***EXERCISE 2***🡪 <https://mygust.com/uploads/BOOK-Systems_analysis_and_design_in_a_changin.pdf> , <https://mygust.com/wp-content/uploads/2014/12/Solutions-Chap03.pdf>

**PROMPT:** The following should be textually analyzed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please give me the PlantUML code for the use case diagram corresponding to the following text:

***One of the functions of the Board of Realtors is to provide a Multiple Listing Service (MLS) system that supplies information that local real estate agents use to help them sell houses to their customers. During the month, agents list houses for sale (listings) by contracting with homeowners. The agent works for a real estate office, which sends information on the listing to the MLS. Therefore, any agent in the community can get information on the listing. Information on a listing includes the address, year built, square feet, number of bedrooms, number of bathrooms, owner name, owner phone number, asking price, and status code. At any time during the month, an agent might directly request information on listings that match customer requirements, so the agent contacts the MLS with the request. Information is provided on the house, on the agent who listed the house, and on the real estate office for which the agent works. For example, an agent might want to call the listing agent to ask additional questions or call the homeowner directly to make an appointment to show the house. Once each week, the MLS produces a listing book that contains information on all listings. These books are sent to some real estate agents. Some real estate agents want the books (which are easier to flip through), so they are provided even though the information is often out of date. Sometimes, agents and owners decide to change information about a listing, such as reducing the price, correcting previous information on the house, or indicating that the house is sold. The real estate office sends in these change requests to the MLS when the agent asks the office to do so.***

ChatGPT

**ChatGPT**

**Actors:**

1. Real Estate Agent
2. Homeowner
3. Multiple Listing Service (MLS)
4. Real Estate Office

**Use Cases:**

1. List House for Sale (by Real Estate Agent)
2. Request Listing Information (by Real Estate Agent)
3. Produce Weekly Listing Book (by MLS)
4. Change Listing Information (by Real Estate Agent)

**Associations:**

1. Real Estate Agent lists houses for sale by contracting with homeowners.
2. Real Estate Agent sends information on the listing to the MLS.
3. Any agent can get information on a listing from the MLS.
4. Real Estate Agent requests information on listings that match customer requirements from the MLS.
5. MLS provides information on the house, the listing agent, and the real estate office.
6. MLS produces a weekly listing book containing information on all listings.
7. Some real estate agents receive the weekly listing book.
8. Real Estate Agent can request changes to listing information from the Real Estate Office.
9. Real Estate Office sends change requests to the MLS.

Now, let's create the PlantUML code for the use case diagram:

@startuml

left to right direction

actor "Real Estate Agent" as Agent

actor Homeowner

actor "Multiple Listing Service (MLS)" as MLS

actor "Real Estate Office" as Office

rectangle "System" {

usecase "List House for Sale" as UC1

usecase "Request Listing Information" as UC2

usecase "Produce Weekly Listing Book" as UC3

usecase "Change Listing Information" as UC4

}

Agent --> UC1

Agent --> UC2

MLS --> UC2

MLS --> UC3

Agent --> UC4

Office --> UC4

@enduml

**PROMPT:** I think that you are missing the "management" actor and I also think that you are missing too many use cases related to the actor "real estate officer". The PlantUML code you provided me is too basic. I really need much more detailed code. Analyze the whole text again and provide me a better code please

ChatGPT

**ChatGPT**

I apologize for the oversight. Let's analyze the text again and provide a more detailed use case diagram with the necessary actors and use cases.

**Actors:**

1. Real Estate Agent
2. Homeowner
3. Multiple Listing Service (MLS)
4. Real Estate Office
5. Management

**Use Cases:**

1. List House for Sale (by Real Estate Agent)
2. Request Listing Information (by Real Estate Agent)
3. Produce Weekly Listing Book (by MLS)
4. Receive Weekly Listing Book (by Real Estate Agent)
5. Change Listing Information (by Real Estate Agent)
6. Request Change to Listing Information (by Real Estate Office)
7. Approve Change Request (by Management)
8. Update Listing Information (by Real Estate Office)

**Associations:**

1. Real Estate Agent lists houses for sale by contracting with homeowners.
2. Real Estate Agent sends information on the listing to the MLS.
3. Any agent can get information on a listing from the MLS.
4. Real Estate Agent requests information on listings that match customer requirements from the MLS.
5. MLS provides information on the house, the listing agent, and the real estate office.
6. MLS produces a weekly listing book containing information on all listings.
7. Some real estate agents receive the weekly listing book.
8. Real Estate Agent can request changes to listing information from the Real Estate Office.
9. Real Estate Office sends change requests to the MLS.
10. Real Estate Office requests changes to listing information.
11. Management approves change requests.
12. Real Estate Office updates listing information.

Now, let's create the PlantUML code for the improved use case diagram:

@startuml

left to right direction

actor "Real Estate Agent" as Agent

actor Homeowner

actor "Multiple Listing Service (MLS)" as MLS

actor "Real Estate Office" as Office

actor Management

rectangle "System" {

usecase "List House for Sale" as UC1

usecase "Request Listing Information" as UC2

usecase "Produce Weekly Listing Book" as UC3

usecase "Receive Weekly Listing Book" as UC4

usecase "Change Listing Information" as UC5

usecase "Request Change to Listing Information" as UC6

usecase "Approve Change Request" as UC7

usecase "Update Listing Information" as UC8

}

Agent --> UC1

Agent --> UC2

MLS --> UC2

MLS --> UC3

Agent --> UC4

Agent --> UC5

Office --> UC5

Office --> UC6

Management --> UC7

Office --> UC8

@enduml

**PROMPT:** What about if you consider just 3 actors: realestate office, agent, and management? Give me a new code

ChatGPT

**ChatGPT**

Certainly, if we simplify the actors to just "Real Estate Office," "Agent," and "Management," and focus on the core interactions, the PlantUML code would look like this:

@startuml

left to right direction

actor "Real Estate Office" as Office

actor "Real Estate Agent" as Agent

actor Management

rectangle "System" {

usecase "List House for Sale" as UC1

usecase "Request Listing Information" as UC2

usecase "Produce Weekly Listing Book" as UC3

usecase "Receive Weekly Listing Book" as UC4

usecase "Change Listing Information" as UC5

usecase "Request Change to Listing Information" as UC6

usecase "Approve Change Request" as UC7

usecase "Update Listing Information" as UC8

}

Office --> UC1

Agent --> UC2

Office --> UC2

Office --> UC3

Agent --> UC4

Agent --> UC5

Office --> UC5

Office --> UC6

Management --> UC7

Office --> UC8

@enduml

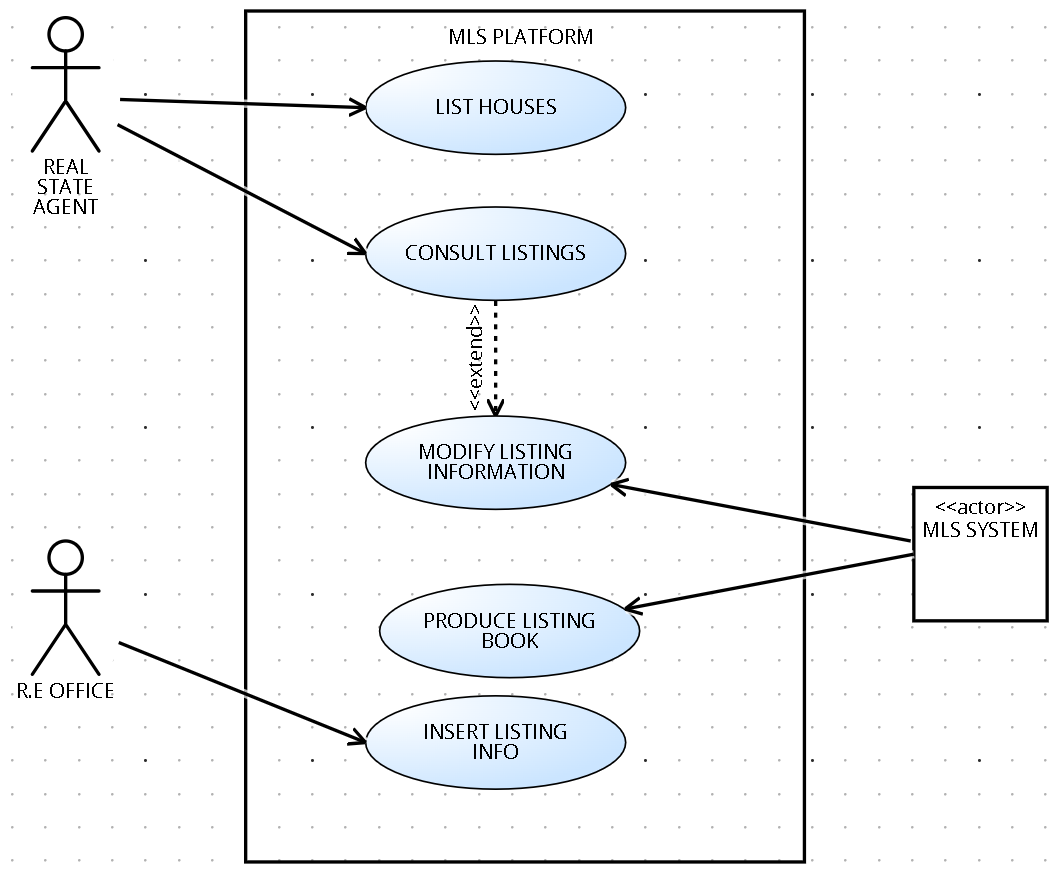
**Analysis:**

This exercise offers an abundance of context and explanations regarding interactions with the system. Unfortunately, ChatGPT did not manage to provide a solution as close to the original one this time. Initially, the generated output code was overly summarized and significantly different from the actual answer. Notably, two new actors emerged: the Multiple Listing Service (MLS) and the Homeowner. The MLS is deemed a necessary actor, omitted in the original answer, as the text clearly delineates real interactions between the MLS and the system. However, the Homeowner is considered unnecessary for this diagram and could have been omitted.

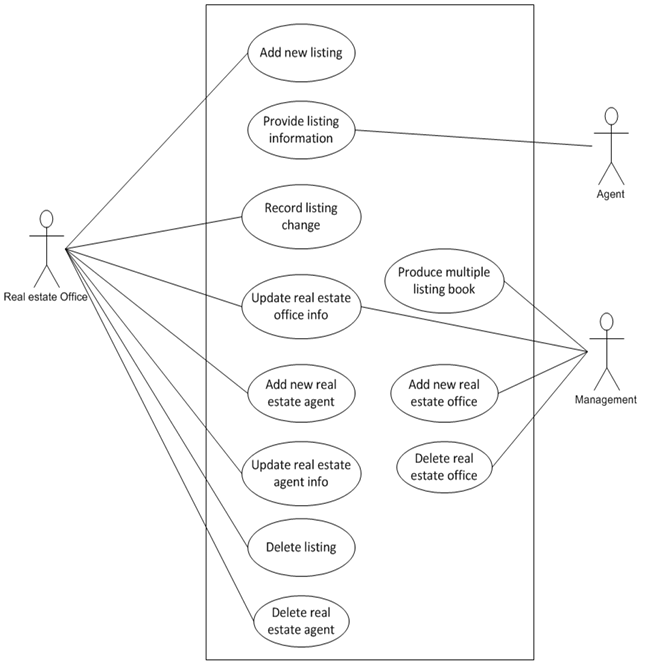
Furthermore, the second code provided was more detailed and specific but still fell short of being comparable to the original. This discrepancy arose due to the inclusion of use cases associated with new actors not present in the original solution. Although the second code managed to generate a Use Case Diagram (UCD) that was essentially correct and understandable in comparison to the text description, I wanted to explore what ChatGPT could produce if we instructed it to consider the same actors as in the original answer. Remarkably, the third solution closely resembled the one given in the exercise, with a notable error—identifying "receive weekly listing book" as an agent's goal, which it is not. This constitutes a clear error in the model's understanding and underscores a limitation when processing extensive descriptive contexts.

An essential point in the comparison between the original solution and ChatGPT's output is that the actor "management" went unidentified initially. Despite my request to include this actor, the model couldn't generate the associated use cases. While this stands as a limitation, it is a minor one, as even as a reader, it wasn't easy for me to identify this actor in the text. At this juncture, it becomes evident that when the textual description lacks explicitness, ChatGPT struggles to identify and incorporate these elements into the code.

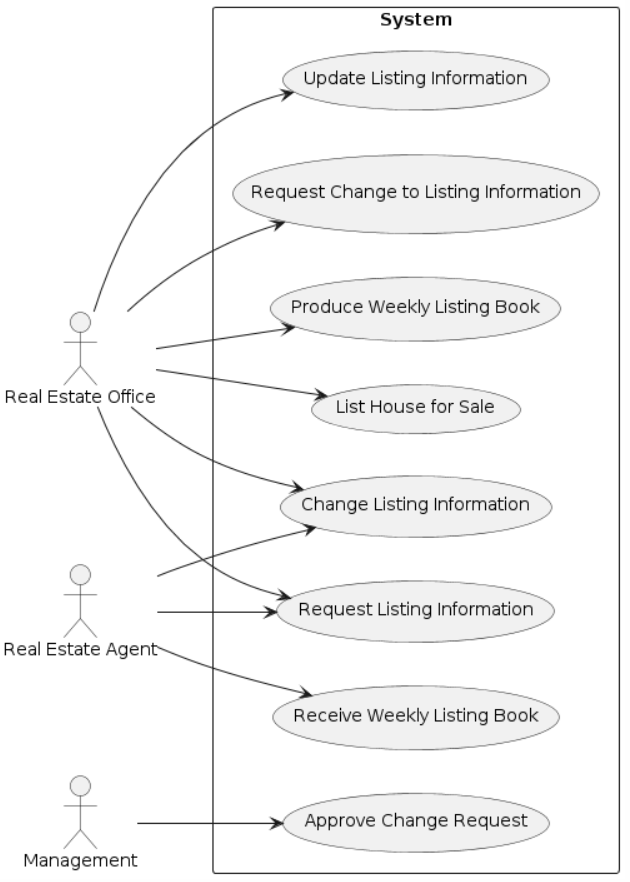
**MY SOLUTION:**

****

**REFERENCE SOLUTION:**

****

**LLM SOLUTION:**

****