import { DataTypes } from 'sequelize';

import bcrypt from 'bcrypt';

import sequelize from '../models/connection.js';

const Student = sequelize.define('Student', {

    id: {

        type: DataTypes.INTEGER,

        primaryKey: true,

        autoIncrement: true

    },

    name: {

        type: DataTypes.STRING,

        allowNull: false

    },

    email: {

        type: DataTypes.STRING,

        allowNull: false,

        unique: true,

        validate: {

            isEmail: true,

            noSpaces(value) {

                if (/\s/.test(value)) {

                    throw new Error('Email must not contain spaces.');

                }

            }

        }

    },

    age: {

        type: DataTypes.INTEGER,

        allowNull: false,

        validate: {

            min: 0,

            max: 200

        }

    },

    grade: {

        type: DataTypes.STRING,

        allowNull: true,

        validate: {

            len: [1, 2]

        }

    },

    password: {

        type: DataTypes.STRING,

        allowNull: false,

        validate: {

            isStrongPassword(value) {

                if (!isStrongPassword(value)) {

                    throw new Error('Password must be 6-15 characters long, contain at least one uppercase letter, one lowercase letter, one number, and one special character.');

                }

            }

        }

    },

    resetPasswordToken: {

        type: DataTypes.STRING,

        allowNull: true

    },

    resetPasswordExpires: {

        type: DataTypes.DATE,

        allowNull: true

    }

}, {

    tableName: 'students',

    timestamps: true,

    hooks: {

        beforeCreate: async (student) => {

            if (student.password) {

                const salt = await bcrypt.genSalt(10);

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

            }

        },

        beforeUpdate: async (student) => {

            if (student.changed('password')) {

                const salt = await bcrypt.genSalt(10);

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

            }

        }

    }

});

function isStrongPassword(password) {

    const passwordRegex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[\W\_]).{6,15}$/;

    return passwordRegex.test(password);

}

Student.prototype.validatePassword = async function (password) {

    return await bcrypt.compare(password, this.password);

};

export default Student;