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");

// Step 1: Define the Schema

const menuSchema = new mongoose.Schema({

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

  breakfast: String,

  lunch: String,

  dinner: String

});

// Step 2: Create the model

const Menu = mongoose.model("menuitem", menuSchema);  // ✅ schema passed correctly

// Step 3: Add utility functions

const getMenu = async function () {

  const menuItems = await Menu.find({}).select({ \_id: 0 });

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

  menuItems.sort((a, b) => {

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

  });

  return menuItems;

};

const setMenus = async function (menus) {

  await Menu.deleteMany({});

  await Menu.insertMany(menus);

};

// Step 4: Export everything

module.exports = {

  Menu,      // ✅ Export the actual model

  getMenu,

  setMenus

};

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;

};

Rating.js

const mongoose = require("mongoose");

const mealSchema = new mongoose.Schema({

  mealType:  { type: String, enum: ['breakfast', 'lunch', 'dinner'], required: true },

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

  rating:    { type: Number, required: true, min: 1, max: 5 },

  createdAt: { type: Date, default: Date.now }

});

const ratingDaySchema = new mongoose.Schema({

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

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

  meals: [mealSchema]

});

// Ensure one document per user+day

ratingDaySchema.index({ email: 1, day: 1 }, { unique: true });

module.exports = mongoose.model("Rating", ratingDaySchema);

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)