const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const OrganizationSchema = new Schema({

  userId: { type: mongoose.Schema.Types.ObjectId, ref: "User" },

  levelId: { type: mongoose.Schema.Types.ObjectId, ref: "OrganizationLevel" }, // Từ level check xem được đăng nhiêu bài

  name: { type: String },

  description: { type: String },

  phone: {type: String},

  adminStatus: {

    type: String,

    default: "pending",

    enum: ["approved", "pending"],

  }, // Trạng thái admin duyệt -> reject thì xóa luôn

  organizationStatus: {

    type: String,

    default: "inactive",

    enum: ["active", "inactive"],

  },

  members: [{ type: Schema.ObjectId, ref: "User" }],

  createdAt: {type: Date},

  licenseDocuments: [{type: String, default: []}],

  contractDocuments: [{type: String, default: []}],

  rating: {type: Number},

  subscribedAt: { type: Date }, // Ngày nâng cấp gói

  expiresAt: { type: Date }, // null nếu basic

});

module.exports = mongoose.model('Organization', OrganizationSchema);

const mongoose = require('mongoose')

const Schema = mongoose.Schema

const OrganizationLevelSchema = new Schema({

   name: { type: String, required: true, enum: ['basic', 'premium'], default: 'basic'}, // Basic, Pro, etc.

   pricePerMonth: { type: Number, default: 0 }, // VNĐ

   description: { type: String }, // mô tả gói

});

module.exports = mongoose.model('OrganizationLevel', OrganizationLevelSchema);

import mongoose from "mongoose";

const EventSchema = new mongoose.Schema({

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

  description: { type: String },

  requirement: { type: String },

  images: [{ type: String }], // URL ảnh của event

  category: { type: String },

  rating: { type: Number },

  currentParticipants: { type: Number, default: 0 },

  pointReward: { type: Number, default: 0 }, // điểm cộng khi hoàn thành

  assignChecker: {type: mongoose.Schema.ObjectId, ref:'User'},

  startAt: { type: Date, required: true },

  endAt: { type: Date, required: true },

  organizationId: {

    type: mongoose.Schema.Types.ObjectId,

    ref: "Organization",

    required: true,

  },

  skillNeeds: [

    {type: String}

  ],

  location: {

    street: { type: String, default: null }, // Số nhà, tên đường (tùy chọn)

    ward: { type: String, default: null }, // Phường/Xã

    district: { type: String, default: null }, // Quận/Huyện

    province: { type: String, default: null }, // Tỉnh/Thành phố

    country: { type: String, default: "VietNam" },

    postalCode: String, // (tùy chọn) Mã bưu chính nếu cần

    fullAddress: { type: String },

  },

  formData: {

    questions: [

      {

        question: { type: String },

        type: { type: String, enum: ["text", "checkbox", "radio", "dropdown"] },

        options: [{ type: String, default: [] }], // nếu applicable

      },

    ],

  },

  status: {

    type: String,

    enum: ["hiring", "prccessing", "completed", "cancelled", "full"], // full là đủ người - waiting là đang tuyển

    default: "hiring",

  },

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

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

  // Những user đã hoàn thành

  completedBy: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],

  // Những user đã điểm danh

  attendedUsers: [

    {

      attendanceId: { type: mongoose.Schema.Types.ObjectId, ref: "Attendance" },

    },

  ],

  // Đánh giá sự kiện

  feedbacks: [

    {

      feedbackId: { type: mongoose.Schema.Types.ObjectId, ref: "Feedback" },

    },

  ],

  // Lọc theo

});

export default mongoose.model("Event", EventSchema);

const mongoose = require('mongoose');

const CertificateSchema = new mongoose.Schema({

  // Liên kết

  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },

  eventId: { type: mongoose.Schema.Types.ObjectId, ref: 'Event', required: true },

  // Thông tin hiển thị

  fullName: { type: String, required: true },               // Tên người nhận

  email: { type: String, required: true },                  // Email để gửi chứng chỉ (nếu có)

  eventName: { type: String, required: true },              // Tên sự kiện

  organizationName: { type: String, required: true },       // Tên tổ chức cấp chứng chỉ

  completionDate: { type: Date, required: true },           // Ngày hoàn thành sự kiện

  duration: { type: Number, required: true },               // Tổng số giờ tham gia

  // Uy tín và nhận dạng

  certificateCode: { type: String, unique: true, required: true }, // Mã định danh duy nhất

  signature: { type: String, required: true },              // Hash dựa trên org + event

  // Trạng thái và truy xuất

  isPurchased: { type: Boolean, default: false },           // Có mua hay được cấp miễn phí

  price: { type: Number, default: 30000 },                  // Nếu trả phí thì bao nhiêu

  certificateUrl: { type: String },                         // Link PDF nếu có

});

module.exports = mongoose.model('Certificate', CertificateSchema);

const mongoose = require('mongoose')

const Schema = mongoose.Schema;

const FeedbackSchema = new Schema({

   userId: {type: Schema.ObjectId, ref: 'User'},

   eventId: {type: Schema.ObjectId, ref: 'Event'},

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

   content: {type: String, default: null},

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

   like: {type: Number, default: 0}

})

module.exports = mongoose.model('Feedback', FeedbackSchema);

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const historyEventSchema = new Schema({

  user: {

    type: mongoose.Schema.Types.ObjectId,

    ref: "User",

    required: true,

  },

  event: {

    type: mongoose.Schema.Types.ObjectId,

    ref: "Event",

    required: true,

  },

  status: {

    type: String,

    enum: [

      "completed", //duoc phep feedback

      "finished",  //sau khi feedback

      "waiting",

      "processing",

      "approved",

      "pending",

      "cancelled",

    ],

    default: "waiting",

  },

  registeredAt: {

    type: Date,

    default: Date.now,

  },

  attendedAt: {

    type: Date,

  },

  completedAt: {

    type: Date,

    default: null,

  },

  earnedPoints: {

    type: Number,

    default: 0,

  },

  earnedHours: {

    type: Number,

    default: 0,

  },

  certificateId: {

    type: mongoose.Schema.Types.ObjectId,

    ref: "Certificate",

    default: null,

  },

});

module.exports = mongoose.model("HistoryEvent", historyEventSchema);

const mongoose = require('mongoose')

const Schema = mongoose.Schema

const AnswerSchema = new Schema({

  question: { type: String, required: true }, // Câu hỏi đúng theo formData

  type: {

    type: String,

    enum: ["text", "checkbox", "radio", "dropdown"],

    required: true

  },

  answer: {

    type: mongoose.Schema.Types.Mixed, // string hoặc array<string>

    required: true

  }

}, { \_id: false });

const EventRegistrationSchema = new Schema({

   event: { type: mongoose.Schema.Types.ObjectId, ref: "Event", required: true },

   user: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

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

   status: {

     type: String,

     enum: ["pending", "approved", "rejected", "cancelled"],

     default: "pending"

   },

   answers: [AnswerSchema],

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

   cancellationReason: { type: String },

   approvedBy: { type: mongoose.Schema.Types.ObjectId, ref: "Admin" },

   approvedAt: { type: Date }

 });

 module.exports = mongoose.model("EventRegistration", EventRegistrationSchema);

const mongoose = require('mongoose')

const Schema = mongoose.Schema

const AttendanceSchema = new Schema({

   eventId: {

     type: mongoose.Schema.Types.ObjectId,

     ref: 'Event',

     required: true,

   },

   userId: {

     type: mongoose.Schema.Types.ObjectId,

     ref: 'User',

     required: true,

   },

   status: {

     type: String,

     enum: ["registered", "attended", "cancelled"],

     default: "registered"

   },

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

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

   createdAt: {

     type: Date,

     default: Date.now

   }

});

module.exports = mongoose.model('Attendance', AttendanceSchema);

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

// Kiểm tra ngày hết hạn auto là 1 tháng kể từ ngày tạo

const OrganizationSubscription = new Schema({

  \_id: {type: Schema.ObjectId, ref: 'User'},

  organizationId: {type: Schema.ObjectId, ref: 'Organization'},

  planId: {type: Schema.ObjectId, ref: 'OrganizationLevel'},

  price: {type: Number},

  subscribedAt: {type: Date, default: new Date.now}, // Ngày bắt đầu dùng gói

  expiredAt:  {type: Date, default: new Date.now}, // Ngày hết hạn

  status: {type: String, default: 'not paid', enum: ["paid"]},

});

module.exports = mongoose.model(

  "OrganizationSubscription",

  OrganizationSubscription

);