1. Create-react-app
2. Make component folder and put the Heder , Footer , Products , Rating into component
3. Using a products.js array of producst and display it on product folder
4. Installing react-bootstrap to use custom components
5. Coping the bootswatch template to src folder
6. Installing react-router-dom
7. Route component for SPA , and instead using href use a to
8. We can route the bootstrap route container
9. We use params /product/:id as match.params
10. Fetching product from react useEffect
11. Connect mongodb to atlas and compass
12. Then using mongoose setup a app

OSNOVNI SETUP ZA BACKEND EXPRESS.JS

const express = require('express')

const products = require('./data/products')

const app = express();

app.get('/', (req, res) => {

    res.send('API is running...')

})

app.get('/api/products', (req, res) => {

    res.json(products)

})

app.get('/api/products/:id', (req, res) => {

    const product = products.find(p => p.\_id === req.params.id)

    res.json(product)

})

app.listen(5000, console.log('Server runing on port 5000'))

THINGS LIKE PRODUCTS , USERS , CART IS GLOBAL STATE ,

USING Forms and Inputs and simple stuffs we use a useState

Fetching data from server with axios

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { Row, Col } from 'react-bootstrap';

import Product from '../components/Product';

const HomeScreen = () => {

    const [products, setProducts] = useState([])

    useEffect(() => {

        const fetchProducts = async () => {

            const { data } = await axios.get('/api/products')

            setProducts(data)

        }

        fetchProducts();

    }, [])

    return (

        <>

ENV in global we can put a API keys,global variables

Connecting Mongo Db and Node.js server

import mongoose from 'mongoose'

const connectDB = async () => {

    try {

        const conn = await mongoose.connect(process.env.MONGO\_URI, {

            useUnifiedTopology: true,

            useNewUrlParser: true,

            useCreateIndex: true

        })

        console.log(`MongoDB Connected: ${conn.connection.host}`)

    } catch (error) {

        console.error(`Error:${error.message}`)

        process.exit(1);

    }

}

export default connectDB;

and import to server.js

installing colors to node.js server

Creating Mongoose schema

import mongoose from 'mongoose'

const userSchema = mongoose.Schema({

    name: {

        type:String,

        required:true

    },

    email: {

        type:String,

        required:true,

        unique: true

    },

    password: {

        type:String,

        required:true

    },

    isAdmin: {

        type:Boolean,

        required:true,

        default: false

    }

}, {

    timestamps: true

})

const User = mongoose.model('User', userSchema)

export default User

 user: {

        type:mongoose.Schema.Types.ObjectId,

        required:true,

        ref: 'User'

    },

This define connect between product and user

Product schema

import mongoose from 'mongoose';

const reviewSchema = mongoose.Schema({

    name: { type: String, required: true},

    rating: { type: Number, required: true},

    comment: { type: String, required: true},

}, {timestamps: true})

const productSchema = mongoose.Schema({

    user: {

        type:mongoose.Schema.Types.ObjectId,

        required:true,

        ref: 'User'

    },

    name: {

        type:String,

        required:true

    },

    image: {

        type:String,

        required:true

    },

    brand: {

        type:String,

        required:true

    },

    category: {

        type:String,

        required:true

    },

    description: {

        type:String,

        required:true

    },

    reviews: [reviewSchema],

    rating: {

        type: Number,

        required: true,

        default: 0

    },

    numReviews: {

        type: Number,

        required: true,

        default:0

    },

    price: {

        type: Number,

        required: true,

        default: 0

    },

    countInStock: {

        type: Number,

        required: true,

        default: 0

    },

}, {

    timestamps: true

})

const Product = mongoose.model('Product', productSchema)

export default User

import mongoose from 'mongoose';

const orderSchema = mongoose.Schema({

    name: { type: String, required: true},

    rating: { type: Number, required: true},

    comment: { type: String, required: true},

}, {timestamps: true})

const productSchema = mongoose.Schema({

    user: {

        type:mongoose.Schema.Types.ObjectId,

        required:true,

        ref: 'User'

    },

    orderItems: [

        {

            name: { type: String, required: true},

            qty: { type: String, required: true},

            image: { type: String, required: true},

            price: { type: Number, required: true},

            product: {

                type: mongoose.Schema.Types.ObjectId,

                required: true,

                ref: 'Product'

            },

        }

    ],

    shippingAddress: {

        address: { type: String, required: true},

        city: { type: String, required: true},

        postalCode: { type: String, required: true},

        country: { type: String, required: true}

    },

    paymentResult: {

        id: { type: String},

        status: { type: String},

        update\_time: { type: String},

        email\_address: { type: String}

    },

    taxPrice: {

        type: Number,

        required: true,

        default: 0.0

    },

    shippingPrice: {

        type: Number,

        required: true,

        default: 0.0

    },

    totalPrice: {

        type: Number,

        required: true,

        default: 0.0

    },

    isPaid: {

        type: Boolean,

        required: true,

        default: false

    },

    paidAt: {

        type: Date,

    },

    isDelivered: {

        type: Boolean,

        required: true,

        default: false

    },

    deliveredAt: {

        type: Date,

    },

}, {

    timestamps: true

})

const Order = mongoose.model('Order', OrderSchema)

export default User