Index.js  
// Import required modules

const cors = require('cors');

const express = require('express');

const mongoose = require('mongoose');

const compression = require('compression');

const dotenv = require('dotenv');

const passport = require('passport');

const session = require('express-session');

const MongoStore = require('connect-mongo');

// Initialize express app

var app = express();

// Load env and set PORT

dotenv.config({ path: './config/config.env' });

const PORT = process.env.PORT || 4000;

// MongoDB connection

mongoose.connect(process.env.MONGO\_URI, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

})

.then(() => console.log("MongoDB connected"))

.catch(err => console.log("MongoDB connection error: ", err));

mongoose.connect(process.env.MONGO\_URI, { serverSelectionTimeoutMS: 30000 }); // 30 seconds

// Passport config

require('./config/passport')(passport);

app.use(cors({

    origin: process.env.FRONTEND || 'http://localhost:3000',

    credentials: true,

    exposedHeaders: ['set-cookie'],

    methods: ['GET', 'POST', 'PUT', 'DELETE'],

}));

app.use(express.json());

app.use(compression());

app.use(express.urlencoded({ extended: true }));

app.use((req, res, next) => {

    res.header('Access-Control-Allow-Credentials', 'true');

    res.header('Access-Control-Expose-Headers', 'Set-Cookie');

    next();

  });

// Session management

app.use(

    session({

        secret: process.env.SECRET || 'Mess\_Portal',

        resave: false,

        saveUninitialized: false,

        store: MongoStore.create({

            mongoUrl: process.env.MONGO\_URI,

            ttl: 1 \* 24 \* 60 \* 60, //

            autoRemove: 'interval',

            autoRemoveInterval: 10 // Minutes

        }),

        cookie: {

            maxAge: 1000 \* 60 \* 60 , // 1 day session duration

            secure: false, // Use `true` for HTTPS

            httpOnly: true,

            sameSite: 'lax',

            path: '/'

        }

    })

);

// Add passport session persistence middleware

app.use((req, res, next) => {

    if (req.session && !req.session.regenerate) {

        req.session.regenerate = true;

        req.session.save(err => {

            if (err) console.error('Session save error:', err);

            next();

        });

    } else {

        next();

    }

});

app.use(passport.initialize());

app.use(passport.session());

// Debugging middleware

// app.use((req, res, next) => {

//     console.log('Session ID:', req.sessionID);

//     console.log('Authenticated:', req.isAuthenticated());

//     console.log('User:', req.user);

//     next();

// });

//Requests

app.use((req, res, next) => {

    console.log(`Received request for ${req.url}`);

    next();

});

// Check authentication for admin endpoints

app.use('/api/admin/\*', (req, res, next) => {

    console.log("here",req);

    if (req.isAuthenticated() && req.user?.email === process.env.ADMIN) next();

    else res.sendStatus(401);

});

// Check authentication for user endpoints

app.use('/api/user/\*', (req, res, next) => {

    if (req.isAuthenticated()) next();

    else res.sendStatus(401);

});

// Routes

app.use('/api/auth', require('./routes/auth'));

app.use('/api/data', require('./routes/data'));

app.use('/api/admin', require('./routes/admin'));

app.use('/api/user', require('./routes/user'));

// Serve frontend

app.use(express.static(\_\_dirname + '/frontend/build'));

app.get('\*', (req, res) => res.sendFile(\_\_dirname + '/frontend/build/index.html'));

// Start server

app.listen(PORT, () => console.log(`Server started on port ${PORT}`));

admin.js

// Import required modules

const express = require("express");

const router = express.Router();

// Import database models

const Menu = require('../models/Menu');

const Time = require('../models/Time');

const Buyer = require('../models/Buyer');

// Set the time and cost of breakfast, lunch, dinner

router.post(

    "/setTime",

    async (req, res) => {

        await Time.setTimes(req.body.times);

        res.send();

    }

);

// Set the weekly menu

router.post(

    "/setMenu",

    async (req, res) => {

        await Menu.setMenus(req.body.menus);

        res.send();

    }

);

// Get the total meals that need to be cooked

router.post(

    "/meals",

    async (req, res) => {

        const buyers = await Buyer.allBuyers();

        let data = {

            monday: { breakfast: 0, lunch: 0, dinner: 0 },

            tuesday: { breakfast: 0, lunch: 0, dinner: 0 },

            wednesday: { breakfast: 0, lunch: 0, dinner: 0 },

            thursday: { breakfast: 0, lunch: 0, dinner: 0 },

            friday: { breakfast: 0, lunch: 0, dinner: 0 },

            saturday: { breakfast: 0, lunch: 0, dinner: 0 },

            sunday: { breakfast: 0, lunch: 0, dinner: 0 }

        }

        for (let buyer of buyers) {

            let meals = buyer[req.body.week];

            for (const [day, val] of Object.entries(meals)) {

                data[day]["breakfast"] += val.breakfast;

                data[day]["lunch"] += val.lunch;

                data[day]["dinner"] += val.dinner;

            }

        }

        const processed = [

            { day: "monday", breakfast: data.monday.breakfast, lunch: data.monday.lunch, dinner: data.monday.dinner },

            { day: "tuesday", breakfast: data.tuesday.breakfast, lunch: data.tuesday.lunch, dinner: data.tuesday.dinner },

            { day: "wednesday", breakfast: data.wednesday.breakfast, lunch: data.wednesday.lunch, dinner: data.wednesday.dinner },

            { day: "thursday", breakfast: data.thursday.breakfast, lunch: data.thursday.lunch, dinner: data.thursday.dinner },

            { day: "friday", breakfast: data.friday.breakfast, lunch: data.friday.lunch, dinner: data.friday.dinner },

            { day: "saturday", breakfast: data.saturday.breakfast, lunch: data.saturday.lunch, dinner: data.saturday.dinner },

            { day: "sunday", breakfast: data.sunday.breakfast, lunch: data.sunday.lunch, dinner: data.sunday.dinner }

        ]

        res.send(processed);

    }

);

module.exports = router;

auth.js

// Import required modules

const express = require("express");

const passport = require("passport");

const router = express.Router();

// Signin button link

router.get(

    "/signin",

    (req, res, next) => {

        console.log("Google sign-in route was called");

        console.log(req, res);

         // Debugging log

        next();  // Proceed with authentication

    },

    passport.authenticate(

        "google",

        {

            prompt: "select\_account",

            scope: ["profile", "email"]

        }

    )

);

router.get(

    "/signout",

    (req, res, next) => {

        console.log("Google sign-out route was called");  // Debugging log

        // Call req.logout and provide a callback function to handle errors

        req.logout(function (err) {

            if (err) {

                return next(err);  // Pass the error to the next middleware

            }

            res.redirect(process.env.FRONTEND);  // Redirect to frontend after successful logout

        });

    }

);

router.get('/google/callback', passport.authenticate('google', { failureRedirect: process.env.FRONTEND }),

    (req, res) => {

        // Call req.login() to establish a session

        req.login(req.user, (err) => {

            if (err) {

                console.error('Error during login:', err);

                return res.redirect(process.env.FRONTEND);

            }

            // Redirect after successful login

            res.redirect(process.env.FRONTEND);

        });

    }

);

module.exports = router;

data.js

// Import required modules

const express = require("express");

const router = express.Router();

// Import database models

const Menu = require('../models/Menu');

const Time = require('../models/Time');

// Get the weekly menu

router.get(

    "/menu",

    async (req, res) => {

        console.log("menu done");

        res.send(await Menu.getMenu());

    }

);

// Get the time and cost of breakfast, lunch, dinner

router.get(

    "/time",

    async (req, res) => {

        console.log("time done", Time.getTimes());

        res.send(await Time.getTimes());

    }

);

// Get the logged in and admin status

router.get(

    "/status",

    async (req, res) => {

        console.log("status done");

        console.log({loggedIn: req.isAuthenticated()});

        res.send({ loggedIn: req.isAuthenticated(), admin: (req.isAuthenticated() && req.user?.email === process.env.ADMIN) });

    }

);

module.exports = router;

user.js

// Import required modules

const express = require("express");

const router = express.Router();

const Razorpay = require("razorpay");

// Import database models

const Buyer = require("../models/Buyer");

const Time = require("../models/Time");

const Order = require("../models/Order");

// Import RazorPay payment validator

var {

  validatePaymentVerification,

} = require("razorpay/dist/utils/razorpay-utils");

// Get the user secret and meals purchased

router.get("/data", async (req, res) => {

  console.log("Buyer:",Buyer.getBuyer(req.user?.email));

  res.send(await Buyer.getBuyer(req.user?.email));

});

// Reset the user secret

router.get("/resetSecret", async (req, res) => {

  res.send(await Buyer.resetSecret(req.user?.email));

});

// Check if the user's coupon is valid for the current day and meal

router.post("/checkCoupon", async (req, res) => {

  res.send(await Buyer.checkCoupon(req.body));

});

// Check if the user has already bought coupons for the next week

router.get("/boughtNextWeek", async (req, res) => {

  console.log("Buyer boughtNextWeek:",Buyer.boughtNextWeek(req.user.email));

  res.send(await Buyer.boughtNextWeek(req.user?.email));

});

// Check if the payment throught RazorPay is successful

router.post("/checkOrder", async (req, res) => {

  const isValid = validatePaymentVerification(

    {

      order\_id: req.body.razorpay\_order\_id,

      payment\_id: req.body.razorpay\_payment\_id,

    },

    req.body.razorpay\_signature,

    process.env.PAY\_SECRET

  );

  if (isValid) {

    const orderObj = await Order.getOrder(req.body.razorpay\_order\_id);

    await Buyer.saveOrder(req.user?.email, orderObj.selected);

  }

  res.send(isValid);

});

// Example usage in route handler

router.post("/createOrder", async (req, res) => {

  try {

    console.log("/createOrder:", req);

    // Validate request

    if (!req.user || !req.body.selected) {

      return res.status(400).json({ error: "Invalid request" });

    }

    console.log("order:")

    console.log(req.body.selected)

    // Create order

    const order = await Order.create({

      user: req.user.\_id,

      selected: req.body.selected,

      status: "completed",

    });

    // Update buyer

    await Buyer.saveOrder(req.user.email, order.selected);

    res.status(201).json({

      message: "Order created successfully",

      orderId: order.\_id,

    });

  } catch (error) {

    console.error("Order creation error:", error);

    res.status(500).json({

      error: "Failed to create order",

      details:

        process.env.NODE\_ENV === "development" ? error.message : undefined,

    });

  }

});

module.exports = router;

models:

buyer.js

const mongoose = require("mongoose");

const BuyerSchema = mongoose.model("buyer", new mongoose.Schema({

    email: String,

    secret: String,

    bought: Boolean,

    this: {

        monday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        tuesday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        wednesday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        thursday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        friday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        saturday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        sunday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        }

    },

    next: {

        monday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        tuesday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        wednesday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        thursday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        friday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        saturday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        },

        sunday: {

            breakfast: { type: Boolean, default: false },

            lunch: { type: Boolean, default: false },

            dinner: { type: Boolean, default: false }

        }

    }

}));

// Get the user details, or if it doesn't exists, create a new user object

module.exports.getBuyer = async function (email) {

    let charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789";

    let randomStr = "";

    for (let i = 0; i < 4; i++)

        randomStr += charset[Math.floor(Math.random() \* charset.length)];

    const Buyer = await BuyerSchema.findOneAndUpdate(

        { email: email },

        {

            $setOnInsert: {

                bought: false,

                secret: randomStr,

                this: {

                    monday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    tuesday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    wednesday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    thursday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    friday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    saturday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    sunday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    }

                },

                next: {

                    monday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    tuesday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    wednesday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    thursday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    friday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    saturday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    },

                    sunday: {

                        breakfast: false,

                        lunch: false,

                        dinner: false

                    }

                }

            }

        },

        { new: true, upsert: true }

    ).select({ \_id: 0 });

    return Buyer;

}

// Resets the user secret and returns the updated user object

module.exports.resetSecret = async function (email) {

    let charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789";

    let randomStr = "";

    for (let i = 0; i < 4; i++)

        randomStr += charset[Math.floor(Math.random() \* charset.length)];

    const Buyer = await BuyerSchema.findOneAndUpdate(

        { email: email },

        { secret: randomStr }).select({ \_id: 0 });

    return Buyer;

}

// Check if the user's coupon is valid for the current day and meal

module.exports.checkCoupon = async function (data) {

    console.log("Coupon is here!!", data.email);

    const Buyer = await BuyerSchema.findOne({ email: data.email, secret: data.secret });

    console.log("Coupon is here Buyer!!", Buyer);

    console.log("Coupon is here Buyer!! this:", Buyer.this[data.day]);

    if (Buyer == null) return false;

    if (Buyer.this[data.day][data.type]) {

        await BuyerSchema.updateOne({ email: data.email }, { ["this." + data.day + "." + data.type]: false });

        console.log("true")

        return true;

    }

    console.log("false")

    return false;

}

// Save the purchased coupons after a successful payment

module.exports.saveOrder = async function (email, selectedMeals) {

    console.log("SelectedMeals:", selectedMeals)

    await BuyerSchema.updateOne(

        { email: email },

        {

            $set: {

                "this": selectedMeals,

                bought: true

            }

        },

        { upsert: true }

    );

}

// Check if the user has already bought the coupons for the coming week

module.exports.boughtNextWeek = async function (email) {

    await module.exports.getBuyer(email);

    const Buyer = await BuyerSchema.findOne({ email: email });

    return Buyer.bought;

}

// Returns details of all the users

module.exports.allBuyers = async function () {

    const Buyers = await BuyerSchema.find({});

    return Buyers;

}

Menu.js

const mongoose = require("mongoose");

const MenuSchema = mongoose.model("menuitem", new mongoose.Schema({

    day: String,

    breakfast: String,

    lunch: String,

    dinner: String

}));

// Get the weekly menu

// module.exports.getMenu = async function () {

//     const menuItems = await MenuSchema.find({})

//         .select({ \_id: 0 });

//     return menuItems;

// }

module.exports.getMenu = async function () {

    const menuItems = await MenuSchema.find({})

        .select({ \_id: 0 });

    const dayOrder = ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday"];

    // Sort manually by day name

    menuItems.sort((a, b) => {

        return dayOrder.indexOf(a.day.toLowerCase()) - dayOrder.indexOf(b.day.toLowerCase());

    });

    return menuItems;

}

// Set the weekly menu

module.exports.setMenus = async function (menus) {

    await MenuSchema.deleteMany({});

    await MenuSchema.insertMany(menus);

}

Order.js

const mongoose = require("mongoose");

const orderSchema = new mongoose.Schema({

    user: {

        type: mongoose.Schema.Types.ObjectId,

        ref: 'User',

        required: true

    },

    selected: {

        type: Object,

        required: true

    },

    status: {

        type: String,

        enum: ['pending', 'completed'],

        default: 'completed' // Default to completed since we're skipping payment

    },

    createdAt: {

        type: Date,

        default: Date.now

    }

});

module.exports = mongoose.model('Order', orderSchema);

// Save an order in progress throught RazorPay

module.exports.saveOrder = async function (orderid, selected) {

  await orderSchema.create({ orderid: orderid, selected: selected });

};

// Get a saved order to update the user after successful payment

module.exports.getOrder = async function (orderid) {

  const orderObj = await OrderSchema.findOne({ orderid: orderid });

  return orderObj;

};

Time.js

const mongoose = require("mongoose");

const TimeSchema = mongoose.model("time", new mongoose.Schema({

    meal: String,

    time: String,

    cost: Number

}));

// Get the cost and time of breakfast, lunch, dinner

module.exports.getTimes = async function () {

    const Times = await TimeSchema.find({})

        .select({ \_id: 0 });

    return Times;

}

// Set the cost and time of breakfast, lunch, dinner

module.exports.setTimes = async function (times) {

    await TimeSchema.deleteMany({});

    await TimeSchema.insertMany(times);

}

User.js

const mongoose = require('mongoose')

const UserSchema = new mongoose.Schema({

    googleId: {

        type: String,

        required: true,

    },

    displayName: {

        type: String,

        required: true,

    },

    email: {

        type: String,

        required: true,

    }

})

// To be used by Passport to manage the google signins

module.exports = mongoose.model('User', UserSchema)

frontend:

index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import 'antd/dist/reset.css'; // For newer Ant Design versions

import App from './App';

import { BrowserRouter } from 'react-router-dom';

import '@ant-design/v5-patch-for-react-19';

import axios from 'axios';

window.APIROOT = 'http://localhost:4000/';

// // Create an Axios instance

const api = axios.create({

  baseURL: 'http://localhost:4000/',

  withCredentials: true,

  headers: {

    'Content-Type': 'application/json',

    'X-Requested-With': 'XMLHttpRequest'

  }

});

// Add request interceptor

api.interceptors.request.use(config => {

  config.headers['Cache-Control'] = 'no-cache';

  config.headers['Pragma'] = 'no-cache';

  return config;

});

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <BrowserRouter>

    <React.StrictMode>

      <App />

    </React.StrictMode>

  </BrowserRouter>

);

export default api;

app.js

import { useState, useEffect } from 'react';

import { Route, Routes, Navigate, useNavigate, useLocation } from 'react-router-dom';

import { Layout } from 'antd';

// Utilities

import AnimationLayout from './utility/routeAnimation';

import RemoveLoader from './utility/removeLoader';

// Components

import MenuBar from './components/MenuBar';

// Route Pages

import HomePage from './routes/Home';

import QRCodePage from './routes/QRCode';

import SchedulePage from './routes/Schedule';

import PurchaseHistoryPage from './routes/PurchaseHistory';

import BuyPage from './routes/Buy';

import AdminPanel from './routes/AdminPanel';

import TotalMealsPage from './routes/TotalMeals';

import ScanQRPage from './routes/ScanQR';

// To check if user is logged in

import api from '.';

import UPIPayment from './components/UPI';

export default function App() {

    // Removes the loader after the site has been fully loaded

    useEffect(RemoveLoader, []);

    // State to track authentication across the app

    const [isAuthenticated, setIsAuthenticated] = useState(false);

    const [loading, setLoading] = useState(true);

    const [justLoggedIn, setJustLoggedIn] = useState(false);

    const navigate = useNavigate();

    const location = useLocation();

    useEffect(() => {

        const checkAuth = async () => {

            try {

                const response = await api.get('api/data/status');

                const newAuthStatus = response.data?.loggedIn || false;

                // If this is a change from not authenticated to authenticated

                if (!isAuthenticated && newAuthStatus) {

                    setJustLoggedIn(true);

                }

                setIsAuthenticated(newAuthStatus);

            } catch (error) {

                setIsAuthenticated(false);

            } finally {

                setLoading(false);

            }

        };

        checkAuth();

    }, [isAuthenticated]);

    // Handle redirect after login

    useEffect(() => {

        console.log("Herre"+window.location.href);

        if (justLoggedIn && isAuthenticated) {

            if(!(window.location.href === "http://localhost:3000/upi-payment")){

                navigate('/schedule');

            }

            setJustLoggedIn(false); // Reset the flag after redirect

        }

    }, [justLoggedIn, isAuthenticated, navigate]);

    // Protected route component

    const ProtectedRoute = ({ children }) => {

        if (loading) {

            return <div>Loading...</div>;

        }

        return isAuthenticated ? children : <Navigate to="/" />;

    };

    return (

        <div className="App">

            <Layout style={{ minHeight: "100vh" }}>

                <MenuBar />

                <Routes>

                    <Route element={<AnimationLayout />}>

                        {/\* Public routes \*/}

                        <Route path="/" element={<HomePage />} />

                        <Route path="/schedule" element={<SchedulePage />} />

                        {/\* Protected routes \*/}

                        <Route path="/qr-code" element={

                            <ProtectedRoute>

                                <QRCodePage />

                            </ProtectedRoute>

                        } />

                        <Route path="/purchase-history" element={

                            <ProtectedRoute>

                                <PurchaseHistoryPage />

                            </ProtectedRoute>

                        } />

                        <Route path="/buy-coupons" element={

                            <ProtectedRoute>

                                <BuyPage />

                            </ProtectedRoute>

                        } />

                        <Route path="/upi-payment" element={

                            <ProtectedRoute>

                                <UPIPayment />

                            </ProtectedRoute>

                        } />

                        {/\* Admin routes (these could have additional admin-specific protection) \*/}

                        <Route path="/admin" element={

                            <ProtectedRoute>

                                <AdminPanel />

                            </ProtectedRoute>

                        } />

                        <Route path="/total-meals" element={

                            <ProtectedRoute>

                                <TotalMealsPage />

                            </ProtectedRoute>

                        } />

                        <Route path="/scan-qr" element={

                            <ProtectedRoute>

                                <ScanQRPage />

                            </ProtectedRoute>

                        } />

                    </Route>

                </Routes>

            </Layout>

        </div>

    );

}

Adminpanel/index.js

import { useState, useEffect } from "react";

import {

  Button,

  Input,

  Card,

  Table,

  message,

  Tabs,

  Tooltip,

  Popconfirm,

  Spin,

  Typography,

  Space,

  Badge

} from 'antd';

import {

  SaveOutlined,

  EditOutlined,

  CheckOutlined,

  CloseOutlined,

  DollarOutlined,

  ClockCircleOutlined,

  MenuOutlined,

  ExclamationCircleOutlined

} from '@ant-design/icons';

import { motion } from "framer-motion";

import { useMediaQuery } from 'react-responsive';

import api from '../..';

import styles from './index.module.css';

const { Title, Text } = Typography;

const { TabPane } = Tabs;

export default function AdminPanel() {

  const isMobile = useMediaQuery({ query: '(max-width: 750px)' });

  // ====== TIMING AND COST SECTION ======

  const [timingData, setTimingData] = useState([]);

  const [savingTime, setSavingTime] = useState(false);

  const [loadingTime, setLoadingTime] = useState(true);

  const [editingTimeKey, setEditingTimeKey] = useState(null);

  const [editTimeValues, setEditTimeValues] = useState({});

  // ====== MENU SECTION ======

  const [menuData, setMenuData] = useState([]);

  const [savingMenu, setSavingMenu] = useState(false);

  const [loadingMenu, setLoadingMenu] = useState(true);

  const [editingCell, setEditingCell] = useState({ day: null, meal: null });

  // Add new state for menu edit value

  const [editMenuValue, setEditMenuValue] = useState('');

  // Define the correct day order

  const dayOrder = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday'];

  // Handle editing for Time & Cost section

  const startEditingTime = (record) => {

    setEditingTimeKey(record.key);

    setEditTimeValues({

      cost: record.cost,

      time: record.time

    });

  };

  const saveTimeEdit = async (record) => {

    const newData = [...timingData];

    const index = newData.findIndex(item => item.key === record.key);

    if (index > -1) {

      newData[index] = {

        ...newData[index],

        cost: editTimeValues.cost || newData[index].cost,

        time: editTimeValues.time || newData[index].time

      };

      setTimingData(newData);

      setEditingTimeKey(null);

      try {

        setSavingTime(true);

        await api.post('api/admin/setTime', { times: newData });

        message.success('Changes saved successfully');

      } catch (error) {

        message.error('Failed to save changes');

        // Rollback if saving fails

        fetchTime();

      } finally {

        setSavingTime(false);

      }

    }

  };

  const cancelTimeEdit = () => {

    setEditingTimeKey(null);

  };

  // Handle editing for Menu section

  const startEditingMenu = (day, mealType) => {

    const dayData = menuData.find(item => item.day === day);

    setEditingCell({ day, meal: mealType });

    setEditMenuValue(dayData[mealType]);

  };

  const saveMenuEdit = async (day, mealType, value) => {

    const newMenuData = [...menuData];

    const dayIndex = newMenuData.findIndex(item => item.day === day);

    if (dayIndex > -1) {

      newMenuData[dayIndex] = {

        ...newMenuData[dayIndex],

        [mealType]: value

      };

      setMenuData(newMenuData);

      setEditingCell({ day: null, meal: null });

      try {

        setSavingMenu(true);

        await api.post('api/admin/setMenu', { menus: newMenuData });

        message.success(`${mealType.charAt(0).toUpperCase() + mealType.slice(1)} for ${day} updated`);

      } catch (error) {

        message.error('Failed to save changes');

        // Rollback if saving fails

        fetchMenu();

      } finally {

        setSavingMenu(false);

      }

    }

  };

  const cancelMenuEdit = () => {

    setEditingCell({ day: null, meal: null });

  };

  // Fetch Time & Cost data

  const fetchTime = async () => {

    setLoadingTime(true);

    try {

      let response = await api.get('api/data/time');

      // Add key property for antd Table

      const dataWithKeys = response.data.map((item, index) => ({

        ...item,

        key: index.toString()

      }));

      setTimingData(dataWithKeys);

    } catch (error) {

      message.error('Failed to fetch timing data');

    } finally {

      setLoadingTime(false);

    }

  };

  // Fetch Menu data

  const fetchMenu = async () => {

    setLoadingMenu(true);

    try {

      const response = await api.get('api/data/menu');

      setMenuData(response.data);

    } catch (error) {

      message.error('Failed to fetch menu data');

    } finally {

      setLoadingMenu(false);

    }

  };

  // Initialize data on component mount

  useEffect(() => {

    fetchTime();

    fetchMenu();

  }, []);

  // Save all Time & Cost changes

  const saveAllTimeChanges = async () => {

    setSavingTime(true);

    try {

      await api.post('api/admin/setTime', { times: timingData });

      message.success('All changes saved successfully');

    } catch (error) {

      message.error('Failed to save changes');

    } finally {

      setSavingTime(false);

    }

  };

  // Save all Menu changes

  const saveAllMenuChanges = async () => {

    setSavingMenu(true);

    try {

      await api.post('api/admin/setMenu', { menus: menuData });

      message.success('All menu changes saved successfully');

    } catch (error) {

      message.error('Failed to save changes');

    } finally {

      setSavingMenu(false);

    }

  };

  // Time & Cost columns configuration

  const timingColumns = [

    {

      title: 'Meal',

      dataIndex: 'meal',

      key: 'meal',

      render: (text) => (

        <Space>

          {text === 'Breakfast' && <ClockCircleOutlined style={{ color: '#1890ff' }} />}

          {text === 'Lunch' && <ClockCircleOutlined style={{ color: '#52c41a' }} />}

          {text === 'Dinner' && <ClockCircleOutlined style={{ color: '#722ed1' }} />}

          <Text strong>{text}</Text>

        </Space>

      )

    },

    {

      title: () => (

        <Space>

          <DollarOutlined />

          <span>Price (Rs)</span>

        </Space>

      ),

      dataIndex: 'cost',

      key: 'cost',

      width: 150,

      render: (text, record) => {

        const editing = record.key === editingTimeKey;

        return editing ? (

          <Input

            value={editTimeValues.cost}

            onChange={(e) => setEditTimeValues({ ...editTimeValues, cost: e.target.value })}

            onPressEnter={() => saveTimeEdit(record)}

            suffix="Rs"

          />

        ) : (

          <Space>

            <Text>{text}</Text>

            <Text type="secondary">Rs</Text>

          </Space>

        );

      }

    },

    {

      title: () => (

        <Space>

          <ClockCircleOutlined />

          <span>Serving Time</span>

        </Space>

      ),

      dataIndex: 'time',

      key: 'time',

      width: 200,

      render: (text, record) => {

        const editing = record.key === editingTimeKey;

        return editing ? (

          <Input

            value={editTimeValues.time}

            onChange={(e) => setEditTimeValues({ ...editTimeValues, time: e.target.value })}

            onPressEnter={() => saveTimeEdit(record)}

          />

        ) : (

          <Text>{text}</Text>

        );

      }

    },

    {

      title: 'Actions',

      key: 'action',

      width: 120,

      render: (\_, record) => {

        const editing = record.key === editingTimeKey;

        return editing ? (

          <Space>

            <Button

              type="primary"

              icon={<CheckOutlined />}

              size="small"

              onClick={() => saveTimeEdit(record)}

            />

            <Button

              danger

              icon={<CloseOutlined />}

              size="small"

              onClick={cancelTimeEdit}

            />

          </Space>

        ) : (

          <Button

            type="link"

            icon={<EditOutlined />}

            onClick={() => startEditingTime(record)}

          >

            Edit

          </Button>

        );

      }

    }

  ];

  // Render meal menu for a specific day

  const renderDayMenu = (day) => {

    const dayData = menuData.find(item => item.day === day);

    if (!dayData) return null;

    const mealTypes = ['breakfast', 'lunch', 'dinner'];

    return (

      <Card

        title={day.charAt(0).toUpperCase() + day.slice(1)}

        className={styles.dayCard}

        size="small"

        bordered

      >

        {mealTypes.map(mealType => {

          const isEditing = editingCell.day === day && editingCell.meal === mealType;

          const mealValue = dayData[mealType];

          return (

            <div key={mealType} className={styles.mealRow}>

              <div className={styles.mealType}>

                {mealType === 'breakfast' && <Badge color="#1890ff" />}

                {mealType === 'lunch' && <Badge color="#52c41a" />}

                {mealType === 'dinner' && <Badge color="#722ed1" />}

                <Text strong style={{ textTransform: 'capitalize' }}>{mealType}</Text>

              </div>

              <div className={styles.mealContent}>

                {isEditing ? (

                  <div className={styles.editActions}>

                    <Input

                      autoFocus

                      value={editMenuValue}

                      onChange={(e) => setEditMenuValue(e.target.value)}

                      onPressEnter={() => saveMenuEdit(day, mealType, editMenuValue)}

                      className={styles.mealInput}

                    />

                    <Space>

                      <Button

                        type="primary"

                        size="small"

                        icon={<CheckOutlined />}

                        onClick={() => saveMenuEdit(day, mealType, editMenuValue)}

                      />

                      <Button

                        danger

                        size="small"

                        icon={<CloseOutlined />}

                        onClick={cancelMenuEdit}

                      />

                    </Space>

                  </div>

                ) : (

                  <div className={styles.viewMode}>

                    <Text>{mealValue}</Text>

                    <Button

                      type="link"

                      icon={<EditOutlined />}

                      onClick={() => startEditingMenu(day, mealType)}

                      size="small"

                    />

                  </div>

                )}

              </div>

            </div>

          );

        })}

      </Card>

    );

  };

  // Sort the menuData according to the day order

  const getSortedMenuData = () => {

    if (!menuData.length) return [];

    // Create a copy of the menu data for sorting

    const sortedData = [...menuData];

    // Sort based on the day order array

    sortedData.sort((a, b) => {

      return dayOrder.indexOf(a.day.toLowerCase()) - dayOrder.indexOf(b.day.toLowerCase());

    });

    return sortedData;

  };

  return (

    <div className={styles.adminContainer}>

      <Tabs defaultActiveKey="menu" type="card">

        <TabPane

          tab={

            <span>

              <MenuOutlined /> Menu Management

            </span>

          }

          key="menu"

        >

          <motion.div

            layout

            className={styles.adminSection}

            initial={{ opacity: 0 }}

            animate={{ opacity: 1 }}

            transition={{ duration: 0.3 }}

          >

            <div className={styles.sectionHeader}>

              <Title level={4}>Weekly Menu</Title>

              <Tooltip title="Save all menu changes">

                <Button

                  type="primary"

                  icon={<SaveOutlined />}

                  loading={savingMenu}

                  onClick={saveAllMenuChanges}

                >

                  Save All Changes

                </Button>

              </Tooltip>

            </div>

            {loadingMenu ? (

              <div className={styles.loadingContainer}>

                <Spin size="large" />

              </div>

            ) : (

              <div className={styles.menuGrid}>

                {getSortedMenuData().map(item => renderDayMenu(item.day))}

              </div>

            )}

          </motion.div>

        </TabPane>

        <TabPane

          tab={

            <span>

              <DollarOutlined /> Pricing & Timing

            </span>

          }

          key="timing"

        >

          <motion.div

            layout

            className={styles.adminSection}

            initial={{ opacity: 0 }}

            animate={{ opacity: 1 }}

            transition={{ duration: 0.3 }}

          >

            <div className={styles.sectionHeader}>

              <Title level={4}>Meal Pricing & Serving Times</Title>

              <Tooltip title="Save all time and price changes">

                <Button

                  type="primary"

                  icon={<SaveOutlined />}

                  loading={savingTime}

                  onClick={saveAllTimeChanges}

                >

                  Save All Changes

                </Button>

              </Tooltip>

            </div>

            {loadingTime ? (

              <div className={styles.loadingContainer}>

                <Spin size="large" />

              </div>

            ) : (

              <Table

                columns={timingColumns}

                dataSource={timingData}

                pagination={false}

                rowClassName={styles.tableRow}

                bordered

              />

            )}

          </motion.div>

        </TabPane>

      </Tabs>

    </div>

  );

}

Buy/index.js

import { useState, useEffect } from 'react';

import { Button, message, Card, Checkbox, Badge, Spin, Typography, Tabs, Tooltip, Empty } from 'antd';

import { ShoppingCartOutlined, CheckCircleOutlined, InfoCircleOutlined, CalendarOutlined, LockOutlined } from '@ant-design/icons';

import { motion } from 'framer-motion';

import { useMediaQuery } from 'react-responsive';

import api from '../..';

import classes from './index.module.css';

import UPIPayment from '../../components/UPI';

const { Title, Text } = Typography;

const { TabPane } = Tabs;

const dayToNum = {

    "monday": 0, "tuesday": 1, "wednesday": 2,

    "thursday": 3, "friday": 4, "saturday": 5, "sunday": 6

};

// Helper function to capitalize first letter

const capitalize = (str) => str ? str.charAt(0).toUpperCase() + str.slice(1) : '';

// Custom icon for meal types

const MealIcon = ({ type }) => {

    const icons = {

        breakfast: '🍳',

        lunch: '🍲',

        dinner: '🍽️'

    };

    return <span className={classes.mealIcon}>{icons[type]}</span>;

};

export async function createOrder(selected) {

    try {

        const response = await api.post('api/user/createOrder', { selected });

        return response.data;

    } catch (error) {

        message.error('Failed to create order');

        throw error;

    }

}

function MealCard({ dayData, selected, setSelected, loading, alreadyPurchased }) {

    // Add safety check to prevent rendering errors

    if (!dayData || !dayData.day || !selected[dayData.day]) {

        return null;

    }

    const handleMealSelection = (meal, checked) => {

        // If meal is already purchased, prevent selection

        if (alreadyPurchased[dayData.day]?.[meal]) {

            return;

        }

        setSelected(prev => ({

            ...prev,

            [dayData.day]: {

                ...prev[dayData.day],

                [meal]: checked

            }

        }));

    };

    const getMealTime = (meal) => {

        const times = {

            breakfast: '08:00 AM',

            lunch: '01:00 PM',

            dinner: '08:00 PM'

        };

        return times[meal];

    };

    return (

        <Card

            className={classes.dayCard}

            bordered={true}

            title={

                <div className={classes.dayHeader}>

                    <CalendarOutlined className={classes.calendarIcon} />

                    <span>{capitalize(dayData.day)}</span>

                </div>

            }

            headStyle={{ backgroundColor: '#f9f9f9' }}

            bodyStyle={{ padding: '12px' }}

        >

            <Spin spinning={loading}>

                {['breakfast', 'lunch', 'dinner'].map(meal => {

                    const isPurchased = alreadyPurchased[dayData.day]?.[meal];

                    return (

                    <motion.div

                        key={meal}

                        initial={{ opacity: 0 }}

                        animate={{ opacity: 1 }}

                        transition={{ duration: 0.3, delay: ['breakfast', 'lunch', 'dinner'].indexOf(meal) \* 0.1 }}

                    >

                        <Card

                            className={`${classes.mealItem}

                                      ${selected[dayData.day][meal] ? classes.selectedMeal : ''}

                                      ${isPurchased ? classes.purchasedMeal : ''}`}

                            size="small"

                            hoverable={!isPurchased}

                            onClick={() => handleMealSelection(meal, !selected[dayData.day][meal])}

                        >

                            <div className={classes.mealHeader}>

                                <div className={classes.mealTitleBlock}>

                                    <MealIcon type={meal} />

                                    <Text strong className={classes.mealType}>{capitalize(meal)}</Text>

                                    <Text type="secondary" className={classes.mealTime}>

                                        {getMealTime(meal)}

                                    </Text>

                                    {isPurchased && (

                                        <Badge

                                            count={<LockOutlined style={{ color: '#fff' }} />}

                                            style={{ backgroundColor: '#52c41a' }}

                                            className={classes.purchasedBadge}

                                        />

                                    )}

                                </div>

                                <Badge

                                    count={`₹${dayData[meal]?.cost || 0}`}

                                    className={classes.priceBadge}

                                    style={{ backgroundColor: isPurchased ? '#7cb305' : '#ff7f50' }}

                                />

                            </div>

                            <div className={classes.menuText}>

                                {dayData[meal]?.text || <Empty image={Empty.PRESENTED\_IMAGE\_SIMPLE} description="Menu not available" />}

                            </div>

                            <div className={classes.mealFooter}>

                                {isPurchased ? (

                                    <div className={classes.alreadyPurchased}>

                                        <CheckCircleOutlined style={{ color: '#52c41a' }} />

                                        <Text className={classes.purchasedText}>Already Purchased </Text>

                                    </div>

                                ) : (

                                    <Checkbox

                                        checked={selected[dayData.day][meal]}

                                        onChange={(e) => e.stopPropagation()}

                                        onClick={(e) => {

                                            e.stopPropagation();

                                            handleMealSelection(meal, !selected[dayData.day][meal]);

                                        }}

                                        className={classes.mealCheckbox}

                                        disabled={isPurchased}

                                    >

                                        Select

                                    </Checkbox>

                                )}

                                <Tooltip title={isPurchased ? "Already purchased" : "Click to select this meal"}>

                                    {isPurchased ?

                                        <CheckCircleOutlined className={classes.infoIcon} style={{ color: '#52c41a' }} /> :

                                        <InfoCircleOutlined className={classes.infoIcon} />

                                    }

                                </Tooltip>

                            </div>

                        </Card>

                    </motion.div>

                    );

                })}

            </Spin>

        </Card>

    );

    // return (

    //     <Card

    //         className={classes.dayCard}

    //         bordered={true}

    //         title={

    //             <div className={classes.dayHeader}>

    //                 <CalendarOutlined className={classes.calendarIcon} />

    //                 <span>{capitalize(dayData.day)}</span>

    //             </div>

    //         }

    //         headStyle={{ backgroundColor: '#f9f9f9' }}

    //         bodyStyle={{ padding: '12px' }}

    //     >

    //         <Spin spinning={loading}>

    //             {['breakfast', 'lunch', 'dinner'].map(meal => {

    //                 const isPurchased = alreadyPurchased[dayData.day]?.[meal];

    //                 // Hide the meal card if purchased

    //                 if (isPurchased) {

    //                     return null;

    //                 }

    //                 return (

    //                     <motion.div

    //                         key={meal}

    //                         initial={{ opacity: 0 }}

    //                         animate={{ opacity: 1 }}

    //                         transition={{ duration: 0.3, delay: ['breakfast', 'lunch', 'dinner'].indexOf(meal) \* 0.1 }}

    //                     >

    //                         <Card

    //                             className={`${classes.mealItem} ${selected[dayData.day][meal] ? classes.selectedMeal : ''}`}

    //                             size="small"

    //                             hoverable

    //                             onClick={() => handleMealSelection(meal, !selected[dayData.day][meal])}

    //                         >

    //                             <div className={classes.mealHeader}>

    //                                 <div className={classes.mealTitleBlock}>

    //                                     <MealIcon type={meal} />

    //                                     <Text strong className={classes.mealType}>{capitalize(meal)}</Text>

    //                                     <Text type="secondary" className={classes.mealTime}>

    //                                         {getMealTime(meal)}

    //                                     </Text>

    //                                 </div>

    //                                 <Badge

    //                                     count={`₹${dayData[meal]?.cost || 0}`}

    //                                     className={classes.priceBadge}

    //                                     style={{ backgroundColor: '#ff7f50' }}

    //                                 />

    //                             </div>

    //                             <div className={classes.menuText}>

    //                                 {dayData[meal]?.text || <Empty image={Empty.PRESENTED\_IMAGE\_SIMPLE} description="Menu not available" />}

    //                             </div>

    //                             <div className={classes.mealFooter}>

    //                                 <Checkbox

    //                                     checked={selected[dayData.day][meal]}

    //                                     onChange={(e) => e.stopPropagation()}

    //                                     onClick={(e) => {

    //                                         e.stopPropagation();

    //                                         handleMealSelection(meal, !selected[dayData.day][meal]);

    //                                     }}

    //                                     className={classes.mealCheckbox}

    //                                 >

    //                                     Select

    //                                 </Checkbox>

    //                                 <Tooltip title="Click to select this meal">

    //                                     <InfoCircleOutlined className={classes.infoIcon} />

    //                                 </Tooltip>

    //                             </div>

    //                         </Card>

    //                     </motion.div>

    //                 );

    //             })}

    //         </Spin>

    //     </Card>

    // );

}

function OrderSummary({ selected, loading, cost, setBought, menu, alreadyPurchased }) {

    const totalMeals = Object.values(selected).reduce((acc, dayMeals) =>

        acc + Object.values(dayMeals).filter(Boolean).length, 0);

    const selectedMealsList = [];

    Object.entries(selected).forEach(([day, meals]) => {

        Object.entries(meals).forEach(([meal, isSelected]) => {

            if (isSelected && dayToNum[day] !== undefined && menu[dayToNum[day]] && menu[dayToNum[day]][meal]) {

                selectedMealsList.push({

                    day: capitalize(day),

                    meal: capitalize(meal),

                    cost: menu[dayToNum[day]][meal].cost || 0

                });

            }

        });

    });

    // const handlePayment = async () => {

    //     try {

    //         // Create combinedSelected by merging alreadyPurchased and selected

    //         const combinedSelected = {};

    //         Object.keys(selected).forEach(day => {

    //             combinedSelected[day] = {};

    //             ['breakfast', 'lunch', 'dinner'].forEach(meal => {

    //                 // Include meal if it's already purchased or newly selected

    //                 combinedSelected[day][meal] = alreadyPurchased[day]?.[meal] || selected[day][meal];

    //             });

    //         });

    //         <UPIPayment/>

    //         await createOrder(combinedSelected);

    //         message.success("Order created successfully!");

    //         setBought(true);

    //         window.location.href = '/buy-coupons';

    //         // window.location.replace('/buy-coupons')

    //     } catch (error) {

    //         message.error("Failed to create order");

    //     }

    // };

    const [TotalLocal, setTotalLocal] = useState();

    const handlePayment = async () => {

        try {

            // Create combinedSelected by merging alreadyPurchased and selected

            const combinedSelected = {};

            Object.keys(selected).forEach(day => {

                combinedSelected[day] = {};

                ['breakfast', 'lunch', 'dinner'].forEach(meal => {

                    // Include meal if it's already purchased or newly selected

                    combinedSelected[day][meal] = alreadyPurchased[day]?.[meal] || selected[day][meal];

                });

            });

            // Store the combined selection in localStorage

            localStorage.setItem('pendingOrder', JSON.stringify(combinedSelected));

            localStorage.setItem('cost', JSON.stringify(TotalLocal));

            // Navigate to the UPI payment page

            window.location.href = '/upi-payment';

        } catch (error) {

            message.error("Failed to initiate payment");

        }

    };

    useEffect(() => {

        setTotalLocal(cost);

    }, [cost]);

    return (

        <Card className={classes.summaryCard}>

            <div className={classes.summaryHeader}>

                <Title level={4}>Order Summary</Title>

                <Badge

                    count={totalMeals}

                    showZero

                    overflowCount={99}

                    style={{ backgroundColor: totalMeals ? '#52c41a' : '#d9d9d9' }}

                >

                    <Text>Selected Meals</Text>

                </Badge>

            </div>

            <div className={classes.mealSummaryList}>

                {selectedMealsList.length > 0 ? (

                    selectedMealsList.map((item, index) => (

                        <div key={index} className={classes.summaryItem}>

                            <Text>{item.day} - {item.meal}</Text>

                            <Text type="secondary">₹{item.cost}</Text>

                        </div>

                    ))

                ) : (

                    <Empty

                        image={Empty.PRESENTED\_IMAGE\_SIMPLE}

                        description="No meals selected yet"

                        className={classes.emptySelection}

                    />

                )}

            </div>

            <div className={classes.summaryFooter}>

                <div className={classes.totalCost}>

                    <Text strong>Total Cost:</Text>

                    <Title level={3} className={classes.costAmount}>₹{cost}</Title>

                </div>

                <Button

                    disabled={loading || !cost}

                    onClick={handlePayment}

                    className={classes.buyButton}

                    type="primary"

                    size="large"

                    icon={<ShoppingCartOutlined />}

                >

                    Confirm Order

                </Button>

            </div>

        </Card>

    );

}

export default function BuyPage() {

    const [cost, setCost] = useState(0);

    const [loading, setLoading] = useState(true);

    const [bought, setBought] = useState(false);

    const [activeTab, setActiveTab] = useState('all');

    const isSmallScreen = useMediaQuery({ maxWidth: 768 });

    const [menu, setMenu] = useState([]);

    const [selected, setSelected] = useState({});

    // New state to track already purchased meals

    const [alreadyPurchased, setAlreadyPurchased] = useState({});

    // Initialize menu and selected states

    useEffect(() => {

        // Initialize menu with empty data for all days

        const initialMenu = Array(7).fill().map((\_, i) => ({

            day: Object.keys(dayToNum)[i],

            breakfast: { text: "", cost: 0 },

            lunch: { text: "", cost: 0 },

            dinner: { text: "", cost: 0 },

        }));

        // Initialize selected state for all days

        const initialSelected = Object.fromEntries(

            Object.keys(dayToNum).map(day => [

                day, { breakfast: false, lunch: false, dinner: false }

            ])

        );

        setMenu(initialMenu);

        setSelected(initialSelected);

    }, []);

    // Calculate cost when selection changes

    useEffect(() => {

        const totalCost = Object.entries(selected).reduce((acc, [day, meals]) => {

            const dayIndex = dayToNum[day];

            const dayData = menu[dayIndex];

            if (!dayData) return acc;

            return acc + Object.entries(meals).reduce((dayAcc, [meal, isSelected]) => {

                if (!isSelected || !dayData[meal]) return dayAcc;

                return dayAcc + (parseInt(dayData[meal].cost) || 0);

            }, 0);

        }, 0);

        setCost(totalCost);

    }, [menu, selected]);

    // Fetch menu and prices

    useEffect(() => {

        const fetchData = async () => {

            try {

                const [menuRes, pricesRes] = await Promise.all([

                    api.get('api/data/menu'),

                    api.get('api/data/time')

                ]);

                const mealPrices = pricesRes.data.reduce((acc, { meal, cost }) =>

                    ({ ...acc, [meal]: cost }), {});

                // Process the menu data and handle "Saturdayday" typo

                const processedMenuData = menuRes.data.map(day => {

                    // Fix for "Saturdayday" typo

                    const dayName = day.day === "Saturday" ? "saturday" : day.day.toLowerCase();

                    return {

                        ...day,

                        day: dayName,

                        breakfast: { text: day.breakfast, cost: mealPrices.breakfast || 0 },

                        lunch: { text: day.lunch, cost: mealPrices.lunch || 0 },

                        dinner: { text: day.dinner, cost: mealPrices.dinner || 0 }

                    };

                });

                // Create a new menu array with the correct structure

                const updatedMenu = Array(7).fill().map((\_, i) => {

                    const dayName = Object.keys(dayToNum)[i];

                    const dayData = processedMenuData.find(d => d.day === dayName);

                    if (dayData) {

                        return dayData;

                    }

                    // Return default data for missing days

                    return {

                        day: dayName,

                        breakfast: { text: "", cost: mealPrices.breakfast || 0 },

                        lunch: { text: "", cost: mealPrices.lunch || 0 },

                        dinner: { text: "", cost: mealPrices.dinner || 0 }

                    };

                });

                setMenu(updatedMenu);

                setLoading(false);

            } catch (error) {

                console.error("Error fetching data:", error);

                message.error('Failed to fetch data');

                setLoading(false);

            }

        };

        fetchData();

    }, []);

    // Check existing orders and fetch already purchased meals

    useEffect(() => {

        Promise.all([

            api.get('api/user/boughtNextWeek'),

            api.get('api/user/data') // Get already purchased meals data

        ])

            .then(([boughtRes, userData]) => {

                setBought(boughtRes.data);

                // Process user data to get already purchased meals

                const thisWeekData = userData.data?.this || {};

                // Create a normalized structure for already purchased meals

                const purchasedMeals = {};

                Object.entries(thisWeekData).forEach(([day, meals]) => {

                    // Normalize day name (handle potential typos)

                    const normalizedDay = day.toLowerCase().replace('saturdayday', 'saturday');

                    purchasedMeals[normalizedDay] = {

                        breakfast: !!meals.breakfast,

                        lunch: !!meals.lunch,

                        dinner: !!meals.dinner

                    };

                });

                setAlreadyPurchased(purchasedMeals);

                // Remove already purchased meals from selection

                setSelected(prev => {

                    const newSelected = { ...prev };

                    Object.entries(purchasedMeals).forEach(([day, meals]) => {

                        if (newSelected[day]) {

                            Object.entries(meals).forEach(([meal, isPurchased]) => {

                                if (isPurchased) {

                                    newSelected[day][meal] = false;

                                }

                            });

                        }

                    });

                    return newSelected;

                });

            })

            .catch(() => {

                message.error('Failed to check existing orders');

            });

    }, []);

    // Filter days based on active tab

    const getFilteredDays = () => {

        if (activeTab === 'all') return menu;

        if (activeTab === 'weekdays')

            return menu.filter(day => day && ['monday', 'tuesday', 'wednesday', 'thursday', 'friday'].includes(day.day));

        if (activeTab === 'weekend')

            return menu.filter(day => day && ['saturday', 'sunday'].includes(day.day));

        return menu;

    };

    return (

        <div className={classes.pageContainer}>

            <div className={classes.headerSection}>

                <Title level={2}>Weekly Meal Planner</Title>

                <Text type="secondary">Choose your meals for the upcoming week</Text>

            </div>

            <Tabs

                activeKey={activeTab}

                onChange={setActiveTab}

                className={classes.menuTabs}

                type="card"

            >

                <TabPane tab="All Days" key="all" />

                <TabPane tab="Weekdays" key="weekdays" />

                <TabPane tab="Weekend" key="weekend" />

            </Tabs>

            <div className={classes.menuLayout}>

                <div className={classes.menuContainer}>

                    {getFilteredDays().filter(dayData => dayData && dayData.day).map((dayData) => (

                        <MealCard

                            key={dayData.day}

                            dayData={dayData}

                            selected={selected}

                            setSelected={setSelected}

                            loading={loading}

                            alreadyPurchased={alreadyPurchased} // Pass already purchased meals

                        />

                    ))}

                </div>

                <div className={classes.orderSummaryContainer}>

                    <OrderSummary

                        selected={selected}

                        loading={loading}

                        cost={cost}

                        setBought={setBought}

                        menu={menu}

                        alreadyPurchased={alreadyPurchased} // Pass already purchased meals

                    />

                </div>

            </div>

        </div>

    );

}

Home/index.js

import classes from './index.module.css';

import { Button } from 'antd';

import { Link } from 'react-router-dom';

import { useEffect, useState } from 'react';

export default function HomePage() {

    // Add this state and function to handle sign-in

    const [status, setStatus] = useState(false);

    // Handle sign-in the same way as in MenuBar

    const handleSignIn = (e) => {

        e.preventDefault();

        try {

            window.location.href = window.APIROOT + 'api/auth/signin';

        } catch (error) {

            console.error('Failed to redirect to sign in');

        }

    };

    return (

        <div className={classes.container}>

            {/\* Hero Section \*/}

            <section className={classes.hero} id="home">

                <div className={classes.heroContent}>

                    <h1>

                        Fresh Food,<br />

                        <span className={classes.highlight}>Delivered</span> To<br />

                        Your Campus

                    </h1>

                    <p>Elevate your campus dining experience with delicious, affordable meals that fuel your academic journey.</p>

                    <div className={classes.buttons}>

                        <Link to="/schedule">

                            <Button type="primary" size="large" className={classes.primaryBtn}>

                                Explore Menu

                            </Button>

                        </Link>

                        {/\* Update Join Now button to use the same sign-in handler \*/}

                        <Button

                            size="large"

                            className={classes.secondaryBtn}

                            onClick={handleSignIn}

                        >

                            Join Now

                        </Button>

                    </div>

                    <div className={classes.statRow}>

                        <div className={classes.stat}>

                            <span className={classes.statNumber}>4.8</span>

                            <span className={classes.statLabel}>Customer Rating</span>

                        </div>

                        <div className={classes.statDivider}></div>

                        <div className={classes.stat}>

                            <span className={classes.statNumber}>15+</span>

                            <span className={classes.statLabel}>Daily Specials</span>

                        </div>

                        <div className={classes.statDivider}></div>

                        <div className={classes.stat}>

                            <span className={classes.statNumber}>3K+</span>

                            <span className={classes.statLabel}>Happy Students</span>

                        </div>

                    </div>

                </div>

                <div className={classes.heroImageContainer}>

                    <div className={classes.heroImageWrapper}>

                        <img src="/assets/Dish.png" alt="Delicious Campus Food" className={classes.heroImage} />

                        <div className={classes.foodTag}>

                            <span className={classes.tagIcon}>🔥</span>

                            <span className={classes.tagText}>Most Popular</span>

                        </div>

                    </div>

                    <div className={classes.heroPattern}></div>

                </div>

            </section>

            {/\* About Us Section \*/}

            <section className={classes.about} id="about">

                <div className={classes.aboutWrapper}>

                    <div className={classes.aboutImageSection}>

                        <div className={classes.aboutImageMain}>

                            <img src="https://images.pexels.com/photos/3217156/pexels-photo-3217156.jpeg?auto=compress&cs=tinysrgb&w=600" alt="Our Team Cooking" />

                        </div>

                        <div className={classes.aboutImageAccent}>

                            <img src="https://media.istockphoto.com/id/1081422898/photo/pan-fried-duck.jpg?s=612x612&w=0&k=20&c=kzlrX7KJivvufQx9mLd-gMiMHR6lC2cgX009k9XO6VA=" alt="Food Preparation" />

                        </div>

                        <div className={classes.aboutPattern}></div>

                    </div>

                    <div className={classes.aboutContent}>

                        <h4 className={classes.aboutPreTitle}>Our Story</h4>

                        <h2 className={classes.aboutTitle}>Crafting Campus Cuisine Since <span className={classes.highlight}>2022</span></h2>

                        <p className={classes.aboutText}>

                            CampusBite began with a simple mission: to transform campus dining with delicious,

                            affordable meals that fuel academic success. We understand the challenges of

                            student life and craft our menu to provide nutritious options that keep you energized.

                        </p>

                        <p className={classes.aboutText}>

                            What sets us apart is our commitment to quality ingredients, diverse flavors,

                            and a seamless dining experience designed specifically for campus life.

                        </p>

                        <div className={classes.aboutFeatures}>

                            <div className={classes.aboutFeature}>

                                <div className={classes.featureIcon}>

                                    <i className="fas fa-leaf"></i>

                                </div>

                                <div className={classes.featureText}>

                                    <h4>Fresh Ingredients</h4>

                                    <p>Locally sourced whenever possible</p>

                                </div>

                            </div>

                            <div className={classes.aboutFeature}>

                                <div className={classes.featureIcon}>

                                    <i className="fas fa-utensils"></i>

                                </div>

                                <div className={classes.featureText}>

                                    <h4>Diverse Menu</h4>

                                    <p>Options for all dietary preferences</p>

                                </div>

                            </div>

                        </div>

                        <Link to="/about">

                        </Link>

                    </div>

                </div>

            </section>

            {/\* Contact Section \*/}

            <section className={classes.contact} id="contact">

                <h2 className={classes.sectionTitle}>Contact Us</h2>

                <p className={classes.contactText}>

                    Have questions? Reach out to us at <a href="mailto:support@campusbite.com">support@campusbite.com</a> or call us at (123) 456-7890.

                </p>

                <div className={classes.contactInfo}>

                    <div className={classes.contactItem}>

                        <i className="fas fa-map-marker-alt"></i>

                        <p>Indira College Of Engineering And Management , Parandwadi, Pune</p>

                    </div>

                    <div className={classes.contactItem}>

                        <i className="fas fa-clock"></i>

                        <p>Open: Mon-Fri, 7:30 AM - 9:00 PM</p>

                    </div>

                </div>

            </section>

        </div>

    );

}

Purchasehistory/index.js

import { Spin, Card, message } from 'antd';

import { useMediaQuery } from 'react-responsive';

import { useState, useEffect } from "react";

import { CheckCircleFilled, CalendarOutlined } from '@ant-design/icons';

import api from '../..';

import styles from './index.module.css';

export default function PurchaseHistoryPage() {

  const isMobile = useMediaQuery({ query: '(max-width: 750px)' });

  const [activeWeek, setActiveWeek] = useState("this");

  const [menu, setMenu] = useState([]);

  const [loading, setLoading] = useState(false);

  // Define normalized day mapping

  const dayMapping = {

    "monday": "Monday",

    "tuesday": "Tuesday",

    "wednesday": "Wednesday",

    "thursday": "Thursday",

    "friday": "Friday",

    "saturday": "Saturday",

    "sunday": "Sunday",

    // Handle potential typo from API

    "saturdayday": "Saturday"

  };

  useEffect(() => {

    const fetchData = async () => {

      try {

        setLoading(true);

        const [menuResponse, buyerResponse] = await Promise.all([

          api.get('api/data/menu'),

          api.get('api/user/data')

        ]);

        const buyerWeekData = buyerResponse.data?.[activeWeek] || {};

        // Process menu data with normalized day names

        const formattedData = menuResponse.data.map(r => {

          // Normalize the day name to lowercase for consistent comparison

          const normalizedDay = r.day.toLowerCase().replace('saturdayday', 'saturday');

          // Get buyer data using the normalized day name

          const buyerDayData = buyerWeekData?.[normalizedDay] ||

                               buyerWeekData?.[r.day] || {};

          return {

            day: normalizedDay,

            displayDay: dayMapping[normalizedDay] || normalizedDay,

            breakfast: {

              text: r.breakfast,

              selected: buyerDayData.breakfast || false

            },

            lunch: {

              text: r.lunch,

              selected: buyerDayData.lunch || false

            },

            dinner: {

              text: r.dinner,

              selected: buyerDayData.dinner || false

            }

          };

        });

        setMenu(formattedData);

      } catch (error) {

        console.error("Fetch error:", error);

        message.error('Failed to fetch purchase history');

      } finally {

        setLoading(false);

      }

    };

    fetchData();

  }, [activeWeek]);

  const days = ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday"];

  const mealTypes = ["breakfast", "lunch", "dinner"];

  const mealLabels = {

    breakfast: "Breakfast",

    lunch: "Lunch",

    dinner: "Dinner"

  };

  const getMealCount = () => {

    return menu.reduce((count, day) => {

      mealTypes.forEach(mealType => {

        if (day[mealType]?.selected) count++;

      });

      return count;

    }, 0);

  };

  return (

    <div className={styles.purchaseHistoryContainer}>

      <div className={styles.headerSection}>

        <h1 className={styles.pageTitle}>

          <span className={styles.iconWrapper}>

            <CalendarOutlined />

          </span>

          Your Meal Plan

        </h1>

        <div className={styles.mealStats}>

          <span>Selected Meals: {getMealCount()}</span>

        </div>

      </div>

      {loading ? (

        <div className={styles.loadingContainer}>

          <Spin size="large" />

        </div>

      ) : (

        <div className={styles.mealCardsContainer}>

          {days.map(day => {

            // Find day data using normalized day name

            const dayData = menu.find(d => d.day === day);

            if (!dayData) {

              console.log(`No data found for ${day}`);

              return null;

            }

            return (

              <Card

                key={day}

                title={dayData.displayDay || day.charAt(0).toUpperCase() + day.slice(1)}

                className={styles.dayCard}

                headStyle={{ backgroundColor: '#f7f7f7' }}

              >

                {mealTypes.map(mealType => (

                  <div

                    key={mealType}

                    className={`${styles.mealItem} ${dayData[mealType].selected ? styles.selectedMeal : ''}`}

                  >

                    <div className={styles.mealHeader}>

                      {dayData[mealType].selected && (

                        <CheckCircleFilled className={styles.checkIcon} />

                      )}

                      <h3 className={styles.mealTitle}>{mealLabels[mealType]}</h3>

                    </div>

                    <p className={styles.mealText}>{dayData[mealType].text || "No menu available"}</p>

                  </div>

                ))}

              </Card>

            );

          })}

        </div>

      )}

    </div>

  );

}

Qrcode/index.js

import { QRCodeSVG } from 'qrcode.react';

import { ReloadOutlined, QuestionOutlined, LoadingOutlined, LockOutlined, CloseOutlined } from '@ant-design/icons';

import { Button, notification, Space, message, Spin, Card, Typography } from 'antd';

import { useState, useEffect } from "react";

import api from '../..';

import { motion } from 'framer-motion';

const { Title, Text } = Typography;

export default function QRCodePage() {

    const [loading, setLoading] = useState(false);

    const [rawCode, setRawCode] = useState(null);

    const [initialLoading, setInitialLoading] = useState(true);

    useEffect(() => {

        const fetchData = async () => {

            setLoading(true);

            try {

                const response = await api.get('api/user/data');

                const code = response.data.secret + response.data.email;

                setRawCode(code);

                setInitialLoading(false);

                setLoading(false);

            } catch (error) {

                message.error('Failed to fetch QR code');

                setInitialLoading(false);

                setLoading(false);

            }

        }

        fetchData();

    }, []);

    const handleCreateNew = async () => {

        setLoading(true);

        try {

            const response = await api.get('api/user/resetSecret');

            const code = response.data.secret + response.data.email;

            setRawCode(code);

            setLoading(false);

            message.success({

                content: 'New QR code created successfully',

                icon: <LockOutlined style={{ color: '#52c41a' }} />

            });

        } catch (error) {

            setLoading(false);

            message.error('Failed to create new QR code');

        }

    };

    const showInfo = () => {

        notification.info({

            message: <Title level={4}>Security Information</Title>,

            description:

                <Text>

                    Keep your QR code private and secure. If you think it has been compromised,

                    click the "Create New" button to generate a new one immediately.

                </Text>,

            message: <b>Information</b>,

            description: 'Keep your QR code private. If you think it has been compromised, reissue a new one using this Create New button.',

            placement: 'top',

            closeIcon: <CloseOutlined style={{ fontSize: '16px' }} />,  // Adding cross symbol

            duration: 10,

            style: {

                borderRadius: '8px',

                boxShadow: '0 4px 12px rgba(0, 0, 0, 0.15)'

            }

        });

    };

    return (

        <motion.div

            initial={{ opacity: 0 }}

            animate={{ opacity: 1 }}

            style={{

                height: 'calc(100vh - 17vh)',

                display: 'flex',

                flexDirection: 'column',

                gap: '2rem',

                justifyContent: 'center',

                alignItems: 'center',

                padding: '0 1rem'

            }}

        >

            <Title level={2}>Your Secure QR Code</Title>

            <Card

                style={{

                    width: 300,

                    height: 350,

                    display: 'flex',

                    justifyContent: 'center',

                    alignItems: 'center',

                    boxShadow: '0 8px 16px rgba(0, 0, 0, 0.1)',

                    borderRadius: '12px',

                    position: 'relative'

                }}

                bordered={false}

            >

                {initialLoading ? (

                    <div style={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100%' }}>

                        <Spin size="large" indicator={<LoadingOutlined style={{ fontSize: 36, color: '#1f94ff' }} spin />} />

                    </div>

                ) : (

                    <div style={{ position: 'relative', display: 'flex', justifyContent: 'center', alignItems: 'center' }}>

                        {loading && (

                            <div style={{

                                position: 'absolute',

                                top: 0,

                                left: 0,

                                width: '100%',

                                height: '100%',

                                backgroundColor: 'rgba(255, 255, 255, 0.8)',

                                display: 'flex',

                                justifyContent: 'center',

                                alignItems: 'center',

                                zIndex: 1,

                                transition: '0.5s'

                            }}>

                                <Spin indicator={<LoadingOutlined style={{ fontSize: 36, color: '#1f94ff' }} spin />} />

                            </div>

                        )}

                        <div>

                            <QRCodeSVG

                                size={256}

                                value={rawCode || 'loading'}

                                level="H"

                                includeMargin={true}

                                renderAs="svg"

                                imageSettings={{

                                    excavate: true,

                                    width: 24,

                                    height: 24

                                }}

                            />

                        </div>

                    </div>

                )}

            </Card>

            <Space size="large">

                <Button

                    type="primary"

                    size="large"

                    icon={<ReloadOutlined />}

                    onClick={handleCreateNew}

                    loading={loading}

                    style={{

                        height: '48px',

                        borderRadius: '8px',

                        boxShadow: '0 2px 6px rgba(24, 144, 255, 0.2)'

                    }}

                >

                    Create New

                </Button>

                <Button

                    size="large"

                    icon={<QuestionOutlined />}

                    onClick={showInfo}

                    style={{

                        height: '48px',

                        borderRadius: '8px'

                    }}

                >

                    Help

                </Button>

            </Space>

        </motion.div>

    );

}

Scanqr/index.js

import { useState, useEffect } from "react";

import { BrowserQRCodeReader } from "@zxing/browser";

import { Radio, Card, Button, message } from "antd";

import {

  CloseSquareOutlined,

  CheckSquareOutlined,

  ReloadOutlined,

  LoadingOutlined,

  QrcodeOutlined

} from "@ant-design/icons";

import { motion } from "framer-motion";

import api from "../..";

export default function ScanQRPage() {

  const [data, setData] = useState();

  const [type, setType] = useState(null);

  const [valid, setValid] = useState(null);

  const [videoElement, setVideoElement] = useState(null);

  const [scanning, setScanning] = useState(true);

  const [controls, setControls] = useState(null);

  const checkCoupon = async (postData) => {

    try {

      setScanning(false);

      const response = await api.post("api/user/checkCoupon", postData);

      setValid(response.data);

    } catch (error) {

      message.error("Failed to verify coupon");

      setData(null);

      setScanning(true);

    }

  };

  useEffect(() => {

    if (!videoElement || !scanning) return;

    const codeReader = new BrowserQRCodeReader();

    let controlsRef = null;

    const decode = async () => {

      try {

        // Store the controls so we can stop/cleanup properly

        controlsRef = await codeReader.decodeFromVideoDevice(

          undefined,

          videoElement,

          (result) => {

            if (result) {

              setData(result.text);

              // Don't call reset - instead let's store the controls

              if (controlsRef) {

                controlsRef.stop();

              }

            }

          }

        );

        setControls(controlsRef);

      } catch (error) {

        console.error("QR Scanner Error:", error);

      }

    };

    decode();

    return () => {

      // Clean up properly using the controls

      if (controlsRef) {

        controlsRef.stop();

      }

    };

  }, [videoElement, scanning]);

  useEffect(() => {

    if (!data || !type) return;

    const secret = data.substring(0, 4);

    const email = data.substring(4);

    const day = new Intl.DateTimeFormat("en-US", { weekday: "long" })

      .format(new Date())

      .toLowerCase();

    checkCoupon({ secret, email, day, type });

  }, [data, type]);

  const resetScan = () => {

    // Stop the current scanner

    if (controls) {

      controls.stop();

    }

    setData(null);

    setValid(null);

    setScanning(true);

  };

  const getStatusIcon = () => {

    if (valid === null && scanning) {

      return <QrcodeOutlined style={{ color: '#1890ff', fontSize: '4rem' }} className="animate-pulse" />;

    } else if (valid === null && !scanning) {

      return <LoadingOutlined style={{ color: '#1890ff', fontSize: '4rem' }} spin />;

    } else if (valid) {

      return <CheckSquareOutlined style={{ color: '#52c41a', fontSize: '4rem' }} />;

    } else {

      return <CloseSquareOutlined style={{ color: '#f5222d', fontSize: '4rem' }} />;

    }

  };

  const getStatusMessage = () => {

    if (valid === null && scanning) {

      return "Ready to scan";

    } else if (valid === null && !scanning) {

      return "Processing...";

    } else if (valid) {

      return "Valid Coupon!";

    } else {

      return "Invalid Coupon";

    }

  };

  return (

    <div style={{

      display: 'flex',

      flexDirection: 'column',

      alignItems: 'center',

      justifyContent: 'space-between',

      minHeight: '100vh',

      background: 'linear-gradient(to bottom, #f0f5ff, #e6f7ff)',

      padding: '1rem'

    }}>

      <div style={{

        width: '100%',

        maxWidth: '500px',

        textAlign: 'center',

        marginBottom: '1rem'

      }}>

        <h1 style={{ fontSize: '1.5rem', fontWeight: 'bold', color: '#1d39c4', marginBottom: '0.5rem' }}>

          Meal QR Scanner

        </h1>

        <p style={{ color: '#666', marginBottom: '1rem' }}>

          Scan your QR code to verify your meal coupon

        </p>

      </div>

      <div style={{ width: '100%', maxWidth: '500px' }}>

        {/\* Camera View \*/}

        <div style={{

          position: 'relative',

          width: '100%',

          borderRadius: '8px',

          overflow: 'hidden',

          boxShadow: '0 4px 12px rgba(0,0,0,0.1)',

          marginBottom: '1rem',

          background: '#fff'

        }}>

          <div style={{

            opacity: !type ? 0.5 : 1,

            transition: 'opacity 0.3s'

          }}>

            {scanning && (

              <div style={{

                position: 'absolute',

                inset: 0,

                border: '4px solid #1890ff',

                borderRadius: '8px',

                zIndex: 10,

                animation: 'pulse 2s infinite'

              }}></div>

            )}

            <video

              ref={setVideoElement}

              style={{

                width: '100%',

                height: 'auto',

                minHeight: '300px',

                objectFit: 'cover',

                borderRadius: '8px'

              }}

            />

          </div>

          {!type && (

            <div style={{

              position: 'absolute',

              inset: 0,

              display: 'flex',

              alignItems: 'center',

              justifyContent: 'center',

              background: 'rgba(0,0,0,0.6)',

              color: 'white',

              fontSize: '1.125rem',

              fontWeight: 600,

              padding: '1rem',

              textAlign: 'center'

            }}>

              Please select a meal type to start scanning

            </div>

          )}

        </div>

        {/\* Controls \*/}

        <Card style={{

          width: '100%',

          boxShadow: '0 4px 12px rgba(0,0,0,0.1)',

          borderRadius: '12px',

          overflow: 'hidden'

        }}>

          <div style={{ marginBottom: '1rem' }}>

            <Radio.Group

              buttonStyle="solid"

              onChange={(e) => setType(e.target.value)}

              value={type}

              size="large"

              style={{ width: '100%', display: 'flex' }}

            >

              <Radio.Button value="breakfast" style={{ flex: 1, textAlign: 'center' }}>

                Breakfast

              </Radio.Button>

              <Radio.Button value="lunch" style={{ flex: 1, textAlign: 'center' }}>

                Lunch

              </Radio.Button>

              <Radio.Button value="dinner" style={{ flex: 1, textAlign: 'center' }}>

                Dinner

              </Radio.Button>

            </Radio.Group>

          </div>

          <motion.div

            layout

            style={{

              display: 'flex',

              flexDirection: 'column',

              alignItems: 'center',

              justifyContent: 'center',

              padding: '1.5rem'

            }}

          >

            {getStatusIcon()}

            <p style={{

              marginTop: '0.5rem',

              fontSize: '1.125rem',

              fontWeight: 500,

              color: '#333'

            }}>

              {getStatusMessage()}

            </p>

            {valid !== null && (

              <p style={{

                fontSize: '0.875rem',

                color: '#666',

                marginTop: '0.5rem',

                textAlign: 'center'

              }}>

                {valid ? "Your meal has been successfully verified" : "Please try again or contact support"}

              </p>

            )}

          </motion.div>

          <Button

            type="primary"

            icon={<ReloadOutlined />}

            onClick={resetScan}

            disabled={!type}

            size="large"

            style={{

              width: '100%',

              background: '#1890ff',

              borderColor: '#1890ff'

            }}

          >

            Scan New QR

          </Button>

        </Card>

      </div>

      <div style={{

        textAlign: 'center',

        color: '#666',

        fontSize: '0.875rem',

        marginTop: '1rem'

      }}>

        Position QR code within frame for scanning

      </div>

      <style jsx>{`

        @keyframes pulse {

          0% {

            opacity: 0.6;

          }

          50% {

            opacity: 1;

          }

          100% {

            opacity: 0.6;

          }

        }

        .animate-pulse {

          animation: pulse 2s infinite;

        }

      `}</style>

    </div>

  );

}

Schedule/index.js

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

import { Table, message, Card, Row, Col, Typography, Spin, Tabs, Badge, Tag, Divider, Button, Empty } from 'antd';

import { useMediaQuery } from 'react-responsive';

import { motion, AnimatePresence } from "framer-motion";

import {

  CalendarOutlined,

  ClockCircleOutlined,

  FireOutlined,

  TagOutlined,

  InfoCircleOutlined,

  StarFilled

} from '@ant-design/icons';

import WeekMenu from '../../components/WeekMenu';

import api from '../..';

import classes from './index.module.css';

const { Title, Text } = Typography;

const { TabPane } = Tabs;

// Enhanced timing columns with icons

const timingCol = [

    {

        title: <><FireOutlined style={{ marginRight: 8 }} />Meal Type</>,

        dataIndex: 'meal',

        key: 'meal',

        render: (text) => ({

            props: {

                style: { background: "#FAFAFA" },

            },

            children: <span style={{ fontWeight: 600, fontSize: '16px' }}>{text}</span>

        })

    },

    {

        title: <><ClockCircleOutlined style={{ marginRight: 8 }} />Serving Time</>,

        dataIndex: 'time',

        key: 'time',

        render: (text) => <Text strong>{text}</Text>

    }

];

export default function SchedulePage() {

    const mobile = useMediaQuery({ query: '(max-width: 750px)' });

    const tablet = useMediaQuery({ query: '(max-width: 992px) and (min-width: 751px)' });

    const [timingRow, setTimingRow] = useState([]);

    const [menu, setMenu] = useState([]);

    const [loading, setLoading] = useState(true);

    const [activeDay, setActiveDay] = useState(null);

    const [highlightedMeal, setHighlightedMeal] = useState(null);

    // Get today's day name

    const getTodayDayName = () => {

        const days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];

        return days[new Date().getDay()];

    };

    useEffect(() => {

        const fetchData = async () => {

            try {

                setLoading(true);

                // Parallel API requests

                const [timeResponse, menuResponse] = await Promise.all([

                    api.get('api/data/time'),

                    api.get('api/data/menu')

                ]);

                console.log("time:", timeResponse);

                console.log("menu:", menuResponse);

                // Format timing data for better display

                const formattedTimingData = timeResponse.data.map(item => ({

                    ...item,

                    key: item.meal.toLowerCase(),

                    // Add meal color codes based on meal type

                    mealColor: item.meal === 'Breakfast' ? '#52c41a' :

                              item.meal === 'Lunch' ? '#1890ff' :

                              item.meal === 'Dinner' ? '#722ed1' : '#faad14'

                }));

                setTimingRow(formattedTimingData);

                setMenu(menuResponse.data);

                // Set active day to today initially

                setActiveDay(getTodayDayName());

                setLoading(false);

            } catch (error) {

                console.error("Error fetching data:", error);

                message.error('Failed to fetch data from server');

                setLoading(false);

            }

        };

        fetchData();

    }, []);

    // Get today's meals

    const getTodayMenu = () => {

        if (!menu.length || !activeDay) return null;

        // Find the day's menu

        const dayMenu = menu.find(item => Object.keys(item).includes(activeDay));

        if (!dayMenu) return null;

        return dayMenu[activeDay];

    };

    // Get all available days from menu data in the correct order

    const getDays = () => {

        if (menu.length === 0) return [];

        // Define the correct order of days (Monday to Sunday)

        const orderedDays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];

        // Get all keys except "key" to find day names

        const availableDays = [];

        menu.forEach(item => {

            Object.keys(item).forEach(key => {

                if (key !== 'key' && !availableDays.includes(key)) {

                    availableDays.push(key);

                }

            });

        });

        // Sort the available days according to the defined order

        return availableDays.sort((a, b) => {

            return orderedDays.indexOf(a) - orderedDays.indexOf(b);

        });

    };

    // Highlight meal when user clicks on a meal timing

    const highlightMeal = (meal) => {

        setHighlightedMeal(meal);

        // Auto-scroll to menu section

        document.getElementById('menu-section')?.scrollIntoView({

            behavior: 'smooth',

            block: 'start'

        });

        // Auto-reset after 3 seconds

        setTimeout(() => {

            setHighlightedMeal(null);

        }, 3000);

    };

    const renderTimingCard = () => {

        return (

            <Card

                title={

                    <div className={classes.cardTitle}>

                        <ClockCircleOutlined className={classes.cardTitleIcon} />

                        <span>Meal Timings</span>

                    </div>

                }

                className={classes.timingCard}

                bordered={true}

            >

                {timingRow.length > 0 ? (

                    <motion.div initial={{ opacity: 0 }} animate={{ opacity: 1 }} transition={{ duration: 0.5 }}>

                        <div className={classes.timingsContainer}>

                            {timingRow.map((timing) => (

                                <motion.div

                                    key={timing.key}

                                    className={classes.timingItem}

                                    whileHover={{ scale: 1.02 }}

                                    onClick={() => highlightMeal(timing.meal)}

                                >

                                    <div className={classes.mealBadge} style={{ backgroundColor: timing.mealColor }}></div>

                                    <div className={classes.mealInfo}>

                                        <Text strong className={classes.mealName}>{timing.meal}</Text>

                                        <Text className={classes.mealTime}>{timing.time}</Text>

                                    </div>

                                </motion.div>

                            ))}

                        </div>

                    </motion.div>

                ) : (

                    <Empty description="No timing information available" />

                )}

            </Card>

        );

    };

    const renderTodayHighlight = () => {

        const todayMenu = getTodayMenu();

        if (!todayMenu) return null;

        return (

            <Card

                className={classes.todayHighlightCard}

                title={

                    <div className={classes.cardTitle}>

                        <StarFilled className={classes.cardTitleIcon} style={{ color: '#faad14' }} />

                        <span>Today's Special</span>

                    </div>

                }

            >

                <div className={classes.todaySpecials}>

                    {Object.entries(todayMenu).map(([meal, items]) => (

                        <div key={meal} className={classes.todayMeal}>

                            <Tag color={

                                meal === 'Breakfast' ? 'green' :

                                meal === 'Lunch' ? 'blue' :

                                meal === 'Dinner' ? 'purple' : 'orange'

                            }>

                                {meal}

                            </Tag>

                            <Text>{items}</Text>

                        </div>

                    ))}

                </div>

            </Card>

        );

    };

    const renderMenuContent = () => {

        return (

            <Card

                id="menu-section"

                className={classes.menuCard}

                title={

                    <div className={classes.cardTitle}>

                        <TagOutlined className={classes.cardTitleIcon} />

                        <span>Weekly Menu</span>

                    </div>

                }

            >

                <AnimatePresence mode="wait">

                    <motion.div

                        key={activeDay}

                        initial={{ opacity: 0, y: 10 }}

                        animate={{ opacity: 1, y: 0 }}

                        exit={{ opacity: 0, y: -10 }}

                        transition={{ duration: 0.3 }}

                        className={classes.weekMenuContainer}

                    >

                        <WeekMenu

                            menu={menu}

                            mobile={mobile}

                            activeDay={activeDay}

                            highlightedMeal={highlightedMeal}

                            getDays={getDays}  // Pass the getDays function to WeekMenu

                        />

                    </motion.div>

                </AnimatePresence>

            </Card>

        );

    };

    if (loading) {

        return (

            <div className={classes.loadingContainer}>

                <Spin size="large" />

                <Text className={classes.loadingText}>Loading menu data...</Text>

            </div>

        );

    }

    // Mobile view with tabs

    if (mobile) {

        return (

            <div className={classes.scheduleContainer}>

                <div className={classes.menuHeader}>

                    <Title level={2} className={classes.mainTitle}>Our Menu</Title>

                    <Text className={classes.subTitleText}>Discover what's cooking this week</Text>

                </div>

                {renderTodayHighlight()}

                <Tabs

                    defaultActiveKey="timing"

                    className={classes.mobileTabs}

                >

                    <TabPane tab={<span><ClockCircleOutlined /> Timings</span>} key="timing">

                        {renderTimingCard()}

                    </TabPane>

                    <TabPane tab={<span><TagOutlined /> Weekly Menu</span>} key="menu">

                        {renderMenuContent()}

                    </TabPane>

                </Tabs>

            </div>

        );

    }

    // Desktop/tablet view - side by side tables

    return (

        <div className={classes.scheduleContainer}>

            <div className={classes.menuHeader}>

                <motion.div

                    initial={{ opacity: 0, y: -20 }}

                    animate={{ opacity: 1, y: 0 }}

                    transition={{ duration: 0.5 }}

                >

                    <Title level={2} className={classes.mainTitle}>Our Menu</Title>

                    <Text className={classes.subTitleText}>Discover what's cooking this week</Text>

                </motion.div>

            </div>

            {renderTodayHighlight()}

            <div className={classes.sideByContainer}>

                <div className={classes.tableCard}>

                    {renderTimingCard()}

                </div>

                <div className={classes.tableCard}>

                    {renderMenuContent()}

                </div>

            </div>

        </div>

    );

}

Totalmeals/index.js

import { useState, useEffect } from 'react';

import {

  Button,

  Space,

  message,

  Card,

  Spin,

  Tabs,

  Empty,

  Badge,

  Tooltip

} from 'antd';

import {

  EditOutlined,

  ReloadOutlined,

  CalendarOutlined

} from '@ant-design/icons';

import api from '../..';

import WeekMenu from '../../components/WeekMenu';

import classes from './index.module.css';

export default function TotalMealsPage() {

    const [activeWeek, setActiveWeek] = useState("this");

    const [menu, setMenu] = useState([]);

    const [loading, setLoading] = useState(true);

    const [lastUpdated, setLastUpdated] = useState(null);

    const [activeTab, setActiveTab] = useState('table');

    const fetchMenu = async (week = activeWeek) => {

        setLoading(true);

        try {

            const response = await api.post('api/admin/meals', { week });

            setMenu(response.data);

            setLastUpdated(new Date());

            setLoading(false);

        } catch (error) {

            message.error('Failed to fetch menu from server');

            setLoading(false);

        }

    };

    useEffect(() => {

        fetchMenu(activeWeek);

    }, [activeWeek]);

    const formatDate = (date) => {

        if (!date) return '';

        return date.toLocaleDateString('en-US', {

            month: 'short',

            day: 'numeric',

            year: 'numeric',

            hour: 'numeric',

            minute: 'numeric',

            hour12: true

        });

    };

    const handleEditMeals = () => {

        message.info('Edit functionality will be implemented here');

    };

    const renderContent = (week) => {

        if (loading && activeWeek === week) {

            return (

                <div className={classes.loadingContainer}>

                    <Spin size="large" tip="Loading menu..." />

                </div>

            );

        }

        if (menu.length === 0) {

            return <Empty description={`No meals scheduled for ${week === 'this' ? 'this' : 'next'} week`} />;

        }

        return (

            <>

                <WeekMenu menu={menu} mobile={false} />

            </>

        );

    };

    const items = [

        {

            key: 'this',

            label: (

                <span>

                    <CalendarOutlined /> This Week

                </span>

            ),

            children: (

                <Card

                    className={classes.menuCard}

                    title={

                        <div className={classes.cardTitle}>

                            <span className={classes.weekTitle}>

                                <Badge status="processing" text="Current Week Menu" />

                            </span>

                        </div>

                    }

                    extra={

                        <Space>

                            <Tooltip title="Refresh data">

                                <Button

                                    icon={<ReloadOutlined />}

                                    onClick={() => fetchMenu("this")}

                                    loading={loading && activeWeek === "this"}

                                >

                                    Refresh

                                </Button>

                            </Tooltip>

                        </Space>

                    }

                >

                    {renderContent('this')}

                </Card>

            )

        },

    ];

    return (

        <div className={classes.pageContainer}>

            <h1 className={classes.simpleTitleHeader}>Meal Management Dashboard</h1>

            <Tabs

                activeKey={activeWeek}

                items={items}

                onChange={setActiveWeek}

                className={classes.tabs}

                tabBarStyle={{ marginBottom: '16px' }}

            />

            {lastUpdated && (

                <div className={classes.lastUpdated}>

                    Last updated: {formatDate(lastUpdated)}

                </div>

            )}

        </div>

    );

}

Menubar/index.js

import classes from "./index.module.css";

import { useState, useEffect } from "react";

import { motion, AnimatePresence } from "framer-motion";

import { Link, useNavigate, useLocation } from "react-router-dom";

import { useMediaQuery } from "react-responsive";

import {

    MenuOutlined,

    CloseOutlined,

    UserOutlined,

    ShoppingCartOutlined,

    QrcodeOutlined,

    CalendarOutlined,

    TableOutlined,

    SettingOutlined,

    SolutionOutlined,

    ScanOutlined,

    HomeOutlined,

    InfoCircleOutlined,

    AppstoreOutlined,

    ContactsOutlined,

    DownOutlined

} from '@ant-design/icons';

import { Menu, message } from 'antd';

import api from "../..";

// Function to handle section navigation

const handleSectionNavigation = (sectionId, navigate, setOpen) => {

    // If already on homepage, just scroll to the section

    if (window.location.pathname === '/') {

        const element = document.getElementById(sectionId);

        if (element) {

            element.scrollIntoView({ behavior: 'smooth' });

        }

    } else {

        // If not on homepage, navigate to homepage with section hash

        navigate(`/#${sectionId}`);

    }

    // Close mobile menu if it's open

    setOpen(false);

};

const getItem = (label, link, icon, key, navigate, setOpen) => {

    // Special handling for section links

    if (link.startsWith('/#')) {

        const sectionId = link.substring(2); // Remove '/#' to get the section ID

        return {

            key,

            icon,

            label: <a onClick={() => handleSectionNavigation(sectionId, navigate, setOpen)}>{label}</a>,

        };

    }

    // Regular navigation links

    return {

        key,

        icon,

        label: <Link to={link}>{label}</Link>,

    };

};

const getLoginItem = (label, link, icon, key, handleSignInOut) => ({

    key,

    icon,

    label: <a href={link} onClick={handleSignInOut}>{label}</a>,

});

export default function MenuBar() {

    const [open, setOpen] = useState(false);

    const [status, setStatus] = useState(false);

    const [showFeatures, setShowFeatures] = useState(false);

    const [transparent, setTransparent] = useState(true);

    const [hoveredCard, setHoveredCard] = useState(null);

    const [expandedMenu, setExpandedMenu] = useState(false);

    const isMobile = useMediaQuery({ maxWidth: 768 });

    const navigate = useNavigate();

    const location = useLocation();

    // Add scroll listener for transparent menu effect

    useEffect(() => {

        const handleScroll = () => {

            const scrollPosition = window.scrollY;

            if (scrollPosition > 50) {

                setTransparent(false);

            } else {

                setTransparent(true);

            }

        };

        window.addEventListener('scroll', handleScroll);

        return () => {

            window.removeEventListener('scroll', handleScroll);

        };

    }, []);

    useEffect(() => {

        const fetchData = async () => {

            try {

                console.log("requesting status");

                const response = await api.get('api/data/status');

                console.log(response);

                setStatus(response.data);

            } catch (error) {

                message.error('Failed to fetch data from server');

            }

        };

        fetchData();

    }, []);

    // Reset hovered card when features menu is closed

    useEffect(() => {

        if (!showFeatures) {

            setHoveredCard(null);

        }

    }, [showFeatures]);

    const handleSignInOut = async (e) => {

        e.preventDefault();

        if (status?.loggedIn) {

            try {

                await api.get('api/auth/signout');

                const response = await api.get('api/data/status');

                setStatus(response.data);

                navigate('/');

            } catch (error) {

                message.error('Failed to sign out');

            }

        } else {

            try {

                window.location.href = window.APIROOT + 'api/auth/signin';

            } catch (error) {

                message.error('Failed to sign in');

            }

        }

    };

    const mainNavItems = status?.loggedIn

      ? [ // Only show these when logged IN

          getItem('Menu', '/schedule', <TableOutlined />, 'menu', navigate, setOpen),

        ]

      : [ // Show full menu when logged OUT

          getItem('Home', '/#home', <HomeOutlined />, 'home', navigate, setOpen),

          getItem('About Us', '/#about', <InfoCircleOutlined />, 'about', navigate, setOpen),

          getItem('Features', '/#features', <AppstoreOutlined />, 'features', navigate, setOpen),

          getItem('Menu', '/schedule', <TableOutlined />, 'menu', navigate, setOpen),

          getItem('Contact', '/#contact', <ContactsOutlined />, 'contact', navigate, setOpen),

        ];

    // Create login item separately so we can position it first

    const loginItem = window.location.pathname === '/'

    ? []

    : [

        getLoginItem(

            status?.loggedIn ? 'Sign out' : 'Sign in',

            status?.loggedIn ? window.APIROOT + 'api/auth/signout' : window.APIROOT + 'api/auth/signin',

            <UserOutlined />,

            '2',

            handleSignInOut

        )

    ];

    // Items for vertical menu - used when logged in

    const verticalMenuItems = [

        { icon: <HomeOutlined />, label: 'Home', link: '/', key: 'v-home' },

        { icon: <TableOutlined />, label: 'Schedule', link: '/schedule', key: 'v-schedule' },

        { icon: <ShoppingCartOutlined />, label: 'Buy Coupons', link: '/buy-coupons', key: 'v-coupons' },

        { icon: <CalendarOutlined />, label: 'Purchase History', link: '/purchase-history', key: 'v-history' },

        { icon: <QrcodeOutlined />, label: 'QR Code', link: '/qr-code', key: 'v-qrcode' },

    ];

    // Admin menu items - only shown if user is admin

    const adminMenuItems = status?.admin ? [

        { icon: <SettingOutlined />, label: 'Admin Panel', link: '/admin', key: 'v-admin' },

        { icon: <SolutionOutlined />, label: 'Total Meals', link: '/total-meals', key: 'v-meals' },

        { icon: <ScanOutlined />, label: 'Scan QR', link: '/scan-qr', key: 'v-scan' },

    ] : [];

    // Rest of the side menu items

    const otherSideMenuItems = [

        ...(status?.loggedIn ? [

            { type: 'divider' },

            getItem('Schedule', '/schedule', <TableOutlined />, '3', navigate, setOpen),

            getItem('Buy coupons', '/buy-coupons', <ShoppingCartOutlined />, '4', navigate, setOpen),

            getItem('Purchase history', '/purchase-history', <CalendarOutlined />, '5', navigate, setOpen),

            getItem('QR code', '/qr-code', <QrcodeOutlined />, '6', navigate, setOpen),

            ...(status?.admin ? [

                { type: 'divider' },

                getItem('Admin panel', '/admin', <SettingOutlined />, '7', navigate, setOpen),

                getItem('Total meals', '/total-meals', <SolutionOutlined />, '8', navigate, setOpen),

                getItem('Scan QR code', '/scan-qr', <ScanOutlined />, '9', navigate, setOpen),

            ] : []),

        ] : [])

    ];

    // Mobile menu items - ensure they include all navigation links

    const mobileMenuItems = [

        getLoginItem(

            status?.loggedIn ? 'Sign out' : 'Sign in',

            status?.loggedIn ? window.APIROOT + 'api/auth/signout' : window.APIROOT + 'api/auth/signin',

            <UserOutlined />,

            'mobile-signin',

            handleSignInOut

        ),

        { type: 'divider' },

        ...(status?.loggedIn

          ? [] // Hide all except essential items when logged in

          : [

              getItem('Home', '/#home', <HomeOutlined />, 'mobile-home', navigate, setOpen),

              getItem('About Us', '/#about', <InfoCircleOutlined />, 'mobile-about', navigate, setOpen),

              getItem('Features', '/#features', <AppstoreOutlined />, 'mobile-features', navigate, setOpen),

              getItem('Menu', '/schedule', <TableOutlined />, 'mobile-menu', navigate, setOpen),

              getItem('Contact', '/#contact', <ContactsOutlined />, 'mobile-contact', navigate, setOpen),

            ]

        ),

        ...otherSideMenuItems

    ];

    const items = isMobile

    ? mobileMenuItems

    : [...loginItem, ...otherSideMenuItems];

    // Feature mega menu items - updated to remove Dietary Preferences and Campus Delivery

    const featureItems = [

        {

            title: "Weekly Menu",

            icon: "fas fa-utensils",

            frontDescription: "Explore our diverse weekly menu",

            backDescription: "Browse through a variety of nutritious meal options updated every week to ensure freshness and variety",

            color: "#FF7043"

        },

        {

            title: "Digital Coupons",

            icon: "fas fa-ticket-alt",

            frontDescription: "Purchase meal coupons online",

            backDescription: "Securely buy and manage your meal coupons through our integrated Razorpay payment system",

            color: "#42A5F5"

        },

        {

            title: "QR Code System",

            icon: "fas fa-qrcode",

            frontDescription: "Quick and convenient redemption",

            backDescription: "Simply scan your unique QR code to redeem meals without waiting in long queues",

            color: "#66BB6A"

        },

        {

            title: "Meal Planning",

            icon: "fas fa-clipboard-list",

            frontDescription: "Plan your meals ahead of time",

            backDescription: "Schedule your meals for the week and get personalized nutrition information",

            color: "#AB47BC"

        }

    ];

    const showBurgerMenu = isMobile || status?.loggedIn || window.location.pathname !== '/';

    // Handle click on vertical menu item

    const handleVerticalMenuClick = (link) => {

        navigate(link);

        if (!isMobile) {

            setExpandedMenu(false); // Collapse menu after click

        }

    };

    return (

        <>

            {/\* Vertical navigation for logged-in users on desktop \*/}

            {status?.loggedIn && !isMobile && (

                <motion.div

                    className={classes.verticalNavContainer}

                    initial={{ width: "70px" }}

                    animate={{ width: expandedMenu ? "220px" : "70px" }}

                    transition={{ duration: 0.3, ease: "easeInOut" }}

                    onMouseEnter={() => setExpandedMenu(true)}

                    onMouseLeave={() => setExpandedMenu(false)}

                >

                    <div className={classes.verticalNavItems}>

                        {verticalMenuItems.map((item) => (

                            <motion.div

                                key={item.key}

                                className={classes.verticalNavItem}

                                whileHover={{ backgroundColor: "#f8f8f8" }}

                                onClick={() => handleVerticalMenuClick(item.link)}

                            >

                                <div className={classes.verticalNavIcon}>

                                    {item.icon}

                                </div>

                                {expandedMenu && (

                                    <motion.div

                                        className={classes.verticalNavLabel}

                                        initial={{ opacity: 0, x: -10 }}

                                        animate={{ opacity: 1, x: 0 }}

                                        transition={{ duration: 0.3 }}

                                    >

                                        {item.label}

                                    </motion.div>

                                )}

                            </motion.div>

                        ))}

                        {adminMenuItems.length > 0 && (

                            <>

                                <div className={classes.verticalNavDivider} />

                                {adminMenuItems.map((item) => (

                                    <motion.div

                                        key={item.key}

                                        className={classes.verticalNavItem}

                                        whileHover={{ backgroundColor: "#f8f8f8" }}

                                        onClick={() => handleVerticalMenuClick(item.link)}

                                    >

                                        <div className={classes.verticalNavIcon}>

                                            {item.icon}

                                        </div>

                                        {expandedMenu && (

                                            <motion.div

                                                className={classes.verticalNavLabel}

                                                initial={{ opacity: 0, x: -10 }}

                                                animate={{ opacity: 1, x: 0 }}

                                                transition={{ duration: 0.3 }}

                                            >

                                                {item.label}

                                            </motion.div>

                                        )}

                                    </motion.div>

                                ))}

                            </>

                        )}

                    </div>

                </motion.div>

            )}

            {/\* Burger menu button - only show when appropriate \*/}

            {showBurgerMenu && (

                <motion.div

                    className={classes.navBtn}

                    onClick={() => setOpen(!open)}

                    animate={{ rotate: open ? 180 : 0 }}

                    transition={{ duration: 0.2 }}

                >

                    {open ? <CloseOutlined /> : <MenuOutlined />}

                </motion.div>

            )}

            <motion.div

                className={`${classes.menuContainer} ${transparent ? classes.transparentMenu : ''}`}

                initial={{ y: -20, opacity: 0 }}

                animate={{ y: 0, opacity: 1 }}

                transition={{ duration: 0.3 }}

            >

                <div className={classes.logoContainer}>

                    <Link

                        to="/"

                        onClick={(e) => {

                            if (status?.loggedIn) {

                                e.preventDefault();

                            }

                        }}

                    >

                        <motion.img

                            src="/assets/icon.png"

                            alt="CampusBite Logo"

                            className={classes.logoImg}

                            whileHover={{ rotate: 10, scale: 1.1 }}

                            transition={{ duration: 0.2 }}

                        />

                        <span className={classes.logoText}>

                            Campus<span className={classes.biteText}>Bite</span>

                        </span>

                    </Link>

                </div>

                {/\* Only show navbar on non-mobile screens \*/}

                {!isMobile && !status?.loggedIn && (

                    <nav className={classes.navbar}>

                        <ul>

                            <motion.li whileHover={{ y: -3 }} transition={{ duration: 0.2 }}>

                                <a onClick={() => handleSectionNavigation('home', navigate, setOpen)}>Home</a>

                            </motion.li>

                            <motion.li whileHover={{ y: -3 }} transition={{ duration: 0.2 }}>

                                <a onClick={() => handleSectionNavigation('about', navigate, setOpen)}>About Us</a>

                            </motion.li>

                            <div

                                className={classes.featuresContainer}

                                onMouseEnter={() => setShowFeatures(true)}

                                onMouseLeave={() => setShowFeatures(false)}

                            >

                                <motion.li

                                    whileHover={{ y: -3 }}

                                    transition={{ duration: 0.2 }}

                                    className={classes.featuresMenuItem}

                                >

                                    <a onClick={() => handleSectionNavigation('features', navigate, setOpen)} className={classes.featureMenuLink}>

                                        Features

                                        <DownOutlined className={classes.menuArrowIcon} />

                                    </a>

                                </motion.li>

                            </div>

                            <motion.li whileHover={{ y: -3 }} transition={{ duration: 0.2 }}>

                                <Link to="/schedule">Menu</Link>

                            </motion.li>

                            <motion.li whileHover={{ y: -3 }} transition={{ duration: 0.2 }}>

                                <a onClick={() => handleSectionNavigation('contact', navigate, setOpen)}>Contact</a>

                            </motion.li>

                        </ul>

                    </nav>

                )}

                {/\* Sign in/out button - kept exactly as in old version \*/}

                <div className={classes.signInButton}>

                    <a

                        href={status?.loggedIn ? window.APIROOT + 'api/auth/signout' : window.APIROOT + 'api/auth/signin'}

                        className={classes.signInLink}

                    >

                        {status?.loggedIn ? 'Sign out' : 'Sign in'}

                    </a>

                </div>

            </motion.div>

            {/\* Full screen features mega menu \*/}

            {!isMobile && !status?.loggedIn && (

                <div

                    className={classes.featuresContainer}

                    onMouseEnter={() => setShowFeatures(true)}

                    onMouseLeave={() => setShowFeatures(false)}

                >

                    <AnimatePresence>

                        {showFeatures && (

                            <motion.div

                                className={classes.fullScreenFeatures}

                                initial={{ opacity: 0 }}

                                animate={{ opacity: 1 }}

                                exit={{ opacity: 0 }}

                                transition={{ duration: 0.3 }}

                            >

                                <div className={classes.featuresMegaMenuContent}>

                                    <div className={classes.featuresMegaMenuHeader}>

                                        <h2>Discover Our <span className={classes.highlightText}>Features</span></h2>

                                        <p>Experience the convenience of modern campus dining</p>

                                    </div>

                                    <div className={classes.featureCardsGrid}>

                                        {featureItems.map((feature, index) => (

                                            <motion.div

                                                className={classes.featureCardContainer}

                                                key={index}

                                                initial={{ opacity: 0, y: 20 }}

                                                animate={{

                                                    opacity: 1,

                                                    y: 0,

                                                    transition: { delay: index \* 0.1 }

                                                }}

                                            >

                                                {/\* Card container with perspective \*/}

                                                <div

                                                    className={classes.featureCard}

                                                    onMouseEnter={() => setHoveredCard(index)}

                                                    onMouseLeave={() => setHoveredCard(null)}

                                                    style={{ perspective: "1000px" }}

                                                >

                                                    {/\* Front of card \*/}

                                                    <motion.div

                                                        className={classes.featureCardFront}

                                                        style={{

                                                            borderTop: `4px solid ${feature.color}`,

                                                            boxShadow: `0 10px 20px rgba(0, 0, 0, 0.05)`,

                                                            backfaceVisibility: "hidden",

                                                            position: "absolute",

                                                            width: "100%",

                                                            height: "100%",

                                                            transformStyle: "preserve-3d"

                                                        }}

                                                        animate={{

                                                            rotateY: hoveredCard === index ? 180 : 0

                                                        }}

                                                        transition={{ duration: 0.6, ease: "easeInOut" }}

                                                    >

                                                        <div className={classes.featureIconWrapper} style={{ backgroundColor: `${feature.color}20` }}>

                                                            <i className={`${feature.icon} ${classes.featureIcon}`} style={{ color: feature.color }}></i>

                                                        </div>

                                                        <h3>{feature.title}</h3>

                                                        <p>{feature.frontDescription}</p>

                                                    </motion.div>

                                                    {/\* Back of card \*/}

                                                    <motion.div

                                                        className={classes.featureCardBack}

                                                        style={{

                                                            backgroundColor: feature.color,

                                                            backfaceVisibility: "hidden",

                                                            position: "absolute",

                                                            width: "100%",

                                                            height: "100%",

                                                            transformStyle: "preserve-3d",

                                                            display: "flex",

                                                            flexDirection: "column",

                                                            justifyContent: "center",

                                                            alignItems: "center",

                                                            padding: "1rem",

                                                            color: "#fff",

                                                            textAlign: "center"

                                                        }}

                                                        initial={{ rotateY: 180 }}

                                                        animate={{

                                                            rotateY: hoveredCard === index ? 0 : 180

                                                        }}

                                                        transition={{ duration: 0.6, ease: "easeInOut" }}

                                                    >

                                                        <p style={{ marginBottom: "1rem" }}>{feature.backDescription}</p>

                                                        <a

                                                            onClick={() => handleSectionNavigation('features', navigate, setOpen)}

                                                            className={classes.learnMoreLink}

                                                            // style={{

                                                            //     color: "#fff",

                                                            //     padding: "0.5rem 1rem",

                                                            //     borderRadius: "4px",

                                                            //     border: "2px solid #fff",

                                                            //     textDecoration: "none",

                                                            //     fontWeight: "bold",

                                                            //     transition: "all 0.3s ease",

                                                            //     cursor: "pointer"

                                                            // }}

                                                        >

                                                            {/\* Learn More \*/}

                                                        </a>

                                                    </motion.div>

                                                </div>

                                            </motion.div>

                                        ))}

                                    </div>

                                </div>

                            </motion.div>

                        )}

                    </AnimatePresence>

                </div>

            )}

            {/\* Side menu with controlled visibility based on "open" state only \*/}

            <motion.div

                className={classes.navWrap}

                animate={{ x: open ? 0 : isMobile ? "100%" : "15rem" }}

                initial={{ x: isMobile ? "100%" : "15rem" }}

                transition={{ duration: 0.3 }}

            >

                <Menu

                    className={classes.nav}

                    theme="light"

                    defaultSelectedKeys={['1']}

                    mode="inline"

                    items={items}

                    onClick={() => setOpen(false)}

                />

            </motion.div>

            {/\* Gap for fixed header - adjust for logged in state \*/}

            <div className={`${classes.gap} ${status?.loggedIn && !isMobile ? classes.loggedInGap : ''}`} />

        </>

    );

}

Weekmenu/index.js

import classes from './index.module.css';

import { Table } from 'antd';

export default function WeekMenu(props) {

    const { menu, mobile, highlight, loading } = props;

    const renderTypeCell = (text) => ({

        props: { style: { background: "#FAFAFA" } },

        children: <span style={{ fontWeight: 500, textTransform: "capitalize" }}>{text}</span>,

    });

    const renderMealCell = (text, selected) => ({

        props: highlight

            ? {

                style: {

                    background: selected ? "#ceface" : "unset",

                    color: selected ? "unset" : "lightgrey"

                }

            }

            : {},

/\*\*\*\*\*\*\*\*\*\*\*\*\*  ✨ Windsurf Command ⭐  \*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

 \* Renders a text element with specific styles.

 \*

/\*\*\*\*\*\*\*  9fca56a8-6eb8-4b11-abb3-8e165b2ef31f  \*\*\*\*\*\*\*/        children: <span>{text}</span>,

    });

    if (mobile) {

        return menu.map((e, index) => (

            <Table

                key={e.day || index}

                loading={loading}

                className={classes.table}

                columns={[

                    {

                        title: e.day && e.day.charAt(0).toUpperCase() + e.day.substring(1),

                        dataIndex: 'Type',

                        key: 'Type',

                        colSpan: 2,

                        width: 50,

                        render: (text) => renderTypeCell(text)

                    },

                    {

                        title: '',

                        dataIndex: 'Meal',

                        key: 'Meal',

                        colSpan: 0,

                        width: 300,

                        render: (text, record) => renderMealCell(text, record.selected)

                    },

                ]}

                dataSource={[

                    {

                        key: `${e.day || index}-breakfast`,

                        Type: 'Breakfast',

                        Meal: highlight ? e.breakfast.text : e.breakfast,

                        selected: e.breakfast.selected

                    },

                    {

                        key: `${e.day || index}-lunch`,

                        Type: 'Lunch',

                        Meal: highlight ? e.lunch.text : e.lunch,

                        selected: e.lunch.selected

                    },

                    {

                        key: `${e.day || index}-dinner`,

                        Type: 'Dinner',

                        Meal: highlight ? e.dinner.text : e.dinner,

                        selected: e.dinner.selected

                    },

                ]}

                pagination={false}

                bordered

            />

        ));

    } else {

        return (

            <Table

                loading={loading}

                className={classes.table}

                columns={[

                    {

                        title: 'Day',

                        dataIndex: 'day',

                        key: 'day',

                        render: (text) => renderTypeCell(text)

                    },

                    {

                        title: 'Breakfast',

                        dataIndex: 'breakfast',

                        key: 'breakfast',

                        render: (text, record) => renderMealCell(text, record.selected.breakfast)

                    },

                    {

                        title: 'Lunch',

                        dataIndex: 'lunch',

                        key: 'lunch',

                        render: (text, record) => renderMealCell(text, record.selected.lunch)

                    },

                    {

                        title: 'Dinner',

                        dataIndex: 'dinner',

                        key: 'dinner',

                        render: (text, record) => renderMealCell(text, record.selected.dinner)

                    },

                ]}

                dataSource={menu.map((e, i) => ({

                    key: i,

                    day: e.day,

                    breakfast: highlight ? e.breakfast.text : e.breakfast,

                    lunch: highlight ? e.lunch.text : e.lunch,

                    dinner: highlight ? e.dinner.text : e.dinner,

                    selected: {

                        breakfast: e.breakfast.selected,

                        lunch: e.lunch.selected,

                        dinner: e.dinner.selected

                    },

                }))}

                pagination={false}

                bordered

            />

        );

    }

}