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}],

quantity: {type: Number, 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**

const userIds = [

ObjectId("67aa0d68c772ce56e1cbbd08"),

ObjectId("67aa0d7ec772ce56e1cbbd0e"),

ObjectId("67aa0d99c772ce56e1cbbd13")

];

const hotelIds = [];

const roomIds = [];

const facilityIds = [];

const bedIds = [];

const reservationIds = [];

// Insert 20 hotels

for (let i = 1; i <= 20; i++) {

let hotel = db.hotels.insertOne({

hotelName: `Hotel ${i}`,

owner: userIds[i % 3],

description: `Description for Hotel ${i}`,

address: `Address ${i}`,

services: [],

facility: [],

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

pricePerNight: Math.floor(Math.random() \* 200) + 50,

images: [`https://picsum.photos/600/400?random=${i}`]

});

hotelIds.push(hotel.insertedId);

}

// Insert 20 beds

for (let i = 1; i <= 20; i++) {

let bed = db.beds.insertOne({

name: `Bed ${i}`,

description: `Comfortable bed ${i}`,

image: `https://picsum.photos/250/150?random=${i}`

});

bedIds.push(bed.insertedId);

}

// Insert 60 rooms (mỗi khách sạn có 3 phòng, mỗi phòng có 3 loại giường khác nhau)

for (let i = 0; i < 20; i++) { // 20 khách sạn

for (let j = 0; j < 3; j++) { // 3 phòng mỗi khách sạn

let selectedBeds = [

{ bed: bedIds[(i \* 3 + j) % 20], quantity: Math.floor(Math.random() \* 3) + 1 },

{ bed: bedIds[(i \* 3 + j + 1) % 20], quantity: Math.floor(Math.random() \* 3) + 1 },

{ bed: bedIds[(i \* 3 + j + 2) % 20], quantity: Math.floor(Math.random() \* 3) + 1 }

];

let room = db.rooms.insertOne({

type: `Room Type ${j + 1} of Hotel ${i + 1}`,

price: Math.floor(Math.random() \* 200) + 50,

capacity: Math.floor(Math.random() \* 4) + 1,

description: `Room ${j + 1} of Hotel ${i + 1}`,

images: [`https://picsum.photos/400/300?random=${i \* 3 + j}`],

quantity: Math.floor(Math.random() \* 10) + 1,

hotel: hotelIds[i],

beds: selectedBeds,

facilities: []

});

roomIds.push(room.insertedId);

}

}

// Insert 20 facilities

for (let i = 1; i <= 20; i++) {

let facility = db.facilities.insertOne({

name: `Facility ${i}`,

description: `Description for Facility ${i}`,

url: `https://picsum.photos/300/200?random=${i}`,

hotel: [hotelIds[i % 20]]

});

facilityIds.push(facility.insertedId);

}

// Gán facility cho hotels

hotelIds.forEach((hotelId, index) => {

db.hotels.updateOne(

{ \_id: hotelId },

{ $push: { facility: facilityIds[index % 20] } }

);

});

// Insert 20 hotel services

for (let i = 1; i <= 20; i++) {

db.hotelservices.insertOne({

hotel: hotelIds[i % 20],

name: `Service ${i}`,

price: Math.floor(Math.random() \* 100) + 20

});

}

// Insert 20 room facilities

for (let i = 1; i <= 20; i++) {

db.roomfacilities.insertOne({

room: roomIds[i % 60],

name: `Room Facility ${i}`,

description: `Facility ${i} for rooms`,

url: `https://picsum.photos/300/200?random=${i}`

});

}

// Insert 20 reservations

for (let i = 1; i <= 20; i++) {

let reservation = db.reservations.insertOne({

user: userIds[i % 3],

hotel: hotelIds[i % 20],

rooms: [roomIds[i % 60]],

checkInDate: new Date(),

checkOutDate: new Date(new Date().getTime() + 1000 \* 60 \* 60 \* 24 \* (i % 7 + 1)),

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

totalPrice: Math.floor(Math.random() \* 500) + 100

});

reservationIds.push(reservation.insertedId);

}

// Insert 20 feedbacks

for (let i = 1; i <= 20; i++) {

db.feedbacks.insertOne({

user: userIds[i % 3],

reservation: reservationIds[i % 20],

hotel: hotelIds[i % 20],

content: `Feedback ${i} for Hotel ${i}`,

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

createdAt: new Date()

});

}

print("Insertion completed!");