const mongoose = require("mongoose");

const bcrypt = require("bcryptjs");

const ownerSchema = new mongoose.Schema({

    role: {

        type: String,

        required: true,

        enum: ["Ground Owner", "Team Owner"] // Ensures only valid roles

    },

    yourname: {

        type: String,

        required: true

    },

    email: {

        type: String,

        required: true,

        unique: true, // Ensures no duplicate emails

        lowercase: true, // Normalize email

        trim: true

    },

    password: {

        type: String,

        required: true

    },

    phonenumber: {

        type: String,

        required: true,

        validate: {

            validator: function (v) {

                return /^\d{10}$/.test(v); // Only allows 10-digit phone numbers

            },

            message: props => `${props.value} is not a valid phone number!`

        }

    },

    groundname: {

        type: String,

        required: function () { return this.role === "Ground Owner"; }, // Only for Ground Owner

        default: null

    },

    groundlocation: {

        type: String,

        required: function () { return this.role === "Ground Owner"; }, // Only for Ground Owner

        default: null

    },

    teamname: {

        type: String,

        required: function () { return this.role === "Team Owner"; }, // Only for Team Owner

        default: null

    }

});

// \*\*🔒 Hash Password Before Saving\*\*

ownerSchema.pre("save", async function (next) {

    if (!this.isModified("password")) return next();

    try {

        const salt = await bcrypt.genSalt(10);

        this.password = await bcrypt.hash(this.password, salt);

        next();

    } catch (err) {

        next(err);

    }

});

// Create a collection

const Signup = mongoose.model("Signup", ownerSchema);

module.exports = Signup;