SOURCE CODE:

Home.js:

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { Checkbox, Radio } from "antd";
import { Prices } from "../components/Prices";
import { useCart } from "../context/cart";
import axios from "axios";
import toast from "react-hot-toast";
import Layout from "./../components/Layout/Layout";
import { AiOutlineReload } from "react-icons/ai";
import "../styles/Homepage.css";

const HomePage = () => {
  const navigate = useNavigate();
  const [cart, setCart] = useCart();
  const [products, setProducts] = useState([]);
  const [categories, setCategories] = useState([]);
  const [checked, setChecked] = useState([]);
  const [radio, setRadio] = useState([]);
  const [total, setTotal] = useState(0);
  const [page, setPage] = useState(1);
  const [loading, setLoading] = useState(false);

  //get all cat
  const getAllCategory = async () => {
    try {
      const { data } = await axios.get("/api/v1/category/get-category");
      if (data?.success) {
        setCategories(data?.category);
      }
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    getAllCategory();
    getTotal();
  }, []);
  //get products
```

```
const getAllProducts = async () => {
  try {
    setLoading(true);
    const { data } = await axios.get(`/api/v1/product/product-list/${page}`);
    setLoading(false);
    setProducts(data.products);
  } catch (error) {
    setLoading(false);
    console.log(error);
  }
};

//getTOtal COunt
const getTotal = async () => {
  try {
    const { data } = await axios.get("/api/v1/product/product-count");
    setTotal(data?.total);
  } catch (error) {
    console.log(error);
  }
};

useEffect(() => {
  if (page === 1) return;
  loadMore();
}, [page]);
//load more
const loadMore = async () => {
  try {
    setLoading(true);
    const { data } = await axios.get(`/api/v1/product/product-list/${page}`);
    setLoading(false);
    setProducts([...products, ...data?.products]);
  } catch (error) {
    console.log(error);
    setLoading(false);
  }
};

// filter by cat
const handleFilter = (value, id) => {
  let all = [...checked];
  if (value) {
    all.push(id);
  } else {
```

```
    all = all.filter((c) => c !== id);
  }
  setChecked(all);
};
useEffect(() => {
  if (!checked.length || !radio.length) getAllProducts();
}, [checked.length, radio.length]);

useEffect(() => {
  if (checked.length || radio.length) filterProduct();
}, [checked, radio]);

//get filterd product
const filterProduct = async () => {
  try {
    const { data } = await axios.post("/api/v1/product/product-filters", {
      checked,
      radio,
    });
    setProducts(data?.products);
  } catch (error) {
    console.log(error);
  }
};
return (
  <Layout title={"E-RKV STORE"}>
    <div className="container-fluid row mt-3 home-page">
      <div className="col-md-3 filters">
        <h4 className="text-center">Filter By Category</h4>
        <div className="d-flex flex-column">
          {categories?.map((c) => (
            <Checkbox
              key={c._id}
              onChange={(e) => handleFilter(e.target.checked, c._id)}
            >
              {c.name}
            </Checkbox>
          ))}
        </div>
        {/* price filter */}
        <h4 className="text-center mt-4">Filter By Price</h4>
        <div className="d-flex flex-column">
          <Radio.Group onChange={(e) => setRadio(e.target.value)}>
            {Prices?.map((p) => (
              <div key={p._id}>
```

```
            <Radio value={p.array}>{p.name}</Radio>
          </div>
        ))}
      </Radio.Group>
    </div>
    <div className="d-flex flex-column">
      <button
        className="btn btn-danger"
        onClick={() => window.location.reload()}
      >
        RESET FILTERS
      </button>
    </div>
  </div>
  <div className="col-md-9 ">
    <h1 className="text-center">All Products</h1>
    <div className="d-flex flex-wrap">
      {products?.map((p) => (
        <div className="card m-2" key={p._id}>
          <img
            src={`/api/v1/product/product-photo/${p._id}`}
            className="card-img-top"
            alt={p.name}
          />
          <div className="card-body">
            <div className="card-name-price">
              <h5 className="card-title">{p.name}</h5>
              <h5 className="card-title card-price">
                {p.price.toLocaleString("en-US", {
                  style: "currency",
                  currency: "USD",
                })}
              </h5>
            </div>
            <p className="card-text ">
              {p.description.substring(0, 60)}...
            </p>
            <div className="card-name-price">
              <button
                className="btn btn-info ms-1"
                onClick={() => navigate(`/product/${p.slug}`)}
              >
                More Details
              </button>
              <button
```

```
                    className="btn btn-dark ms-1"
                    onClick={() => {
                      setCart([...cart, p]);
                      localStorage.setItem(
                        "cart",
                        JSON.stringify([...cart, p])
                      );
                      toast.success("Item Added to cart");
                    }}
                  >
                    ADD TO CART
                  </button>
                </div>
              </div>
            </div>
          ))}
        </div>
        <div className="m-2 p-3">
          {products && products.length < total && (
            <button
              className="btn loadmore"
              onClick={(e) => {
                e.preventDefault();
                setPage(page + 1);
              }}
            >
              {loading ? (
                "Loading ..."
              ) : (
                <>
                  {" "}
                  Loadmore <AiOutlineReload />
                </>
              )}
            </button>
          )}
        </div>
      </div>
    </div>
  </Layout>
 );
};

export default HomePage;
```

Admin Dashboard:

```
import React from "react";
import AdminMenu from "../../components/Layout/AdminMenu";
import Layout from "./../../components/Layout/Layout";
import { useAuth } from "../../context/auth";
const AdminDashboard = () => {
  const [auth] = useAuth();
  return (
    <Layout>
      <div className="container-fluid m-3 p-3 dashboard">
        <div className="row">
          <div className="col-md-3">
            <AdminMenu />
          </div>
          <div className="col-md-9">
            <div className="card w-75 p-3">
              <h3> Admin Name : {auth?.user?.name}</h3>
              <h3> Admin Email : {auth?.user?.email}</h3>
              <h3> Admin Contact : {auth?.user?.phone}</h3>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default AdminDashboard;
```

Backend:

Authcontroller.js:

```
import userModel from "../models/userModel.js";
import orderModel from "../models/orderModel.js";

import { comparePassword, hashPassword } from "./../helpers/authHelper.js";
import JWT from "jsonwebtoken";
```

```
export const registerController = async (req, res) => {
  try {
    const { name, email, password, phone, address, answer } = req.body;
    //validations
    if (!name) {
      return res.send({ error: "Name is Required" });
    }
    if (!email) {
      return res.send({ message: "Email is Required" });
    }
    if (!password) {
      return res.send({ message: "Password is Required" });
    }
    if (!phone) {
      return res.send({ message: "Phone no is Required" });
    }
    if (!address) {
      return res.send({ message: "Address is Required" });
    }
    if (!answer) {
      return res.send({ message: "Answer is Required" });
    }
    //check user
    const exisitingUser = await userModel.findOne({ email });
    //exisiting user
    if (exisitingUser) {
      return res.status(200).send({
        success: false,
        message: "Already Register please login",
      });
    }
    //register user
    const hashedPassword = await hashPassword(password);
    //save
    const user = await new userModel({
      name,
      email,
      phone,
      address,
      password: hashedPassword,
      answer,
    }).save();

    res.status(201).send({
```

```
      success: true,
      message: "User Register Successfully",
      user,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Errro in Registeration",
      error,
    });
  }
};

//POST LOGIN
export const loginController = async (req, res) => {
  try {
    const { email, password } = req.body;
    //validation
    if (!email || !password) {
      return res.status(404).send({
        success: false,
        message: "Invalid email or password",
      });
    }
    //check user
    const user = await userModel.findOne({ email });
    if (!user) {
      return res.status(404).send({
        success: false,
        message: "Email is not registerd",
      });
    }
    const match = await comparePassword(password, user.password);
    if (!match) {
      return res.status(200).send({
        success: false,
        message: "Invalid Password",
      });
    }
    //token
    const token = await JWT.sign({ _id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "7d",
    });
    res.status(200).send({
```

```
      success: true,
      message: "login successfully",
      user: {
        _id: user._id,
        name: user.name,
        email: user.email,
        phone: user.phone,
        address: user.address,
        role: user.role,
      },
      token,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error in login",
      error,
    });
  }
};

//forgotPasswordController

export const forgotPasswordController = async (req, res) => {
  try {
    const { email, answer, newPassword } = req.body;
    if (!email) {
      res.status(400).send({ message: "Emai is required" });
    }
    if (!answer) {
      res.status(400).send({ message: "answer is required" });
    }
    if (!newPassword) {
      res.status(400).send({ message: "New Password is required" });
    }
    //check
    const user = await userModel.findOne({ email, answer });
    //validation
    if (!user) {
      return res.status(404).send({
        success: false,
        message: "Wrong Email Or Answer",
      });
    }
```

```
    const hashed = await hashPassword(newPassword);
    await userModel.findByIdAndUpdate(user._id, { password: hashed });
    res.status(200).send({
      success: true,
      message: "Password Reset Successfully",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Something went wrong",
      error,
    });
  }
};

//test controller
export const testController = (req, res) => {
  try {
    res.send("Protected Routes");
  } catch (error) {
    console.log(error);
    res.send({ error });
  }
};

//update prfole
export const updateProfileController = async (req, res) => {
  try {
    const { name, email, password, address, phone } = req.body;
    const user = await userModel.findById(req.user._id);
    //password
    if (password && password.length < 6) {
      return res.json({ error: "Passsword is required and 6 character long" });
    }
    const hashedPassword = password ? await hashPassword(password) : undefined;
    const updatedUser = await userModel.findByIdAndUpdate(
      req.user._id,
      {
        name: name || user.name,
        password: hashedPassword || user.password,
        phone: phone || user.phone,
        address: address || user.address,
      },
      { new: true }
```

```
    );
    res.status(200).send({
      success: true,
      message: "Profile Updated SUccessfully",
      updatedUser,
    });
  } catch (error) {
    console.log(error);
    res.status(400).send({
      success: false,
      message: "Error WHile Update profile",
      error,
    });
  }
};

//orders
export const getOrdersController = async (req, res) => {
  try {
    const orders = await orderModel
      .find({ buyer: req.user._id })
      .populate("products", "-photo")
      .populate("buyer", "name");
    res.json(orders);
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error WHile Geting Orders",
      error,
    });
  }
};
//orders
export const getAllOrdersController = async (req, res) => {
  try {
    const orders = await orderModel
      .find({})
      .populate("products", "-photo")
      .populate("buyer", "name")
      .sort({ createdAt: "-1" });
    res.json(orders);
  } catch (error) {
    console.log(error);
    res.status(500).send({
```

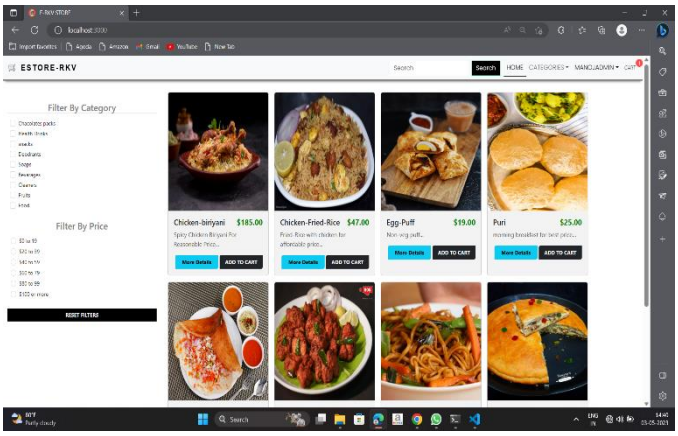
```
      success: false,
      message: "Error WHile Geting Orders",
      error,
    });
  }
};

//order status
export const orderStatusController = async (req, res) => {
  try {
    const { orderId } = req.params;
    const { status } = req.body;
    const orders = await orderModel.findByIdAndUpdate(
      orderId,
      { status },
      { new: true }
    );
    res.json(orders);
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Error While Updateing Order",
      error,
    });
  }
};
```

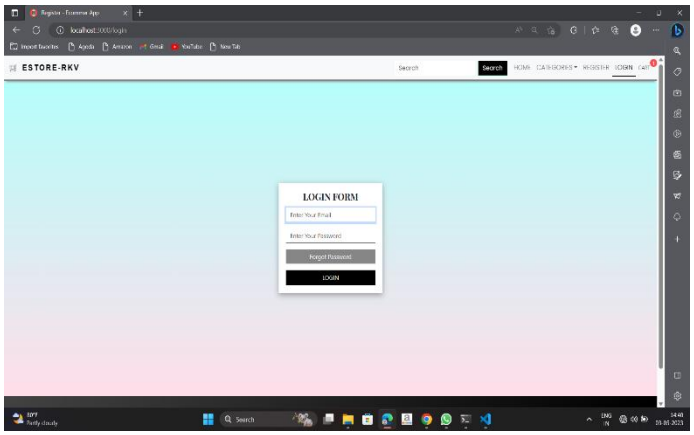
middleware.js:

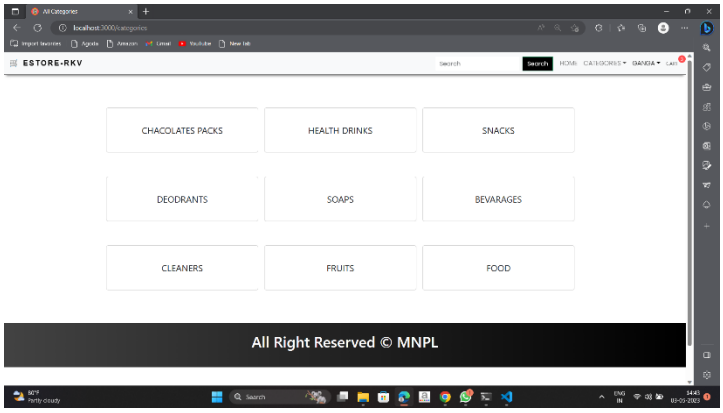
```
import JWT from "jsonwebtoken";
import userModel from "../models/userModel.js";

//Protected Routes token base
export const requireSignIn = async (req, res, next) => {
  try {
    const decode = JWT.verify(
      req.headers.authorization,
      process.env.JWT_SECRET
    );
    req.user = decode;
    next();
```

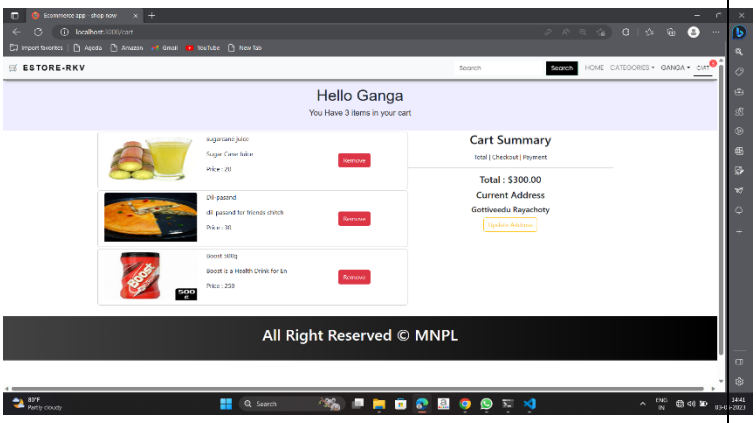
```
  } catch (error) {
    console.log(error);
  }
};

//admin acceess
export const isAdmin = async (req, res, next) => {
  try {
    const user = await userModel.findById(req.user._id);
    if (user.role !== 1) {
      return res.status(401).send({
        success: false,
        message: "UnAuthorized Access",
      });
    } else {
      next();
    }
  } catch (error) {
    console.log(error);
    res.status(401).send({
      success: false,
      error,
      message: "Error in admin middelware",
    });
  }
};
```

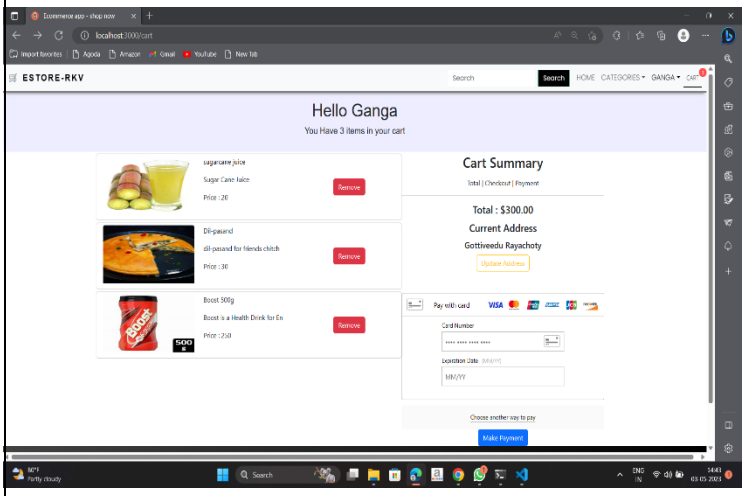
OUTPUT RESULTS:

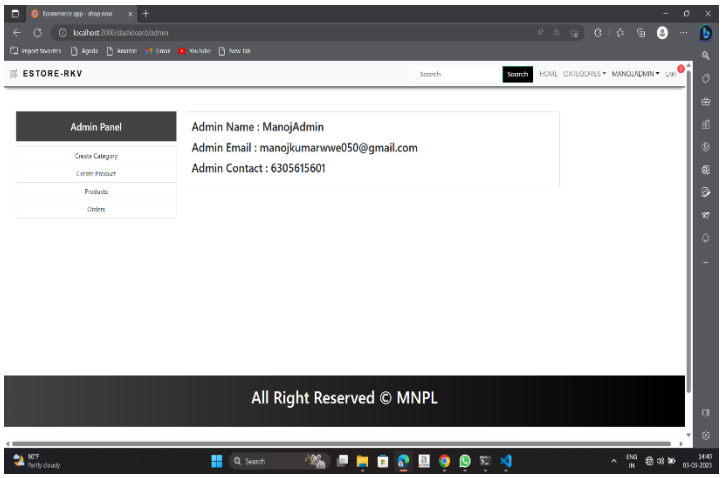












Conclusion:

This web Application provides Costumers to Purchase items through online without standing in the Queue for billing and improve Crowd management in the Store. It saves time and also makes the sellers life also easy and helps them to increase their sales. The E-store RKV e-commerce site need to be tested thoroughly tested before implementation to find any security gaps.

Future Enhancement:

Introduce fast delivery system

Introduce Coupan and voucher mechanism

Introduce OTP mechanism after order and delivery.

References:

https://www.w3schools.com
https://www.mongoodb.com/
https://react.dev/
https://www.javatpoint.com/
https://nodejs.org/en/docs