-.491l1.216.456c.356.133.751.072
1.076-.124.072-.044.146-.086.22-.128.332-.183.582-.495.644-.869l.214-1.28Z
"
          />
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M15 12a3 3 0 1 1-6 0 3 3 0 0 1 6 0Z"
          />
        </svg>
        Settings
      </Link>
    </nav>
  </aside>
 );
}
```



components/Layout.js

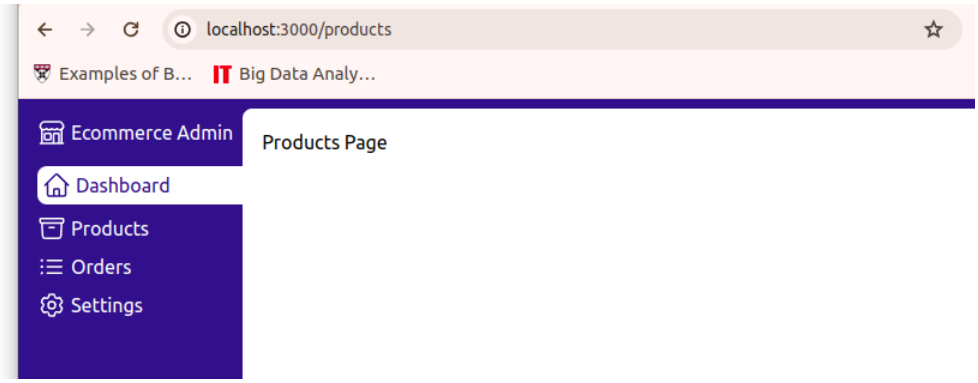
```
import Nav from "@/components/Nav";
import { useSession, signIn, signOut } from "next-auth/react";
```

```
export default function Layout({children}) {
 const { data: session } = useSession();

 if (!session) {
   return (
     <div className="bg-blue-900 w-screen h-screen flex items-center">
       <div className="text-center w-full">
         <button
           onClick={() => signIn("google")}
           className="bg-white p-2 rounded-lg px-4"
         >
           Login with google
         </button>
       </div>
     </div>
   );
 }

 return (
   <div className="bg-blue-900 min-h-screen flex">
     <Nav/>
     <div className="bg-white flex-grow mt-2 mr-2 rounded-lg p-4 mb-2">
       {children}
     </div>
   </div >
 );
}
```

Now we can use the same page for the Products also

```
import Layout from "@/components/Layout";

export default function Products(){
   return (
       <Layout>
           Products Page
       </Layout>
   )
}
```

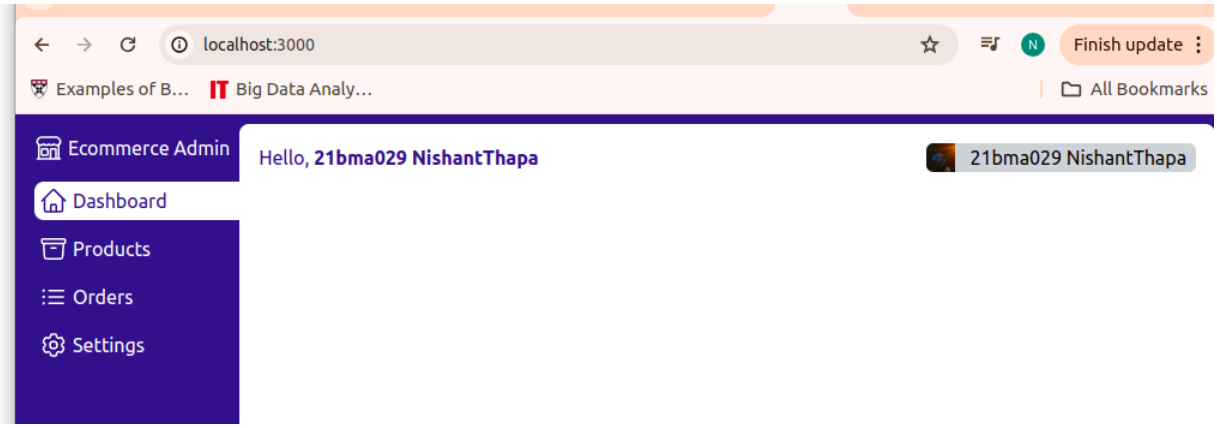
But we can see that even on clicking on the products page still the Dashboard page is showing active



See the changes made on the Nav.js in Github Repository

pages/index.js

```
import Layout from "@/components/Layout";
import { useSession } from "next-auth/react";

export default function Home() {
 const { data: session } = useSession();
 return (
   <Layout>
     <div className="text-blue-900 flex justify-between">
       <h2>Hello, <b>{session?.user?.name}</b></h2>
       <div className="flex bg-gray-300 text-black gap-1 rounded-md
overflow-hidden">
         <img src={session?.user?.image} alt="" className="w-6 h-6 "/>
         <span className="px-2">
         {session?.user?.name}
         </span>

       </div>
     </div>
   </Layout>
 );
}
```



Repository at this moment :

https://github.com/nthapa000/ecommerce-admin/commits/main/

Now lets work on Products page

```
import Layout from "@/components/Layout";
import Link from "next/link";

export default function Products(){
   return (
       <Layout>
           <Link className="bg-blue-900 text-white py-1 px-2 rounded-md"
href={'/products/new'}>
               Add a new product
           </Link>
       </Layout>
   )
}
```

products/new.jsx

```
import Layout from "@/components/Layout";

export default function NewProduct(){
   return (
       <Layout>
           <h1 >New Products</h1>
           <label>Product Name</label>
```

```
            <input type="text" placeholder="product name" />
            <label>Description</label>
            <textarea placeholder="description"></textarea>
            <label>Price</label>
            <input type="number" placeholder="price"/>
            <button className="btn-primary">
                Save
            </button>
        </Layout>
    )
}
```

Now lets add global styling for all our inputs

Global.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

input, textarea {
    @apply border-2 border-gray-300 rounded-md px-1 w-full;
    @apply mb-2;
}
input:focus, textarea:focus{
    @apply border-blue-900
}
h1{
    @apply text-blue-900 mb-2 text-xl;
}
label{
    @apply text-blue-900;
}
.btn-primary{
    @apply bg-blue-900 text-white px-4 rounded-md py-1;
}
```

npm install axios

```jsx
import Layout from "@/components/Layout";
import { useState } from "react";
import axios from "axios";

export default function NewProduct() {
 const [title, setTitle] = useState("");
 const [description, setDescription] = useState("");
 const [price, setPrice] = useState("");
 async function createProduct() {
   // We need to send request to api will do through axios
   const data = {title,description,price}
   await axios.post('/api/products',data);
 }

 return (
   <Layout>
     <form onSubmit={createProduct}>
       <h1>New Products</h1>
       <label>Product Name</label>
       <input
         type="text"
         placeholder="product name"
         value={title}
         onChange={(ev) => setTitle(ev.target.value)}
       />
       <label>Description</label>
       <textarea
         placeholder="description"
         value={description}
         onChange={(ev) => setDescription(ev.target.value)}
       ></textarea>
       <label>Price</label>
       <input
         type="number"
         placeholder="price"
         value={price}
         onChange={(ev) => setPrice(ev.target.value)}
       />
       <button type="submit" className="btn-primary">
         Save
```

```
        </button>
      </form>
    </Layout>
  );
}
```

Now we have to make this route for api

api/products.js

```
export default function handle(req,res){
    res.json(req.method)
}
```

And on checking in network tab its working

Now lets add mongoose since we will be adding new products we need a database connection

npm install mongoose

Now we need to create Model for Products

model/product.js

```
const { Schema } = require("mongoose");

const ProductSchema = new Schema({
    title: {type:String, required:true},
    description: String,
    price: {type:Number, required:true},
});

export const Product = model('product',ProductSchema)
```

Now lets work on our api/products.js

```
import { Product } from "@/model/Product";
```

```
import mongoose from "mongoose";

export default async function handle(req,res){
   const {method} = req;
   // Now to put new products into database we need database connection
hence we will use mongoose

   if(method==='POST'){
       const {title,description,price}= req.body;
       const productDoc = await Product.create({
           title,description,price
       })
       res.json(productDoc)
   }
}
```

Now in this we need to connect the mongoose to the database

lib/mongoose.js

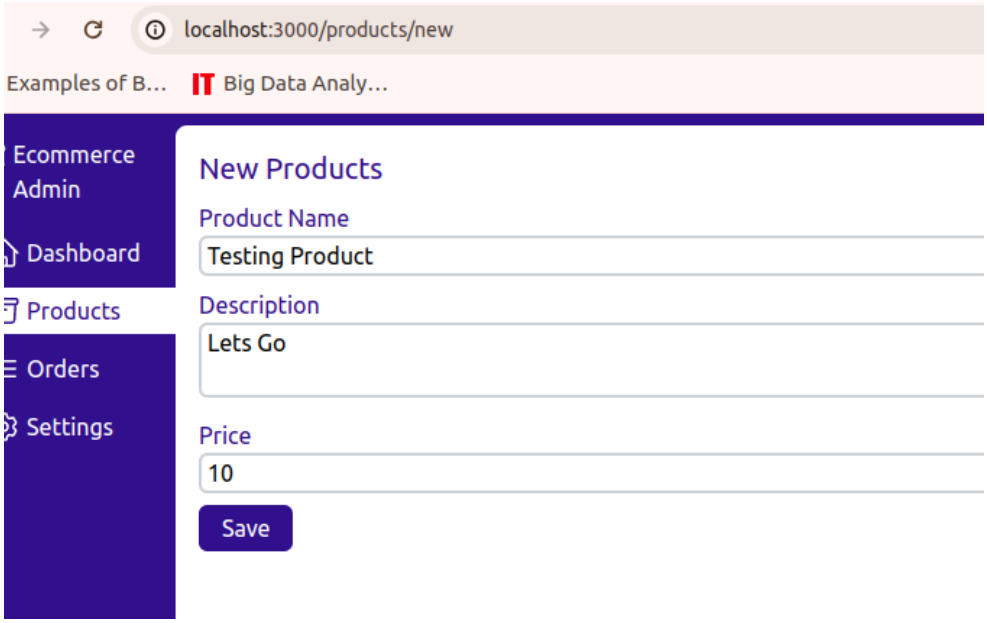
```
import mongoose from "mongoose";

export function mongooseConnect(){
   const uri = process.env.MONGODB_URI;
   if(mongoose.connection.readyState === 1){
       return mongoose.connection.asPromise();
   } else{
       const uri = process.env.MONGODB_URI;
       return mongoose.connect(uri);
   }
}
```

Now add this mongooseConnect() in api/products

```
import { mongooseConnect } from "@/lib/mongoose";
import { Product } from "@/model/Product";

export default async function handle(req,res){
   const {method} = req;
```

```
    // Now to put new products into database we need database connection
hence we will use mongoose
    await mongooseConnect();
    if(method==='POST'){
        const {title,description,price}= req.body;
        const productDoc = await Product.create({
            title,description,price
        })
        res.json(productDoc)
    }
}
```

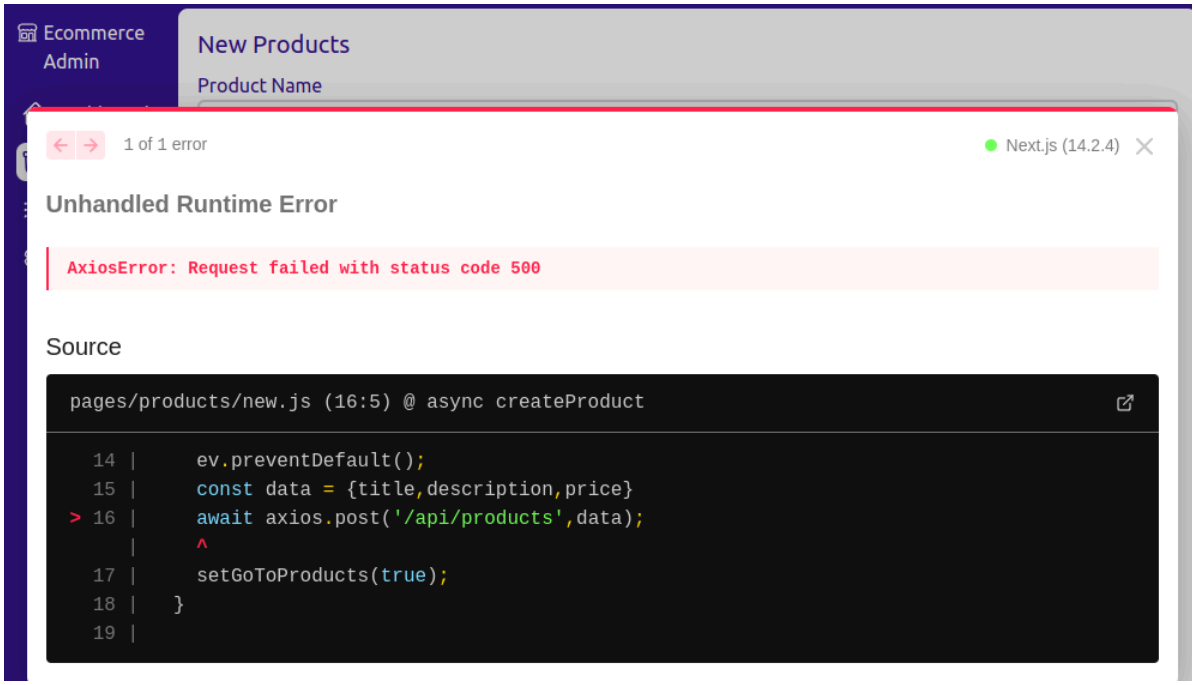
Now lets test this



We can also verify it by seeing it on mongoDB

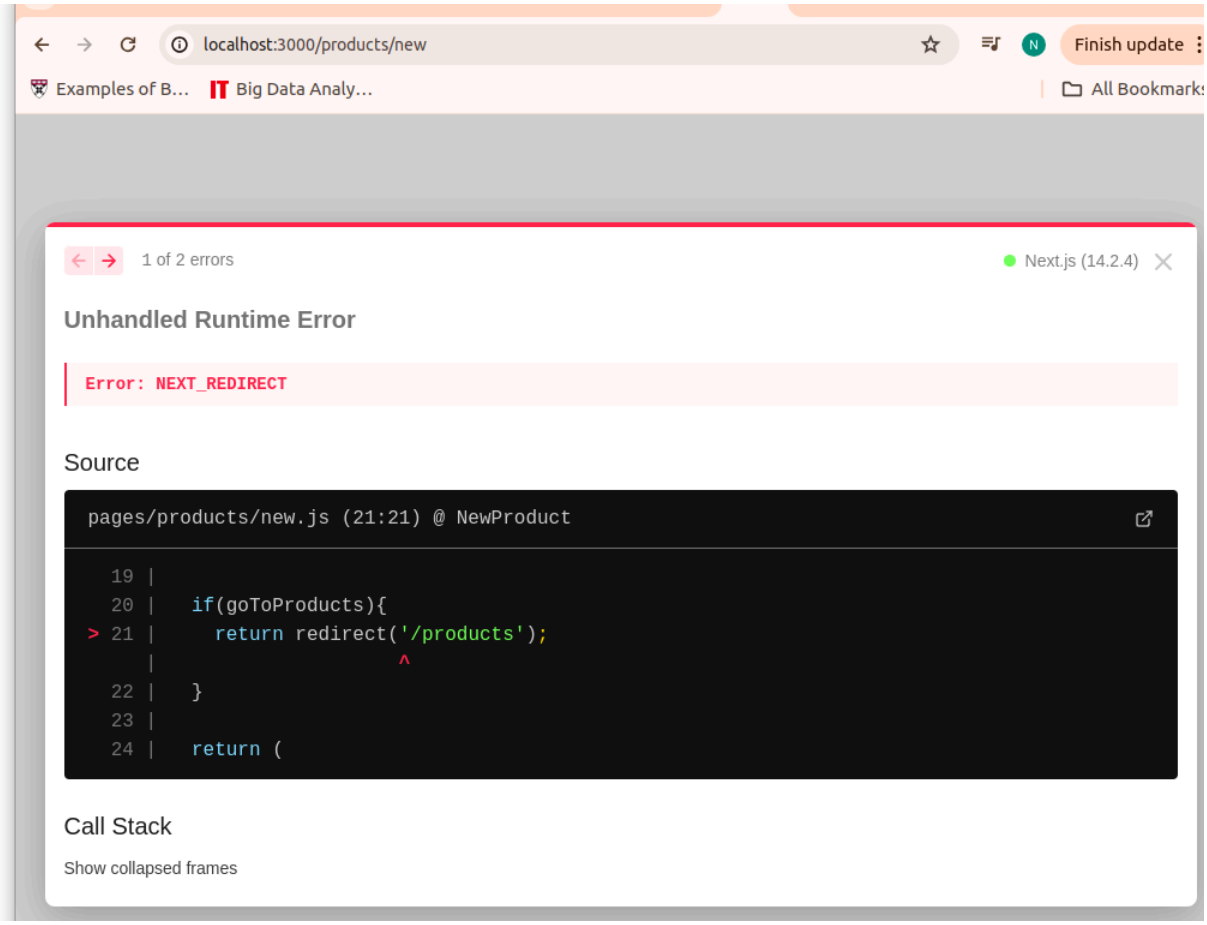
Now in products page we should be able to display this product and in Products/new.js we should redirect once we submitted the product

Now if we again try to create a new product then it show error since we are again trying to create the model Product which is already made



```
export const Product =models.Product || model('Product',ProductSchema);
```

Now we are getting this error

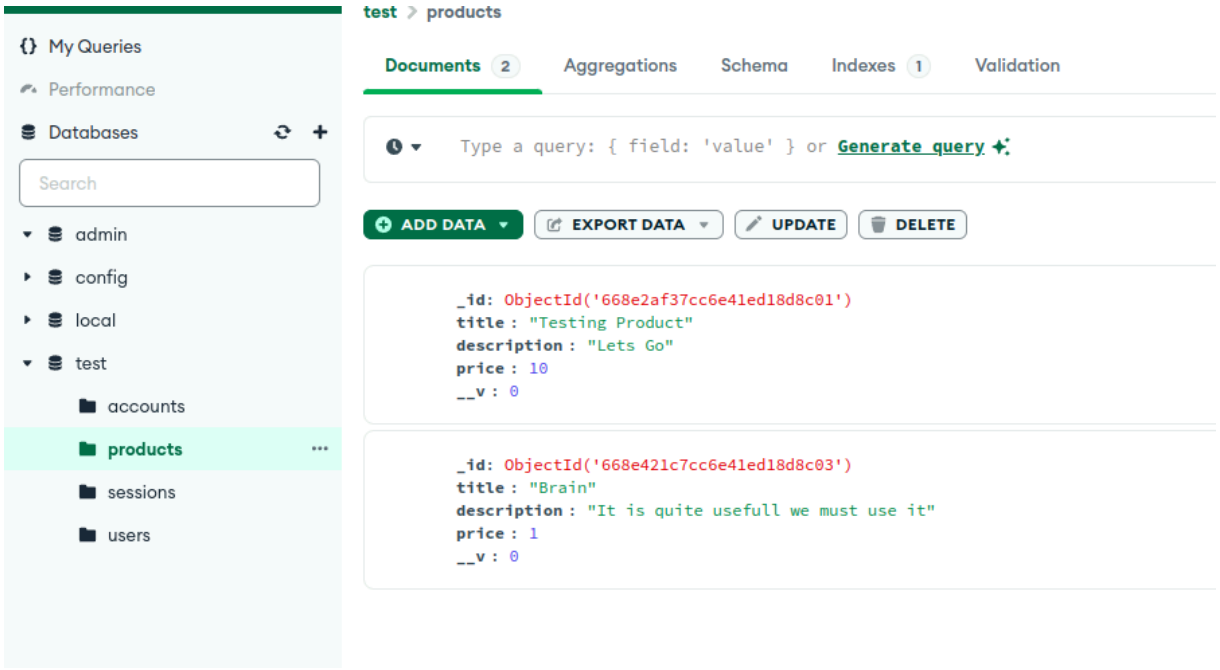


pages/products/new.js

```
if(goToProducts){
  router.push('/products')
}
```

Now we are redirected to the product page.

Now we want to list all the products from the database



Now lets work on

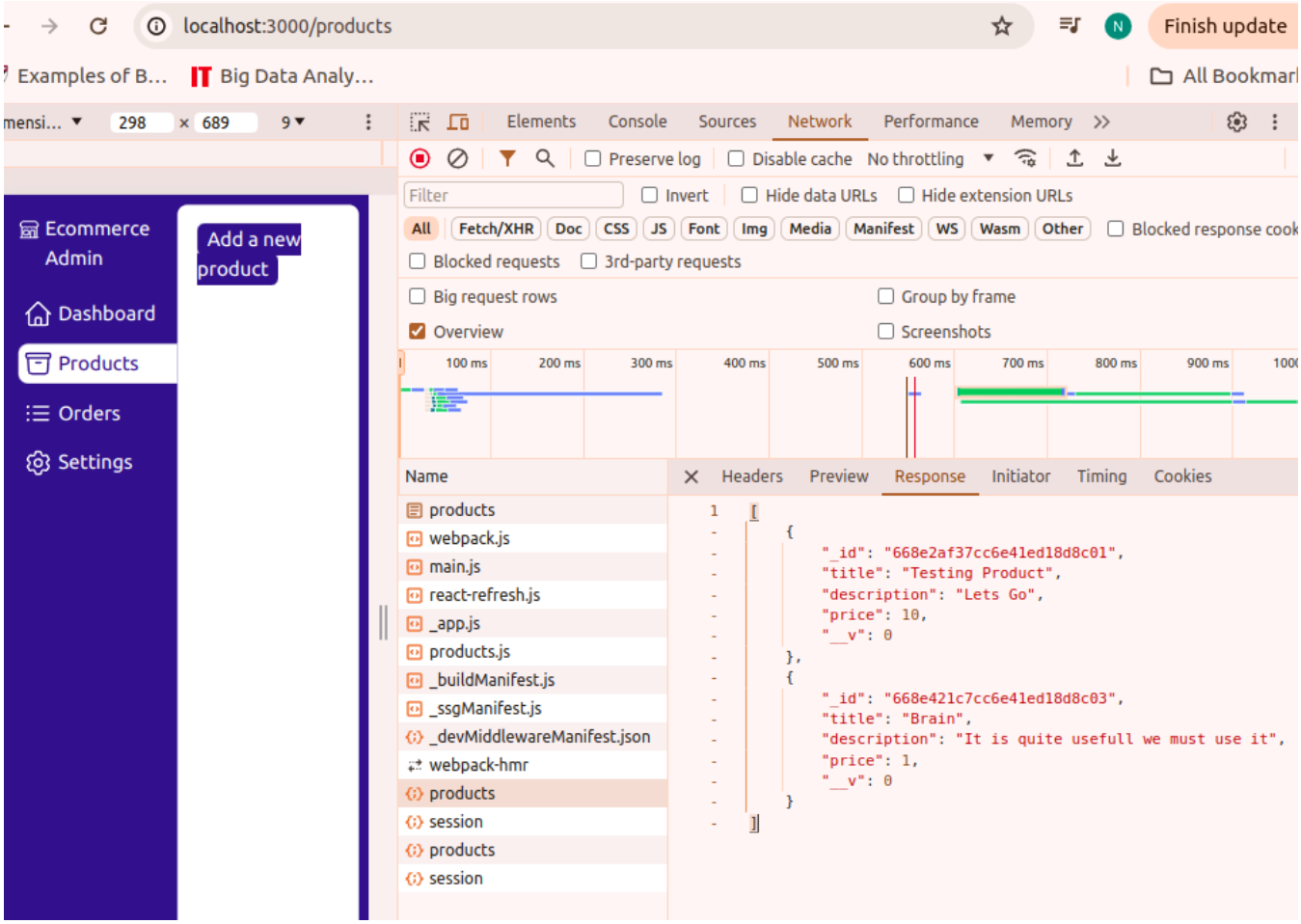
pages/product.js

```
import Layout from "@/components/Layout";
import axios from "axios";
import Link from "next/link";
import { useEffect } from "react";

export default function Products(){
   useEffect(()=>{
       axios.get('/api/products').then(response =>{
           console.log(response.data);
       })
   },[])
   return (
       <Layout>
           <Link className="bg-blue-900 text-white py-1 px-2 rounded-md"
href={'/products/new'}>
               Add a new product
           </Link>
       </Layout>
   )
}
```

Now lets go back to api/product.js make a get request

```
if(method === 'GET'){
      res.json(await Product.find())
  }
```

Now we can see that we are getting the request



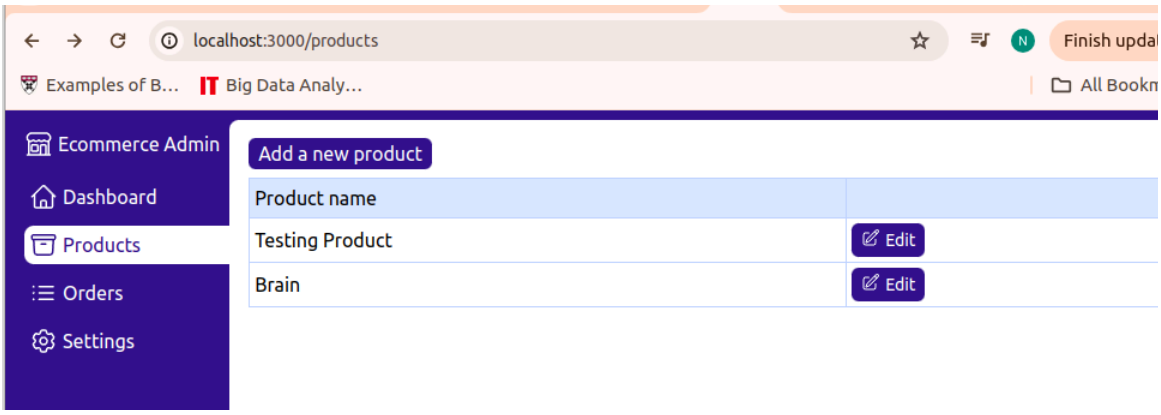
Now lets make a table to display these products

pages/product.js

```
import Layout from "@/components/Layout";
import axios from "axios";
import Link from "next/link";
import { useEffect, useState } from "react";

export default function Products() {
 const [products, setProducts] = useState([]);
```

```
useEffect(() => {
  axios.get("/api/products").then((response) => {
    setProducts(response.data);
  });
}, []);
return (
  <Layout>
    <Link
      className="bg-blue-900 text-white py-1 px-2 rounded-md"
      href={"/products/new"}
    >
      Add a new product
    </Link>
    <table className="basic mt-2">
      <thead>
        <tr>
          <td>Product name</td>
          <td></td>
        </tr>
      </thead>
      <tbody>
        {products.map((product) => (
          <tr>
            <td>{product.title}</td>
            <td>
              <Link href={"/products/edit/" + product._id}>
                <svg
                  xmlns="http://www.w3.org/2000/svg"
                  fill="none"
                  viewBox="0 0 24 24"
                  strokeWidth={1.5}
                  stroke="currentColor"
                  className="w-4 h-4"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    d="m16.862 4.487 1.687-1.688a1.875 1.875 0 1 1 2.652
2.652L10.582 16.07a4.5 4.5 0 0 1-1.897 1.13L6 18l.8-2.685a4.5 4.5 0 0 1
```

```
1.13-1.897l8.932-8.931Zm0 0L19.5 7.125M18 14v4.75A2.25 2.25 0 0 1 15.75
21H5.25A2.25 2.25 0 0 1 3 18.75V8.25A2.25 2.25 0 0 1 5.25 6H10"
                    />
                  </svg>
                  Edit
                </Link>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </Layout>
  );
}
```

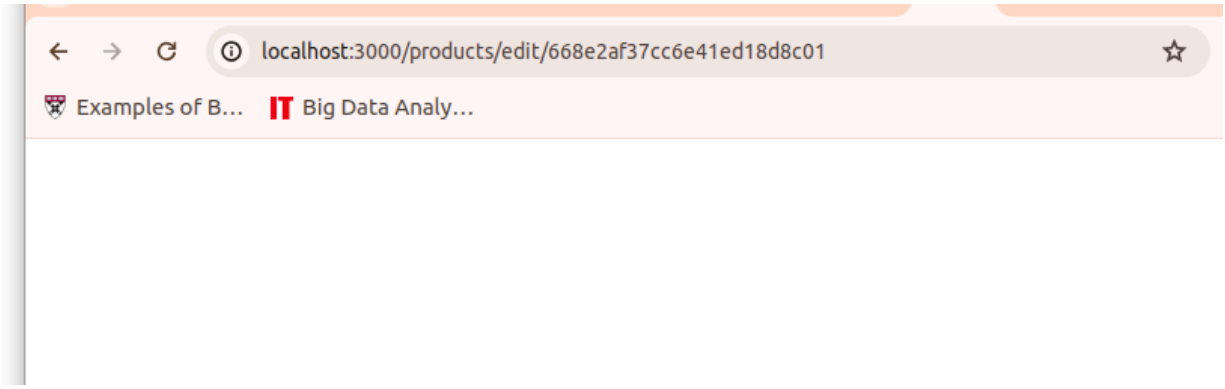


Global.css

```
table.basic{
    @apply w-full
}
table.basic thead tr td {
    @apply bg-blue-100
}
table.basic tr td{
    @apply border p-1 border-blue-200;
}
table.basic a{
    @apply bg-blue-900 text-white text-sm py-1 rounded-md px-2 inline-flex
gap-1
```

```
}
```

On clicking on Edit



Lets make this route pages/products/edit/[...id].js

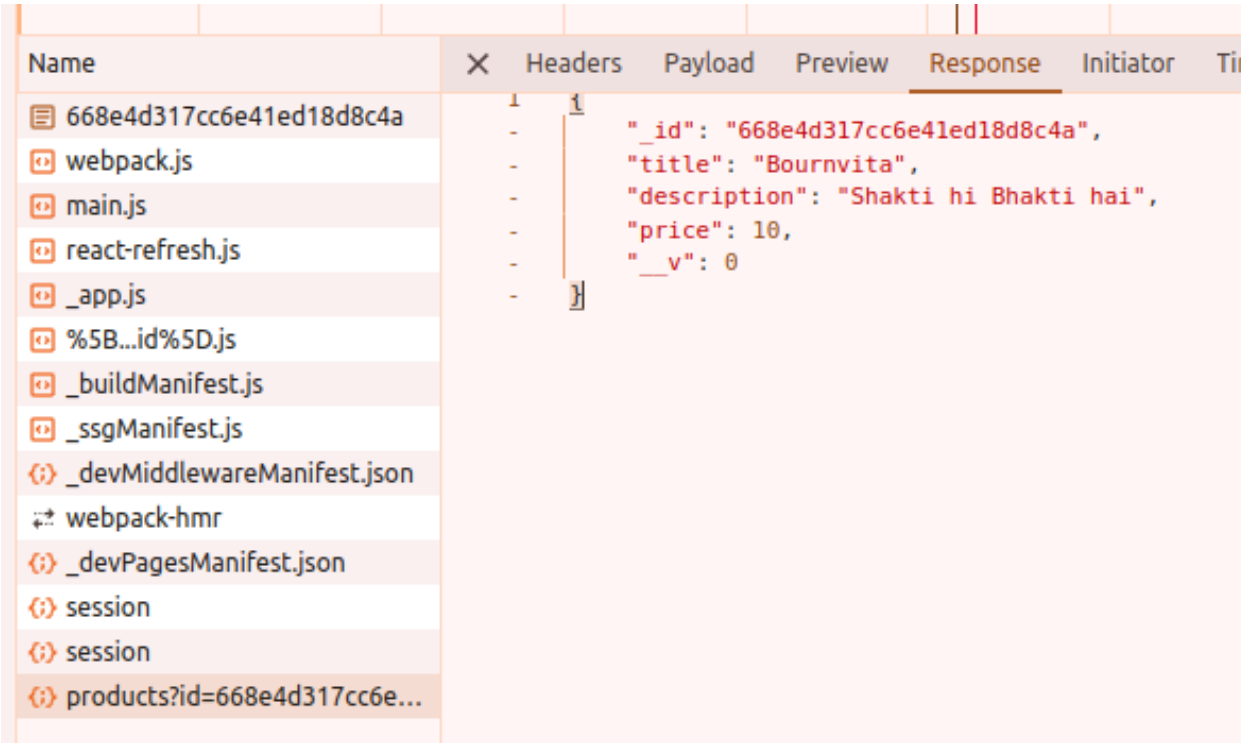
```
import Layout from "@/components/Layout";
import axios from "axios";
import { useRouter } from "next/router";
import { useEffect } from "react";

export default function EditProductPage(){
   const router = useRouter();
   const {id} = router.query;
   useEffect(()=>{
       axios.get('/api/products?id='+id).then(response =>{
           console.log(response.data)
       })
   },[id])
   return (
       <Layout>
           Edit Product From Here
       </Layout>
   )
}
```

Currently we are getting all the products

Now lets only fetch a single product

api/products.js

```
    if(method === 'GET'){
        if(req.query?.id){
            // No need for question mark here since we know query will
exist
            res.json(await Product.findOne({_id:req.query.id}))
        }else{
            res.json(await Product.find())
        }
    }
```



Now create a separate component for the create Product form

components/ProductForm.js

```
import axios from "axios";
import { useRouter } from "next/router";
import { useState } from "react";

export default function ProductForm({
 _id,
```

```
 title: existingTitle,
 description: existingDescription,
 price: existingPrice,
}) {
 const [title, setTitle] = useState(existingTitle || "");
 const [description, setDescription] = useState(existingDescription ||
"");
 const [price, setPrice] = useState(existingPrice || "");
 const [goToProducts, setGoToProducts] = useState(false);
 const router = useRouter();

 async function saveProduct(ev) {
   // We need to send request to api will do through axios
   ev.preventDefault();
   const data = { title, description, price };
   if (_id) {
     // update we will also need the id
     await axios.put('/api/products',{...data,_id});
   } else {
     // create
     await axios.post("/api/products", data);
   }
   setGoToProducts(true);
 }

 if (goToProducts) {
   router.push("/products");
 }

 return (
   <form onSubmit={saveProduct}>
     <label>Product Name</label>
     <input
       type="text"
       placeholder="product name"
       value={title}
       onChange={(ev) => setTitle(ev.target.value)}
     />
     <label>Description</label>
     <textarea
```

```
        placeholder="description"
        value={description}
        onChange={(ev) => setDescription(ev.target.value)}
      ></textarea>
      <label>Price</label>
      <input
        type="number"
        placeholder="price"
        value={price}
        onChange={(ev) => setPrice(ev.target.value)}
      />
      <button type="submit" className="btn-primary">
        Save
      </button>
    </form>
  );
}
```

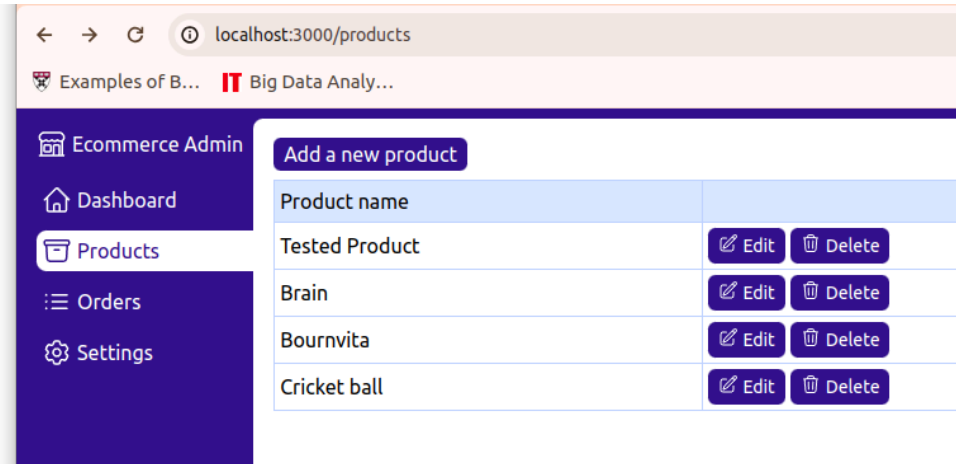
Now we need the put api

```
  if(method === 'PUT'){
      const {title,description,price,_id}= req.body;
      await Product.updateOne({_id},{title,description,price})
      res.json(true);
  }
```

pages/products/edit/[...id].js Looks like this

'Now lets add Link to add Delete button

```
              <Link href={"/products/delete/" + product._id}>
                <svg
                  xmlns="http://www.w3.org/2000/svg"
                  fill="none"
                  viewBox="0 0 24 24"
                  strokeWidth={1.5}
                  stroke="currentColor"
                  className="w-4 h-4"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    d="m14.74 9-.346 9m-4.788 0L9.26
9m9.968-3.21c.342.052.682.107 1.022.166m-1.022-.165L18.16 19.673a2.25 2.25
0 0 1-2.244 2.077H8.084a2.25 2.25 0 0 1-2.244-2.077L4.772 5.79m14.456
0a48.108 48.108 0 0 0-3.478-.397m-12 .562c.34-.059.68-.114 1.022-.165m0
0a48.11 48.11 0 0 1 3.478-.397m7.5
0v-.916c0-1.18-.91-2.164-2.09-2.201a51.964 51.964 0 0 0-3.32
0c-1.18.037-2.09 1.022-2.09 2.201v.916m7.5 0a48.667 48.667 0 0 0-7.5 0"
                  />
                </svg>
                Delete
              </Link>
```



create pages/products/delete/[...id].js

```
import Layout from "@/components/Layout";
import axios from "axios";
import { useRouter } from "next/router";
import { useEffect, useState } from "react";

export default function DeleteProductPage() {
 const router = useRouter();
 const [productInfo, setProductInfo] = useState();
 const { id } = router.query;
 useEffect(() => {
   if (!id) {
     return;
   }
   axios.get("/api/products?id=" + id).then((response) => {
     setProductInfo(response.data);
   });
 }, [id]);
 function goBack() {
   router.push("/products");
 }
 async function deleteProduct(){
   // we can use await since it is not inside the useEffect
   await axios.delete('/api/products?id='+id)
   goBack();
 }
```

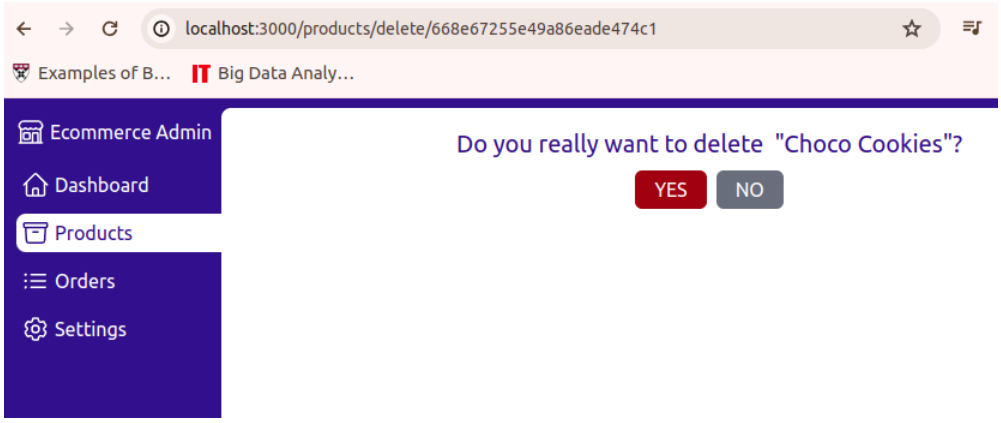
```
 return (
   <Layout>
     <h1 className="text-center">Do you really want to delete
 "{productInfo?.title}"?</h1>
     <div className="flex gap-2 justify-center">
       <button className="btn-red" onClick={deleteProduct}>YES</button>
       <button onClick={goBack} className="btn-default">
         NO
       </button>
     </div>
   </Layout>
 );
}
```

Global.css

```
.btn-red, .btn-default{
   @apply px-4 py-1 rounded-md;
}
.btn-red{
   @apply bg-red-800 text-white
}
.btn-default{
   @apply bg-gray-500 text-white
}
```

Now lets work on delete api

```
   if(method === 'DELETE'){
       if(req.query?.id){
           await Product.deleteOne({_id:req.query?.id});
           res.json(true)
       }
   }
```



In both YES and NO button both will redirect to the products page

Repository at this point

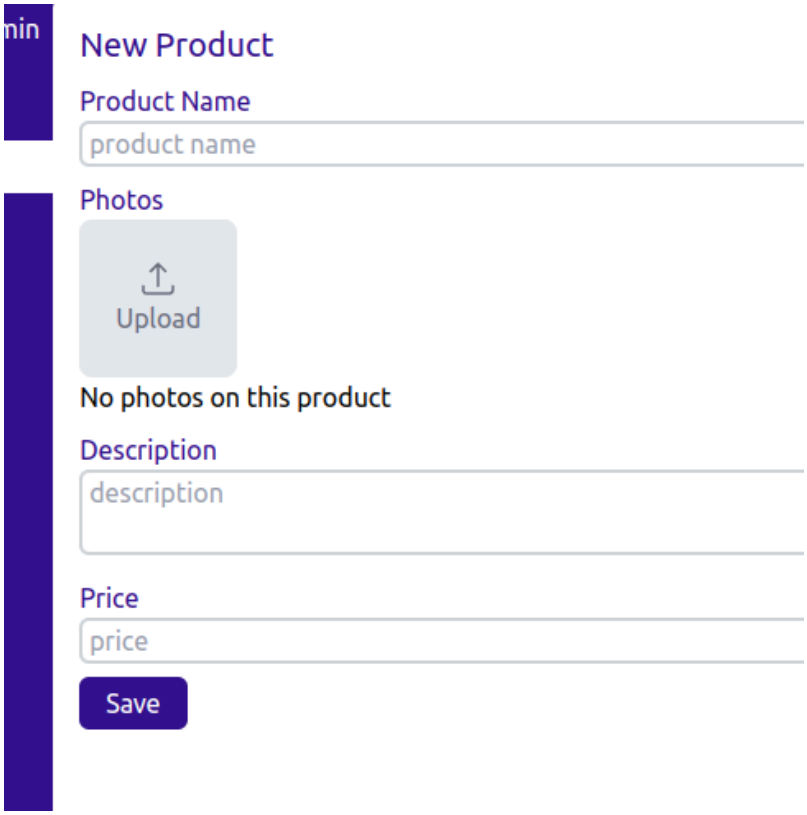
https://github.com/nthapa000/ecommerce-admin/tree/6f83d29e1ef5637a77f526c5dd131283dcbf4dfc

Now lets add Photos

productForm

```
function uploadImages(ev){
   console.log(ev);
 }
.....

  <label>Photos</label>
     <div className="mb-2">
       {/* We want to display all the photos and know the question is how we know how many photos we have */}
       <label className="w-24 h-24  text-center flex flex-col items-center justify-center text-gray-500 rounded-lg bg-gray-200 cursor-pointer">
         <svg
           xmlns="http://www.w3.org/2000/svg"
           fill="none"
           viewBox="0 0 24 24"
           strokeWidth={1.5}
           stroke="currentColor"
```

```
          className="size-6"
        >
          <path
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M3 16.5v2.25A2.25 2.25 0 0 0 5.25 21h13.5A2.25 2.25 0 0 0
21 18.75V16.5m-13.5-9L12 3m0 0 4.5 4.5M12 3v13.5"
          />
        </svg>
        <div>
          Upload
        </div>
        {/* due to this we can pick a file from the computer */}
        <input type="file" onChange={uploadImages} className="hidden"/>
      </label>
      {!images?.length && <div>No photos on this product</div>}
    </div>
```



When selecting an file from system we get following log

Will be creating separate api file for upload

```
export default async function handle(req,res){


}

export const config = {
   api: {bodyParser:false},
}
```

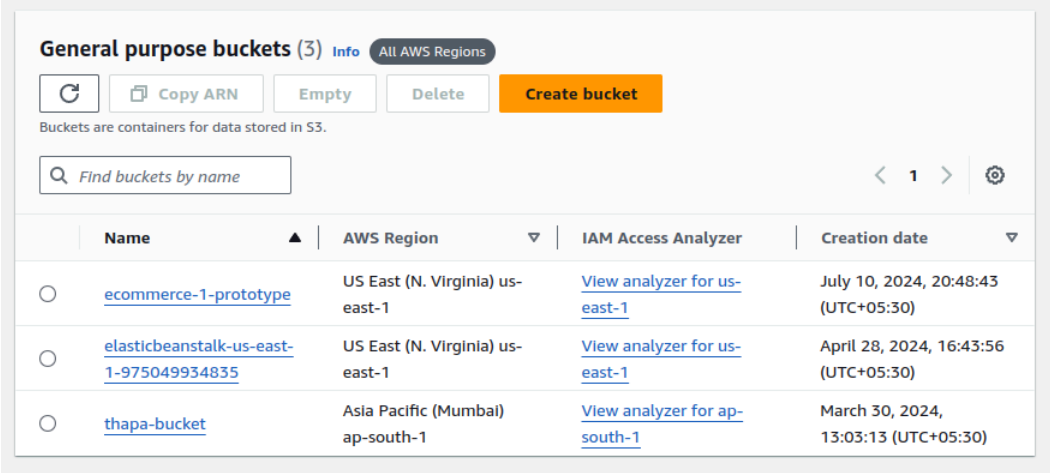
To get the file from the req we will be using package called **multiparty**

npm install add multiparty

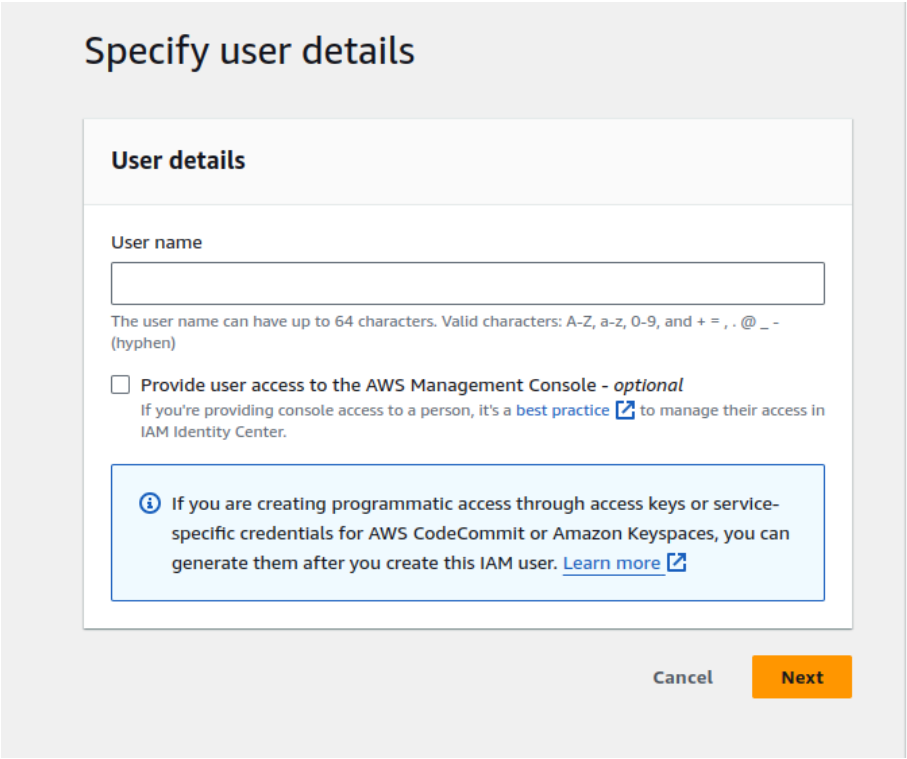
```
import multiparty from 'multiparty'

export default async function handle(req,res){
   const form =new multiparty.Form();
```

```
    const {fields,files} = await new Promise((resolve,reject)=>{
        form.parse(req, async(err,fields,files)=>{
            if(err) reject(err);
            resolve({fields,files})
        })
    })
    console.log('length:',files.file.length);

    return res.json('ok');
}

export const config = {
    api: {bodyParser:false},
}
```

Now we will store the photos in the s3 bucket



Security Credentials -> users -> Create user

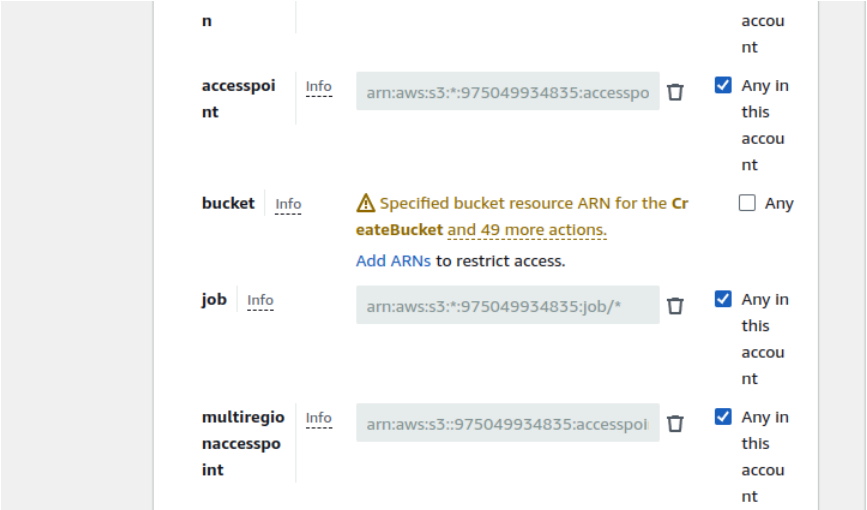


Give username-> attach policies -> create policy

Choose S3

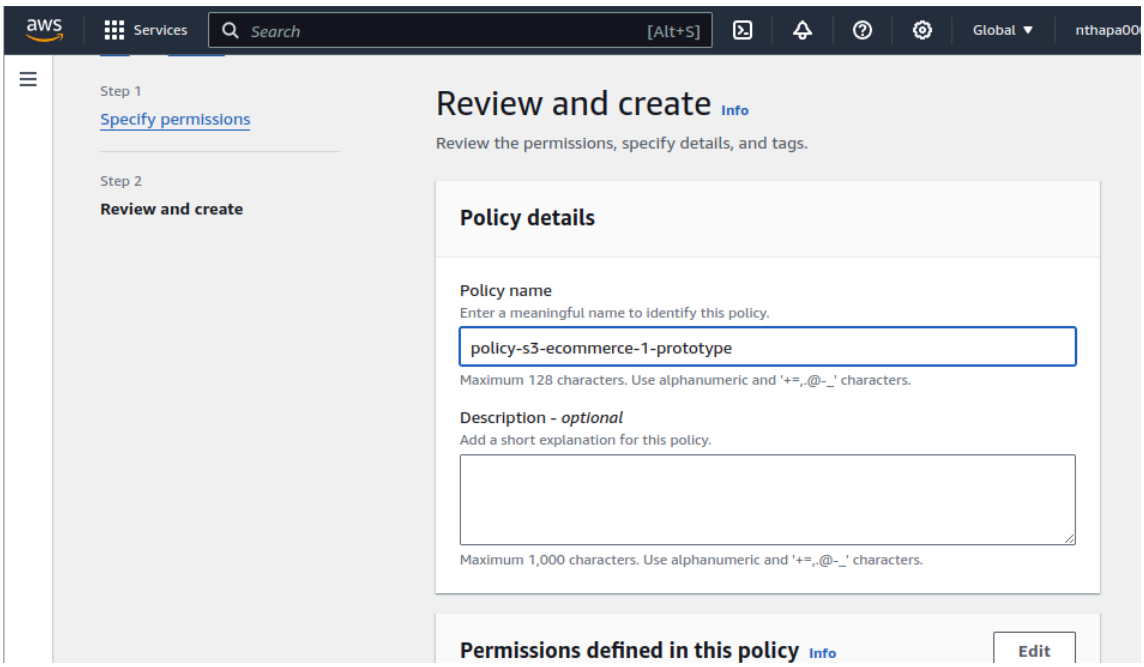
Allow all S3 actions

On resources all any in this account for all except bucket

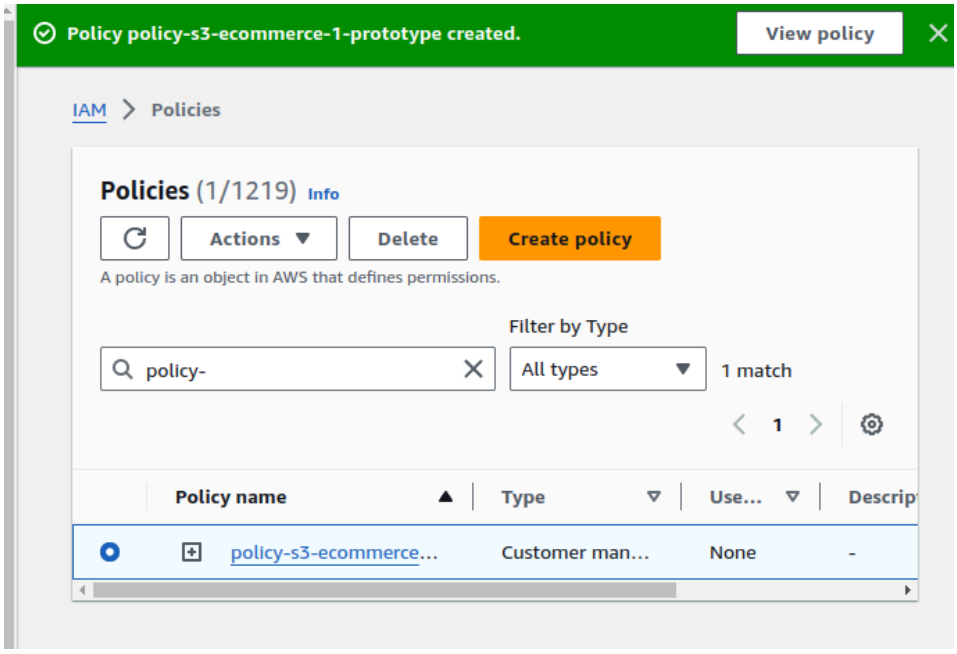


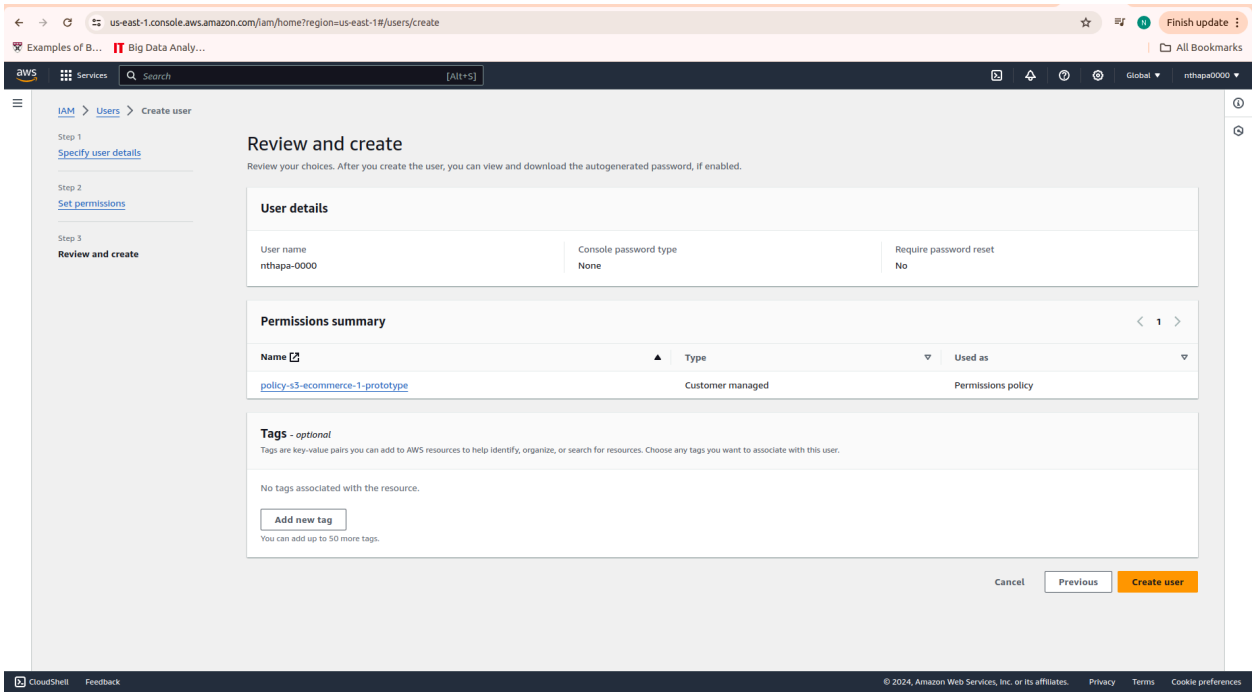
For bucket add ARN

Next

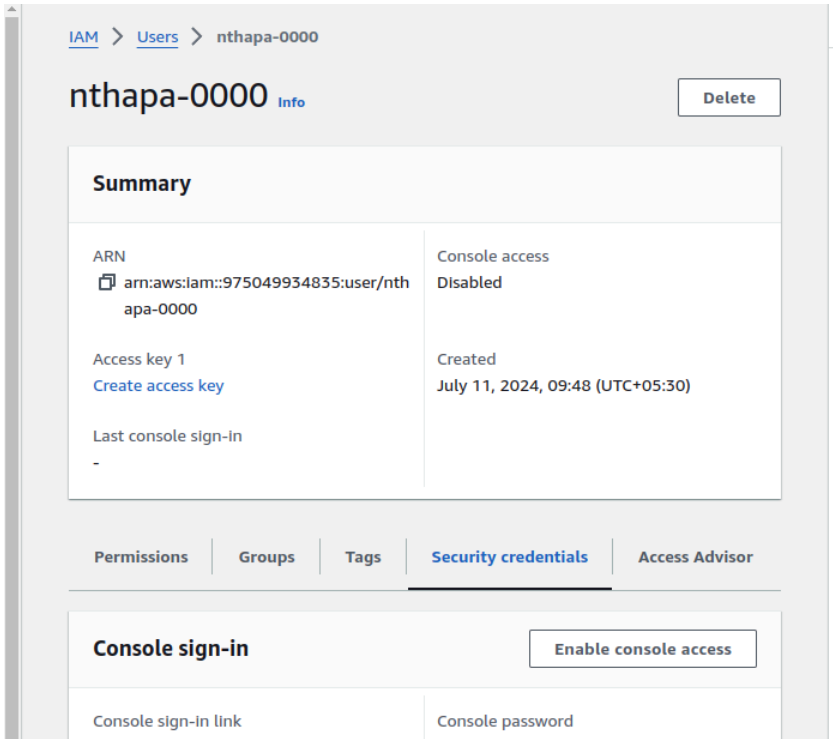


Create policy

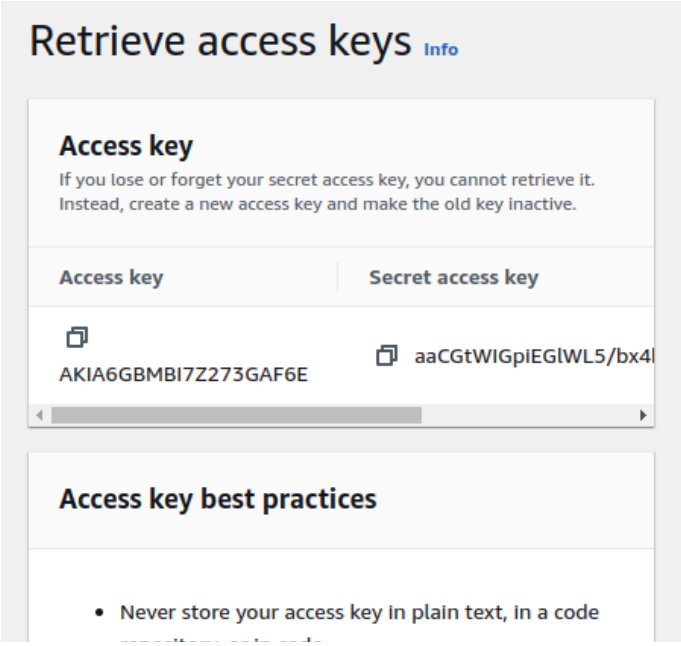




Select the user -> Click on the Security credentials tab and then create access key



Select the radio button with option Application running outside AWS -> Next -> Create access key



Access key: AKIA6GBMBI7Z273GAF6E

Secret access key: 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

And Store these in .env files

Choose S3 bucket ecommerce-1-prototype

Now we will upload something inside here

npm install @aws-sdk/client-s3

pages/api/upload.js

```
import multiparty from 'multiparty'
import {PutObjectCommand, S3Client} from '@aws-sdk/client-s3'
const bucketName = 'ecommerce-1-prototype'

export default async function handle(req,res){
   const form =new multiparty.Form();
   const {fields,files} = await new Promise((resolve,reject)=>{
       form.parse(req, async(err,fields,files)=>{
```

```
            if(err) reject(err);
            resolve({fields,files})
        })
    })

    console.log('length:',files.file.length);
    const client = new S3Client({
        region:'us-east-1',
        credentials:{
            accessKeyId:process.env.S3_ACCESS_KEY,
            secretAccessKey:process.env.S3_SECRET_ACCESS_KEY
        }
    });
    // From already known file name we want the extension of the file
    for(const file of files.file){
        const ext = file.originalFilename.split('.').pop();
        console.log({ext,file})
    //    await client.send(new PutObjectCommand({
    //          Bucket:bucketName,
    //          Key:''
    //     }));
    }

    return res.json('ok');
}

export const config = {
    api: {bodyParser:false},
}
```

Now we can see on console.log

```
 GET /api/auth/session 304 in 197ms
 GET /api/auth/session 304 in 193ms
 ✓ Compiled /api/upload in 121ms (151 modules)
length: 1
{
  ext: 'png',
  file: {
    fieldName: 'file',
    originalFilename: 'logo.png',
    path: '/tmp/4h0yqsfXS2M9ocZSWUKKFdYH.png',
    headers: {
      'content-disposition': 'form-data; name="file"; filename="lo
go.png"',
      'content-type': 'image/png'
    },
    size: 5121
  }
}
 POST /api/upload 200 in 233ms
 ✓ Compiled in 191ms (453 modules)
```

We will store file name randomly

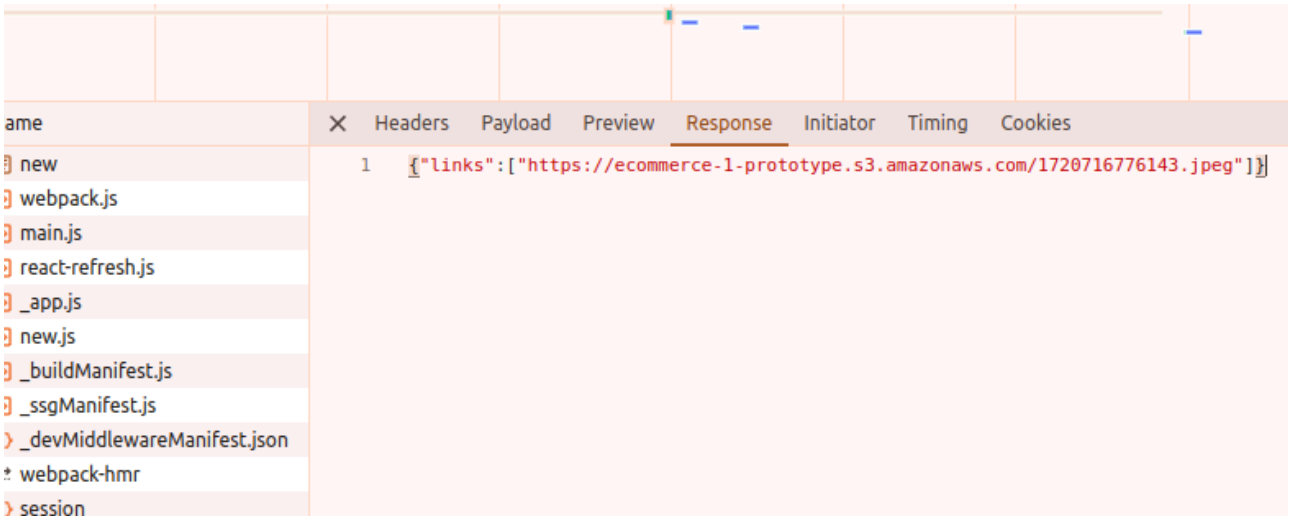
npm add mime-types

To know the type of the file

```
import multiparty from 'multiparty'
import {PutObjectCommand, S3Client} from '@aws-sdk/client-s3'
// to read the content of body
import fs from 'fs'
import mime from 'mime-types'
const bucketName = 'ecommerce-1-prototype'

export default async function handle(req,res){
   const form =new multiparty.Form();
   const {fields,files} = await new Promise((resolve,reject)=>{
       form.parse(req, async(err,fields,files)=>{
           if(err) reject(err);
           resolve({fields,files})
       })
   })
```

```
    console.log('length:',files.file.length);
    const client = new S3Client({
        region:'us-east-1',
        credentials:{
            accessKeyId:process.env.S3_ACCESS_KEY,
            secretAccessKey:process.env.S3_SECRET_ACCESS_KEY
        }
    });
    // From already known file name we want the extension of the file
    // In case we have several files
    const links = [];
    for(const file of files.file){
        const ext = file.originalFilename.split('.').pop();
        const newFilename = Date.now() + '.'+ext;
        console.log({ext,file})
       await client.send(new PutObjectCommand({
            Bucket:bucketName,
            Key:newFilename,
            Body:fs.readFileSync(file.path),
            // So that the file is publically available and we can just
give the link
            ACL:'public-read',
            ContentType:mime.lookup(file.path)
        }));
        const link =
`https://${bucketName}.s3.amazonaws.com/${newFilename}`;
        links.push(link);
    }

    return res.json({links});
}

export const config = {
    api: {bodyParser:false},
}
```

Now on the network we get the response as url



Now we need to display every uploaded file

ProductForm

```
<label>Photos</label>
     <div className="mb-2 flex flex-wrap gap-2 ">
       {/* had to convert it to boolean */}
       {!!images?.length && images.map(link => (
           <div key={link} className=" h-24">
               <img src={link} alt="" className="rounded-lg"/>
               {/* {link} I want it as images */ }
           </div>
       ))}
       {/* We want to display all the photos and know the question is how
we know how many photos we have */}
       <label className=" w-24 h-24  text-center flex flex-col
items-center justify-center text-gray-500 rounded-lg bg-gray-200
cursor-pointer">
         <svg
           xmlns="http://www.w3.org/2000/svg"
           fill="none"
           viewBox="0 0 24 24"
           strokeWidth={1.5}
           stroke="currentColor"
           className="size-6"
         >
           <path
```

```
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M3 16.5v2.25A2.25 2.25 0 0 0 5.25 21h13.5A2.25 2.25 0 0 0
21 18.75V16.5m-13.5-9L12 3m0 0 4.5 4.5M12 3v13.5"
          />
        </svg>
        <div>
          Upload
        </div>
        {/* due to this we can pick a file from the computer */}
        <input type="file" onChange={uploadImages} className="hidden"/>
      </label>
      {!images?.length && <div>No photos on this product</div>}
    </div>
```

Now we want the images link to be stored in products table but currently in product schema there is no such column

```
const {model, Schema, models } = require("mongoose");

const ProductSchema = new Schema({
   title: {type:String, required:true},
   description: String,
   price: {type:Number, required:true},
   images:{type:[String]},
});

export const Product =models.Product || model('Product',ProductSchema);
```

Now we are able to store the images in db

```
description : "Shakti hi Bhakti hai"
price : 10
__v : 0
```

```
_id: ObjectId('668e54415e49a86eade4742e')
title : "Cricket ball"
description : "Red in Color "
price : 50
__v : 0
images : Array (1)
  0: "https://ecommerce-1-prototype.s3.amazonaws.com/1720722299081.jpeg"
```

```
_id: ObjectId('668e67255e49a86eade474c1')
title : "Choco Cookies"
description : "100 Percent made with organic items"
price : 100
```


And it is also present there in the product if we save and reopen it

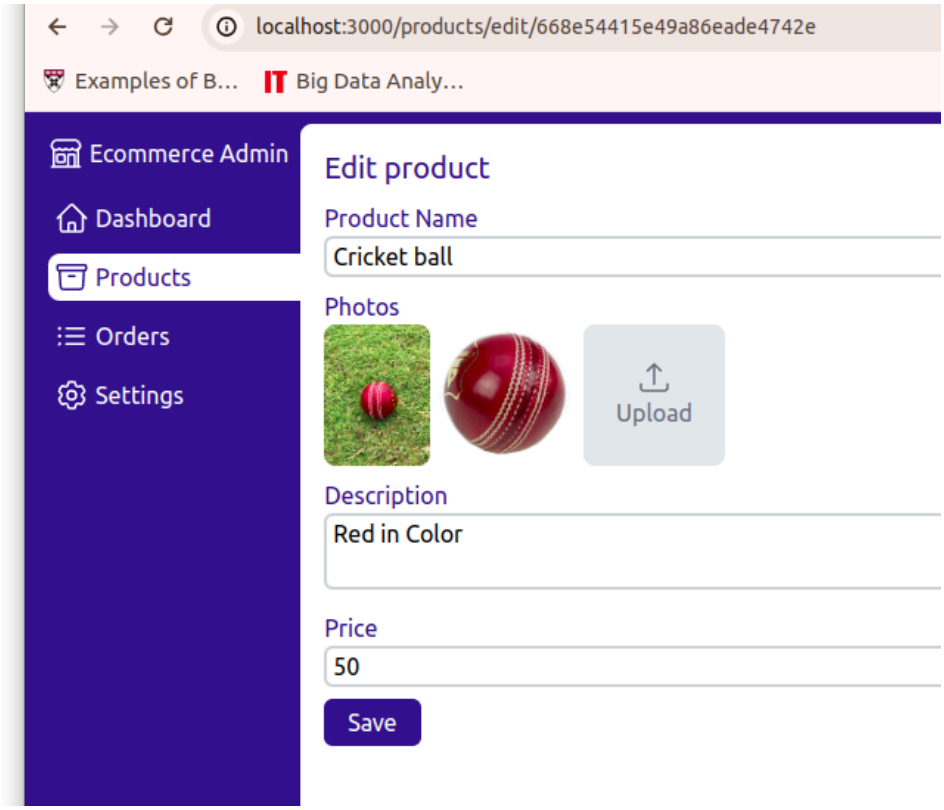
Now we want an indicator when we are uploading something

npm install react-spinners

components/spinner.js

```
import { BounceLoader } from "react-spinners";

export default function Spinner(){
   return (
       <BounceLoader color={'#1E3A8A'} speedMultiplier={2}/>
   )
}
```

We can choose other spinners according to our need.



Now we want to add a functionality where we can drag the images according to our priority

npm install react-sortablejs

npm install sortablejs

```
import { ReactSortable } from "react-sortablejs";
.....
function updateImagesOrder(images){
   setImages(images);
}
------------

<ReactSortable list={images} setList={updateImagesOrder} className="flex
flex-wrap gap-1">
       {!!images?.length && images.map(link => (
           <div key={link} className=" h-24">
               <img src={link} alt="" className="rounded-lg"/>
               {/* {link} I want it as images */ }
```

```
            </div>
        ))}
        </ReactSortable>
```

Drag and drop the images according to the order you want and then click on the save to save the desired order.

Repository at this moment:

https://github.com/nthapa000/ecommerce-admin/tree/d030de516ae95ff60d5be8c8c099f26e0930a03d

Admin - Categories Management

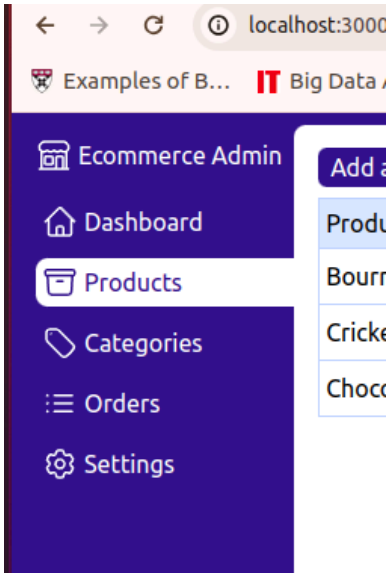
Each product need to have a category

components/nav.js

```
<Link
          href={"/categories"}
          className={pathname.includes("/categories") ? activeLink :
inactiveLink}
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            strokeWidth={1.5}
            stroke="currentColor"
            className="size-6"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="M9.568 3H5.25A2.25 2.25 0 0 0 3 5.25v4.318c0 .597.237
1.17.659 1.59119.581 9.581c.699.699 1.78.872 2.607.33a18.095 18.095 0 0 0
5.223-5.223c.542-.827.369-1.908-.33-2.607L11.16 3.66A2.25 2.25 0 0 0 9.568
3Z"
            />
            <path
```

```
            strokeLinecap="round"
            strokeLinejoin="round"
            d="M6 6h.008v.008H6V6Z"
          />
        </svg>
        Categories
      </Link>
```

It looks like this



pages/categories.js

```
import Layout from "@/components/Layout";
import axios from "axios";
import { useState } from "react";

export default function CategoriesPage() {
 const [name, setName] = useState("");
 async function saveCategory() {
   await axios.post('/api/categories',{name});
   setName('');
 }
 return (
   <Layout>
     <h1>Categories</h1>
     <label>New Category Name</label>
```

```
      <form onSubmit={saveCategory} className="flex gap-1">
        <input
          className="mb-0"
          type="text"
          placeholder={"Category name"}
          onChange={ev => setName(ev.target.value)}
          value={name}
        />
        <button type={"submit"} className="btn-primary py-1 ">
          Save
        </button>
      </form>
    </Layout>
  );
}
```

Later we will display the categories

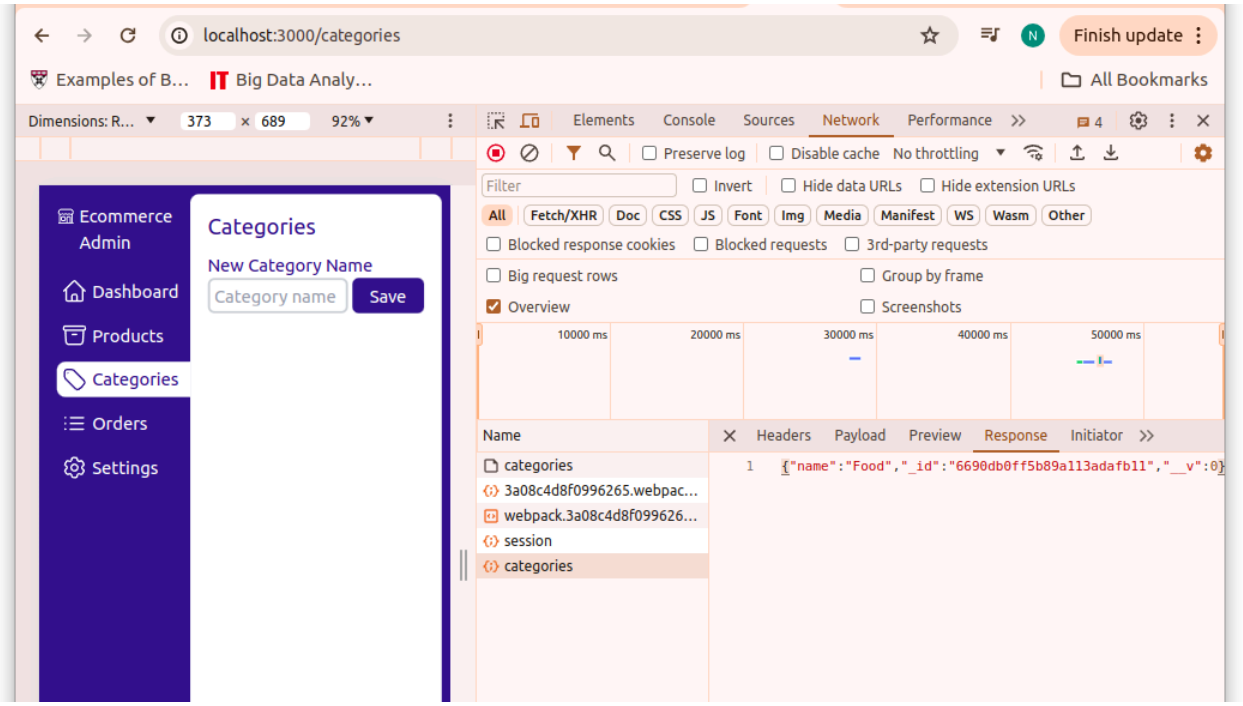
api/categories.js

```
export default function handle(req,res){
    const {method} = req;

    if(method === 'POST'){
        const {name} = req.body;
    }
}
```

We will complete it later now we will create model for the Categories

model/category.js

```
const { Schema, model, models } = require("mongoose");

const CategorySchema = new Schema({
    name:{type:String, required:true},
});

export const Category = models?.Category ||
model('Category',CategorySchema)
```



api/Categories

```
import { Category } from "@/model/Category";

export default async function handle(req,res){
    const {method} = req;

    if(method === 'POST'){
        const {name} = req.body;
        const categoryDoc = await Category.create({name})
        res.json(categoryDoc)
    }
}
```

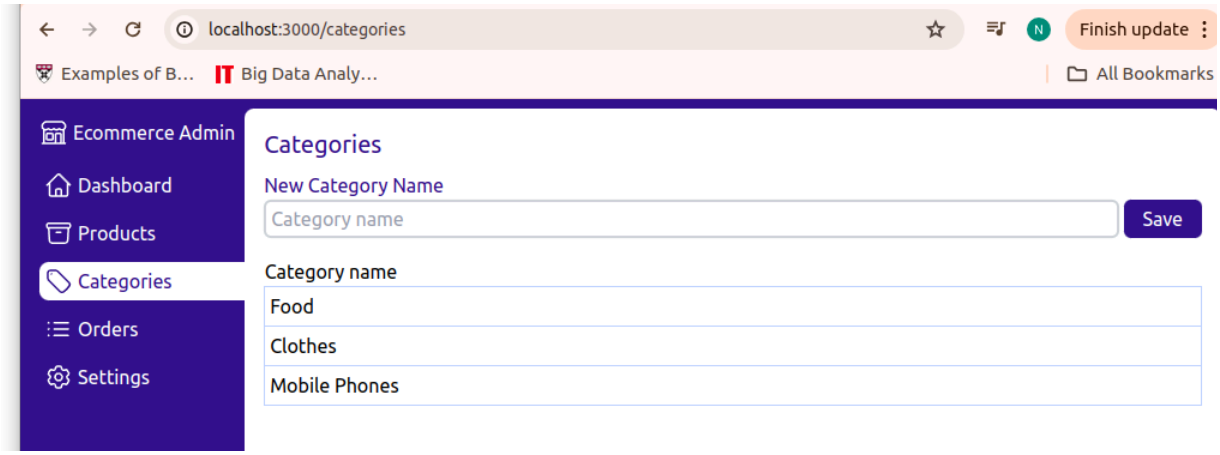
We can check in database it is inserted

Now we will display the categories

Before that we need endpoint from where we can get all the categories from the database

```
  if(method==='GET'){
      res.json(await Category.find());
  }
```

Now we suppose want to add subcategories like android for Mobile phones

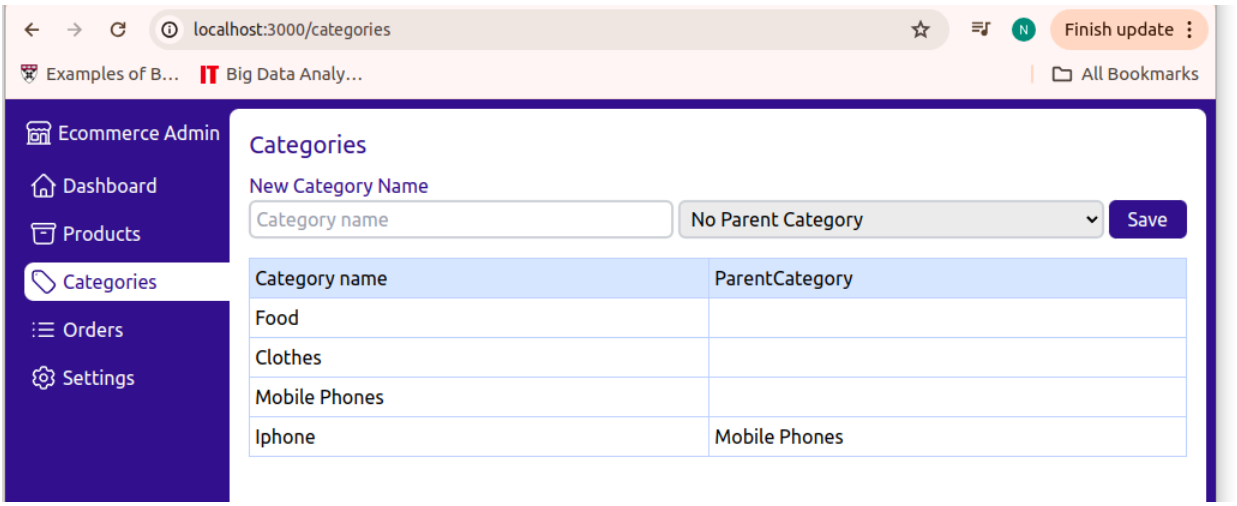



```
import Layout from "@/components/Layout";
import axios from "axios";
import { useEffect, useState } from "react";

export default function CategoriesPage() {
 const [name, setName] = useState("");
 //   Fetching all the categories
 const [parentCategory, setParentCategory] = useState("");
 const [categories, setCategories] = useState([]);

 useEffect(() => {
   fetchCategories();
 }, []);
 function fetchCategories() {
   axios.get("/api/categories").then((result) => {
     setCategories(result.data);
   });
 }
 async function saveCategory(ev) {
   ev.preventDefault();
   await axios.post("/api/categories", { name, parentCategory });
   setName("");
```

```
  fetchCategories();
}
return (
  <Layout>
    <h1>Categories</h1>
    <label>New Category Name</label>
    <form onSubmit={saveCategory} className="flex gap-1">
      <input
        className="mb-0"
        type="text"
        placeholder={"Category name"}
        onChange={(ev) => setName(ev.target.value)}
        value={name}
      />
      <select
        className="mb-0"
        onChange={(ev) => setParentCategory(ev.target.value)}
        value={parentCategory}
      >
        <option value="">No Parent Category</option>
        {/* all the categories we have */}
        {categories.length > 0 &&
          categories.map((category) => (
            <option key={category._id} value={category._id}>
              {category.name}
            </option>
          ))}
      </select>
      <button type={"submit"} className="btn-primary py-1 ">
        Save
      </button>
    </form>
    <table className="basic mt-4">
      <thead>
        <tr>
          <td>Category name</td>
          <td>ParentCategory</td>
        </tr>
      </thead>
      <tbody>
```

```
          {categories.length > 0 &&
            categories.map((category) => (
              <tr key={category._id}>
                <td>{category.name}</td>
                <td>{category?.parent?.name}</td>
              </tr>
            ))}
        </tbody>
      </table>
    </Layout>
  );
}
```

Now we can see the parent category



Now we will add delete and edit button to delete the categories

Repository till moment:

https://github.com/nthapa000/ecommerce-admin/commits/main/

Edit

Put api

```
if(method === 'PUT'){
   const { name, parentCategory,_id } = req.body;
```

```
   const categoryDoc = await Category.updateOne({_id},{ name, parent:
parentCategory, });
   res.json(categoryDoc);
 }
```

npm install react-sweetalert2

Repository:

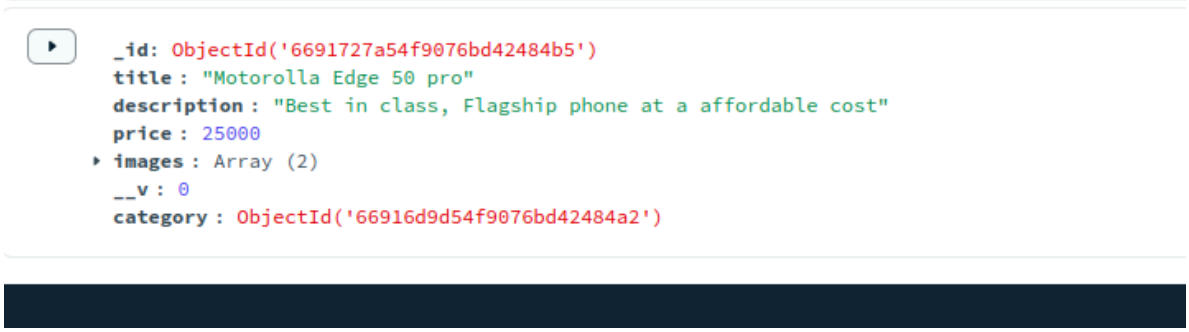
https://github.com/nthapa000/ecommerce-admin/tree/9b155c359aef002a13e29206434e8e1dfed31168

Now we able to assign category to product

Lets start by inserting some images on some product, making product for motorola edge 50 pro

Now lets add category section in the ProductForm:

Add label for Category, Check whether there is some category already exist for the product in case we have to fetch it and then if no category maintain a state variable to store the category , update the api endpoint for the products to store the category as id



Now we can see the category

Now lets have separate property section for products so that we can showcase it in the frontend, For that we have to assign the properties to the Category first

```
<div className="mb-2">
          <label className="block">Properties</label>
          {/* to need properties we need a state variable */}
          <button
            type="button"
            onClick={addProperty}
            className="btn-default text-sm"
          >
            Add new property
          </button>
          {properties.length > 0 && properties.map((property,index) => (
              <div className="flex gap-1" key={property._id}>
                <input
                  type="text"
                  value={property.value}
                  onChange={ev =>
handlePropertyNameChange(index,property,ev.target.value)}
                  placeholder="property name (example: color)"
                />
                <input
                  type="text"
                  value={property.value}
                  placeholder="values, comma separated"
                />
              </div>
            ))}
        </div>
```

We are currently getting error before we remove the last property
It is happening because we haven't defined the type of the button and hence it is trying to be send

```
<button
                  className="btn-default "
```

```
                onClick={() => removeProperty(index)}
                type="button"
              >
                Remove
              </button>
```

Now when we click on the certain category edit option then the properties should reload

Finaly the categories inserted in the database

Repository till now:
https://github.com/nthapa000/ecommerce-admin/tree/99e10df593a7820129f5fa40ea94ac85fa2a92e3

Now when we go to the product , suppose we choose iphone 14 pro , then even if the category is Iphone , I also want to see the properties from the parent of Iphone , mobile phone

ProductForm.jsx

Repository till now:
https://github.com/nthapa000/ecommerce-admin/tree/23dd040d2ecd5a0af107d011fb7b148f1b0e446c

Now lets work on the security

Lets first work on the Logout button

```
        <button onClick={() => signOut()} className={inactiveLink}>
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            strokeWidth={1.5}
            stroke="currentColor"
            className="size-6"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="M8.25 9V5.25A2.25 2.25 0 0 1 10.5 3h6a2.25 2.25 0 0 1 2.25
2.25v13.5A2.25 2.25 0 0 1 16.5 21h-6a2.25 2.25 0 0 1-2.25-2.25V15m-3
0-3-3m0 0 3-3m-3 3H15"
            />
          </svg>
          Logout
        </button>
```

It is doing logout and login but anyone can come and change

api/nextauth