User.js

const UserSchema = new Schema(

{

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

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

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

createOn: { type: Date, default: new Date().getTime() },

cmnd: { type: String, require: true },

updatedAt: { type: Date, default: new Date().getTime() },

role: { type: String, enum: ["CUSTOMER", "ADMIN", "OWNER"], default: "CUSTOMER" },

reservations: [

{ type: Schema.Types.ObjectId, ref: "Reservation" },

],

ownedHotels: [

{

type: Schema.Types.ObjectId,

ref: "Hotel",

},

], // Mảng các khách sạn người dùng sở hữu (tham chiếu đến Hotel),

favorites: [

{

type: Schema.Types.ObjectId,

ref: "Hotel",

},

], // Mảng các khách sạn yêu thích (tham chiếu đến Hotel)

},

{ versionKey: false }

);

// Schema cho Hotel

const hotelSchema = new Schema(

{

hotelName: {

type: String,

required: true,

},

owner: {

type: Schema.Types.ObjectId,

ref: "User",

},

description: {

type: String,

required: true,

},

address: {

type: String,

required: true,

},

services: [

{

type: Schema.Types.ObjectId,

ref: "HotelService",

},

],

facility: [

{

type: Schema.Types.ObjectId,

ref: "Facility",

},

],

rating: {

type: Number,

required: true,

},

pricePerNight: {

type: Number,

required: true,

},

images: [

{

type: String,

required: true

},

],

},

{ versionKey: false }

);

module.exports = mongoose.model("Hotel", hotelSchema);

const RoomSchema = new Schema({

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

price: { type: Number, required: true },

capacity: {type: Number, required: true},

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

images: [{ type: String, required: true}],

//References

hotel: { type: Schema.Types.ObjectId, ref: 'Hotel', required: true },

bed: [{

bed: { type: Schema.Types.ObjectId, ref: 'Bed', required: true },

quantity: { type: Number, required: true }

}],

facilities: [{ type: Schema.Types.ObjectId, ref: 'RoomFacility'}],

}, {versionKey: false});

module.exports = mongoose.model('Room', RoomSchema);

const RoomFacilitySchema = new Schema({

room: { type: mongoose.Schema.Types.ObjectId, ref: 'Room', required: true },

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

description: { type: String },

url: {type: Buffer, required: true},

}, {versionKey: false});

module.exports = mongoose.model('RoomFacility', RoomFacilitySchema);

const reservationSchema = new Schema(

{

user: {

type: mongoose.Schema.Types.ObjectId,

ref: "User",

required: true,

},

hotel: {

type: mongoose.Schema.Types.ObjectId,

ref: "Hotel",

required: true,

},

rooms: [

{

type: mongoose.Schema.Types.ObjectId,

ref: "Room",

required: true,

},

], // Danh sách phòng được đặt

checkInDate: {

type: Date,

required: true,

},

checkOutDate: {

type: Date,

required: true,

},

status: {

type: String,

enum: [

"BOOKED", // Đã đặt, trả tiền nhưng chưa check-in

"CHECKED IN", // Đang ở, đã check-in

"CHECKED OUT", // Đã check-out, có thể để lại phản hồi

"COMPLETED", // Hoàn thành, đã phản hồi

"PENDING", // Chờ xử lý hoặc xác nhận

"CANCELLED", // Đã hủy

"NOT PAID" // Chưa trả tiền

],

default: "PENDING", // Mặc định là chờ xử lý

},

totalPrice: {

type: Number,

required: true,

default: 0

},

},

{ timestamps: true },

{versionKey: false}

);

module.exports = mongoose.model("Reservation", reservationSchema);

const HotelServiceSchema = new Schema({

hotel: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Hotel',

required: true

}, // Tham chiếu đến mô hình Hotel

name: {

type: String,

required: true,

},

price: {

type: Number,

required: true

} // Giá của dịch vụ tại khách sạn này

}, {versionKey: false});

module.exports = mongoose.model('HotelService', HotelServiceSchema);

const FacilitySchema = new Schema({

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

description: { type: String },

url: {type: Buffer, required: true},

hotel: [

{

type: Schema.Types.ObjectId,

ref: "Hotel",

},

], // Mảng các khách sạn người dùng sở hữu (tham chiếu đến Hotel),

}, {versionKey: false});

module.exports = mongoose.model('Facility', FacilitySchema);

const bedSchema = new mongoose.Schema({

name: {

type: String,

required: true,

},

description: {

type: String,

required: true,

},

}, {versionKey: false});

module.exports = mongoose.model("Bed", bedSchema);

const FeedbackSchema = new Schema({

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }, //User ID

reservation: {type: Schema.Types.ObjectId, ref: 'Reservation', required: true}, //Reservation ID

hotel: { type: mongoose.Schema.Types.ObjectId, ref: 'Hotel', required: true }, //Hotel ID

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

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

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

}, {versionKey: false});

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

**SCRIPT INSERT DATABASE**

// 1. Insert Users

let users = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

name: `User ${i + 1}`,

email: `user${i + 1}@example.com`,

password: "hashedpassword",

createdOn: new Date(),

cmnd: `CMND${i + 1000}`,

updatedAt: new Date(),

role: ["CUSTOMER", "ADMIN", "OWNER"][i % 3],

reservations: [],

ownedHotels: [],

favorites: []

}));

db.users.insertMany(users);

// 2. Insert Hotels

let hotels = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

hotelName: `Hotel ${i + 1}`,

owner: users[i % users.length].\_id, // Liên kết với 1 user làm chủ

description: `A luxurious hotel number ${i + 1}`,

address: `Address ${i + 1}`,

services: [],

facility: [],

rating: (Math.random() \* 5).toFixed(1),

pricePerNight: Math.floor(Math.random() \* 500) + 50,

images: ["image1.jpg", "image2.jpg"]

}));

db.hotels.insertMany(hotels);

// 3. Insert Rooms

let rooms = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

type: ["Single", "Double", "Suite"][i % 3],

price: Math.floor(Math.random() \* 300) + 50,

capacity: (i % 4) + 1,

description: `Room description ${i + 1}`,

images: ["room1.jpg", "room2.jpg"],

hotel: hotels[i % hotels.length].\_id, // Liên kết với 1 hotel

bed: [], // Để cập nhật sau

facilities: []

}));

db.rooms.insertMany(rooms);

// 4. Insert Reservations

let reservations = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

user: users[i % users.length].\_id, // Liên kết với 1 user

hotel: hotels[i % hotels.length].\_id, // Liên kết với 1 hotel

rooms: [rooms[i % rooms.length].\_id], // Liên kết với 1 room

checkInDate: new Date(),

checkOutDate: new Date(new Date().setDate(new Date().getDate() + 3)),

status: ["BOOKED", "CHECKED IN", "CHECKED OUT", "COMPLETED", "PENDING", "CANCELLED", "NOT PAID"][i % 7],

totalPrice: Math.floor(Math.random() \* 2000) + 100

}));

db.reservations.insertMany(reservations);

// 5. Insert Hotel Services

let hotelServices = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

hotel: hotels[i % hotels.length].\_id, // Liên kết với 1 hotel

name: `Service ${i + 1}`,

price: Math.floor(Math.random() \* 100) + 10

}));

db.hotelservices.insertMany(hotelServices);

// 6. Insert Facilities

let facilities = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

name: `Facility ${i + 1}`,

description: `Description for facility ${i + 1}`,

url: BinData(0, ""),

hotel: [hotels[i % hotels.length].\_id] // Liên kết với 1 hotel

}));

db.facilities.insertMany(facilities);

// 7. Insert Beds

let beds = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

name: `Bed Type ${i + 1}`,

description: `Description for bed ${i + 1}`

}));

db.beds.insertMany(beds);

// Cập nhật Room để liên kết với Beds

rooms.forEach((room, i) => {

db.rooms.updateOne(

{ \_id: room.\_id },

{ $set: { bed: [{ bed: beds[i % beds.length].\_id, quantity: (i % 2) + 1 }] } }

);

});

// 8. Insert Feedbacks

let feedbacks = [...Array(50)].map((\_, i) => ({

\_id: ObjectId(),

user: users[i % users.length].\_id, // Liên kết với 1 user

reservation: reservations[i % reservations.length].\_id, // Liên kết với 1 reservation

hotel: hotels[i % hotels.length].\_id, // Liên kết với 1 hotel

content: `Feedback content ${i + 1}`,

rating: Math.floor(Math.random() \* 5) + 1,

createdAt: new Date()

}));

db.feedbacks.insertMany(feedbacks);