# Introduction

Gathering requirement is the crucial phase of any software development as this is the phase where we will know the purpose of the application. How application final output should be and other. If requirement is not clear it might create problem during implementation phase. As I am following waterfall model it’s very important to clarify the requirement as it is hard to reverse the process once it is done in case of waterfall model. In this phase I will also prioritize the requirements that I have identified. I will be creating different Diagrams which have their own purpose of making development work run smoothly.

# Use case Diagram

Use case diagram helps to represent the action that will be performed by different **actors**. Action performed by the actor is shown is **use cases.** Actor can be user, customers etc. it is based on the requirement of the system.

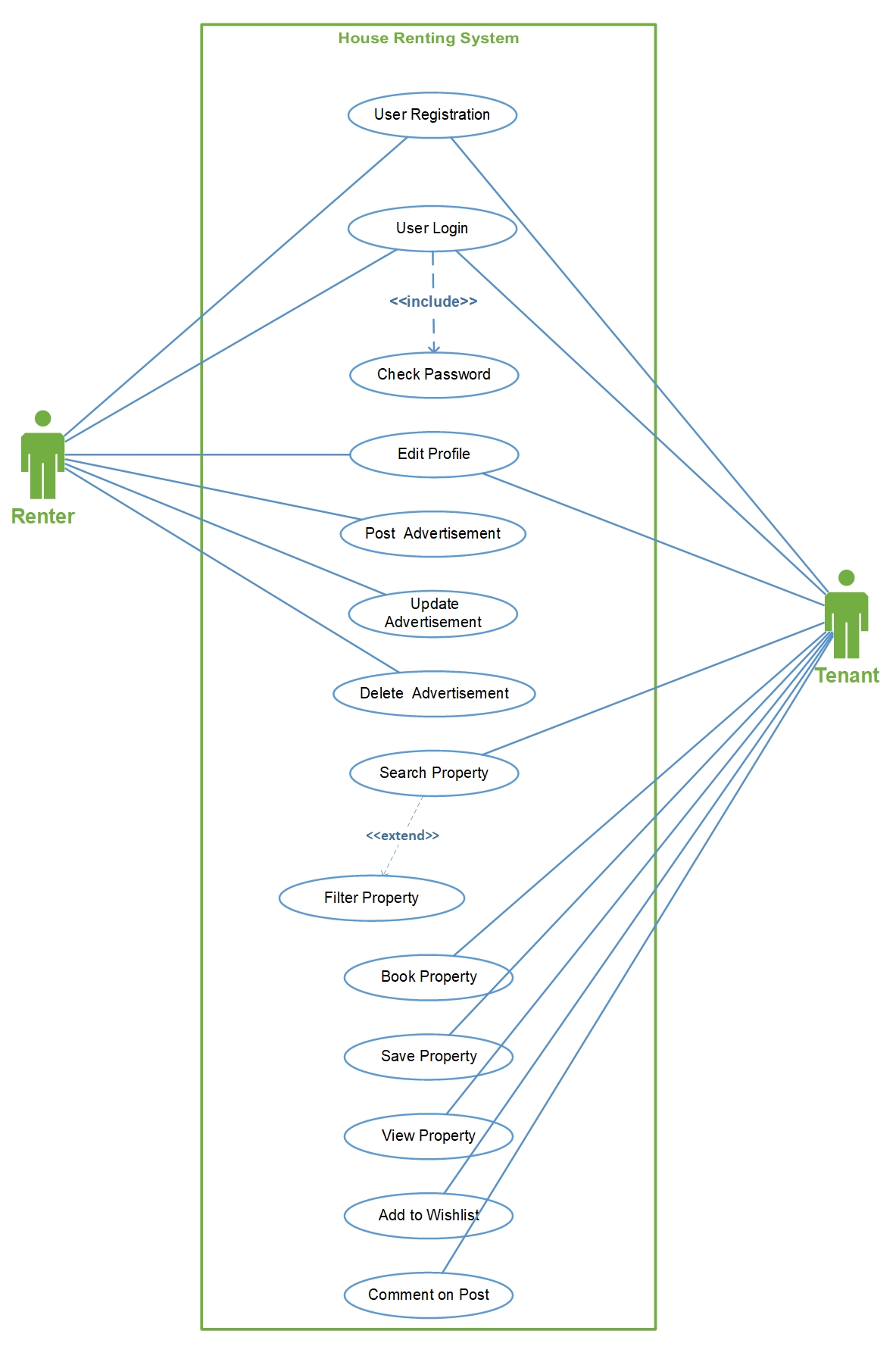


Figure HRS\_Use\_Case\_Diagram

# Requirements

## Functional Requirement

Functional requirement specifies the behaviors or function. Any requirements that specifies something the system should do is functional requirements.

Typical example of functional requirement is:

* Business rules
* Transaction corrections
* Administrative Functions
* Authentication etc.

## Non-functional Requirement

Non-Functional requirement specifies the criteria that judge the operation of the oystercatcher that the specific behaviors. It specifies how the system should behave.

Typical example of non-functional requirement is:

* Performance
* Availability
* Scalability
* Recoverability etc.

# Prioritization

I have prioritized my requirements to understand its importance on the project. For prioritizing the task, I have used MoSCoW prioritization.

**M – Must have**

**S – Should have**

**C – Could have**

**W – Won’t have**

**Functional prioritization:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Functional Requirement** | **MoSCoW Prioritization** | **Justification** |
|  | Registration | M | Allows user to register to the system. |
|  | Login | M | Login grants access to the user to use facility of an application. |
|  | Post(add) advertisement | M | Allows user to post advertisement about the property |
|  | Book property | M | Buyer can book the property and contact owner later to buy the property |
|  | Search property | M | Desired property can be searched. Property can be searched according to location and own customization. |
|  | Add to Wishlist | C | Property that is liked by a user can be added to Wishlist so they can check it later. |
|  | View detail | M | Detail of the property can be viewed. |
|  | Filter property | S | Filtering the search of property according to the wish. |
|  | Comment | S | Comment can be provided so owner can get feedback regarding the price and other. |
|  | Update property | S | Added property can be updated. |
|  | Delete property | M | If we are not interested on posting an advertisement then we can delete it. |
|  | View property | M | Basically, we can view the property. |
|  | Edit profile | S | Profile of the user can be updated. |
|  | Chat | C | Buyer can directly chat with the owner if they are online. |
|  | Cost calculation | M | Cost of the product can be calculated by including the tax. |
|  | Online payment | W | This feature might no be available in the final product but it basically helps to pay for property online. |
|  | Bid property | W | Bidding for the price where owner starts the bit from low possible price. |

**Non-Functional prioritization**

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Non-Functional Requirement** | **MoSCoW Prioritization** | **Justification** |
|  | Scalability | M | Application should be able to run in different environment. |
|  | Efficiency | M | Application should be effective regarding time, cost and other. |
|  | Verification | M | Verification of the user should be done to know their identity (Via login). |
|  | Reliability | M | Application should be trustworthy (should be able to generate correct result ). |
|  | Usability | M | Should be easily useable. |
|  | Interoperability | S | Exchange of the information. |
|  | Maintainability | M | Application can be easily maintainable. |

# Architecture

## Initial class diagram

It is not a final class diagram but it helps to provide jist overview and structure of system in term of classes. Relation between the classes can also be identified (Inheritance, Association etc.). Initial class diagram is shown below.

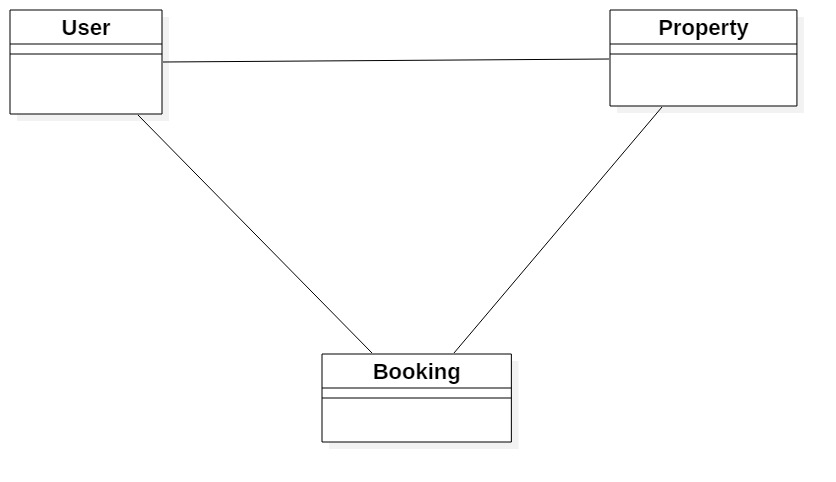


Figure Initial class diagram

## System architecture (ER-Diagram)

Entity relationship diagram (ERD) shows the relation of the entities. It is created by normalizing the existing table. It is a structural diagram used in database design. It contains different notation which helps to identify the relation. I have used crows-feet notation in my ER-Diagram.

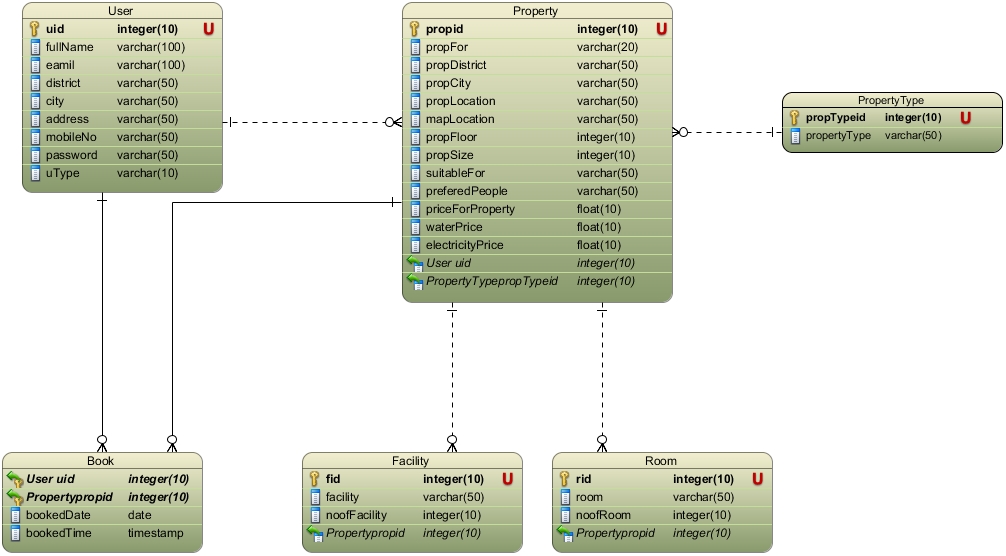


Figure ER-Diagram

## Class diagram showing MVC architecture

**Class Diagram** is a static structure diagram that describes the structure of the system by showing different classes, their attributes, their operations. Relationship among the object of the class is also shown. (Paradigm, 2019)

I have created the class diagram by showing MVC architecture of the diagram also. There are various ways of identifying classes for class diagram among the NLA is one.

**Natural Language Analysis (NLA):**

Before drawing class diagram, it is necessary to identify the possible classes, attributes and relation between different classes. NLA is the analysis process which helps to identify Nouns, Verb and adjective in the form of the descriptive text.

* **Nouns are the candidate class**
* **Verb are the are the potential functions of the class**
* **Adjectives are the potential attributes.**

Steps of constructing class diagram:

* Identify all the possible nouns and verbs
* Filtration is necessary as I am required to identify the genuine classes among all the classes.

For filtration following task was performed:

* Got rid of duplicate
* Complex words were removed
* Removing Irrelevancies candidate class (out of scope)
* Synonymous word was removed example: Meeting and Gathering
* Technical word was removed since they should be mention in the future example: keep a database.

Similar filtration process is also performed to identify the verbs.

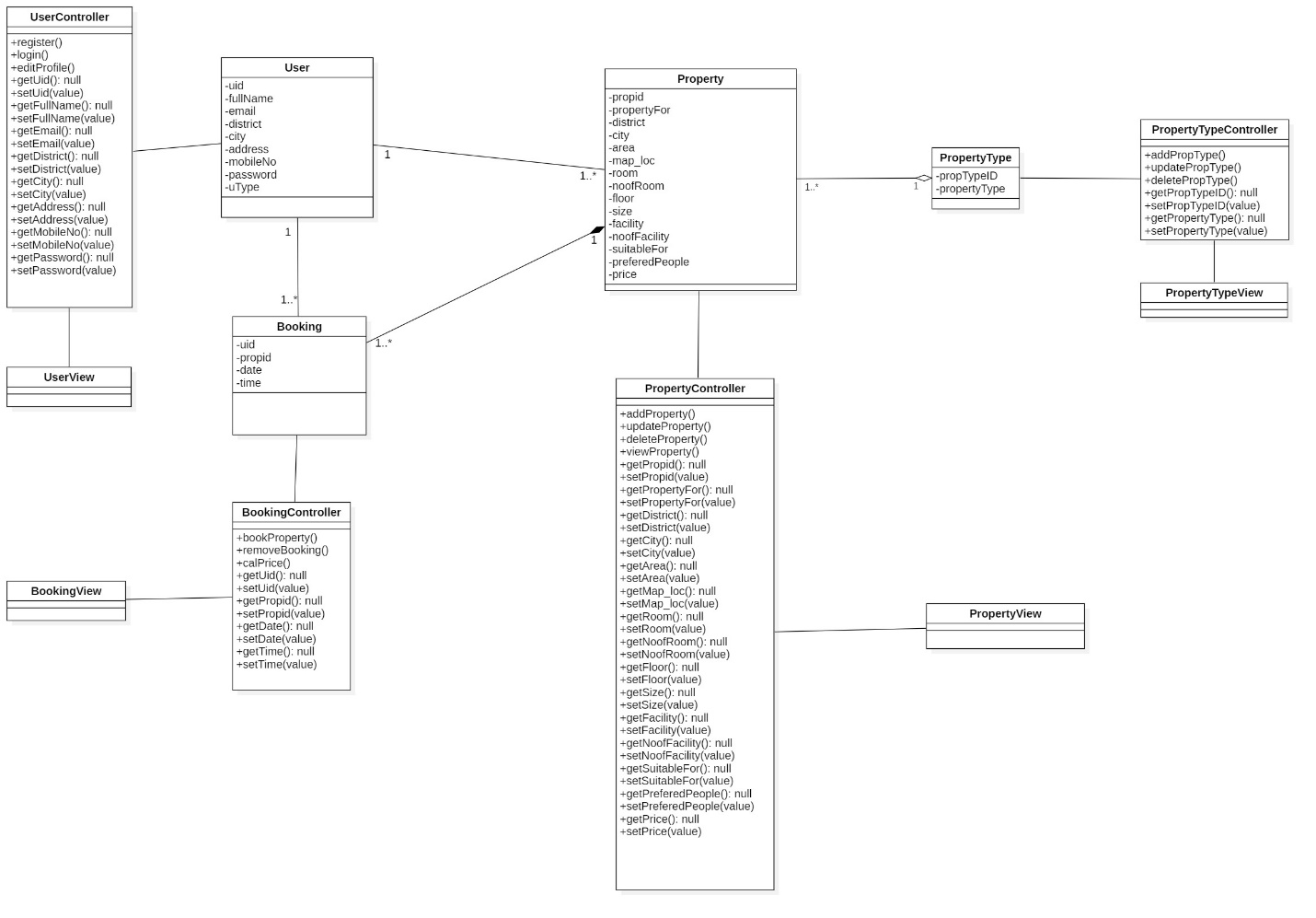


Figure 4 class diagram showing MVC architecture

# Activity diagram

Activity diagram is a dynamic diagram which helps to represent logical process of a system in diagrammatic format. It represents dynamic aspect of a system. Basically, it is a flow chart that shows flow of one activity to other activity. (TutorialPoints, 2019) Activity diagram also help to show detail process of different use cases that is identified in use case diagram.

These are the notations I have used in my activity diagrams:

* Swim-Lane: All of the classes are divided into different swim-lane to make it easy to understand.
* Initial node: Initial node represents the starting point of the activity.
* Activity: Rectangular boxes that represent the actions.
* Decision: Diamond shape that shows different choices and conditions.
* Flow: they are the arrow which guides towards the end of the diagram.
* Final node: This represent the end of activity.

The activity diagram is shown below:

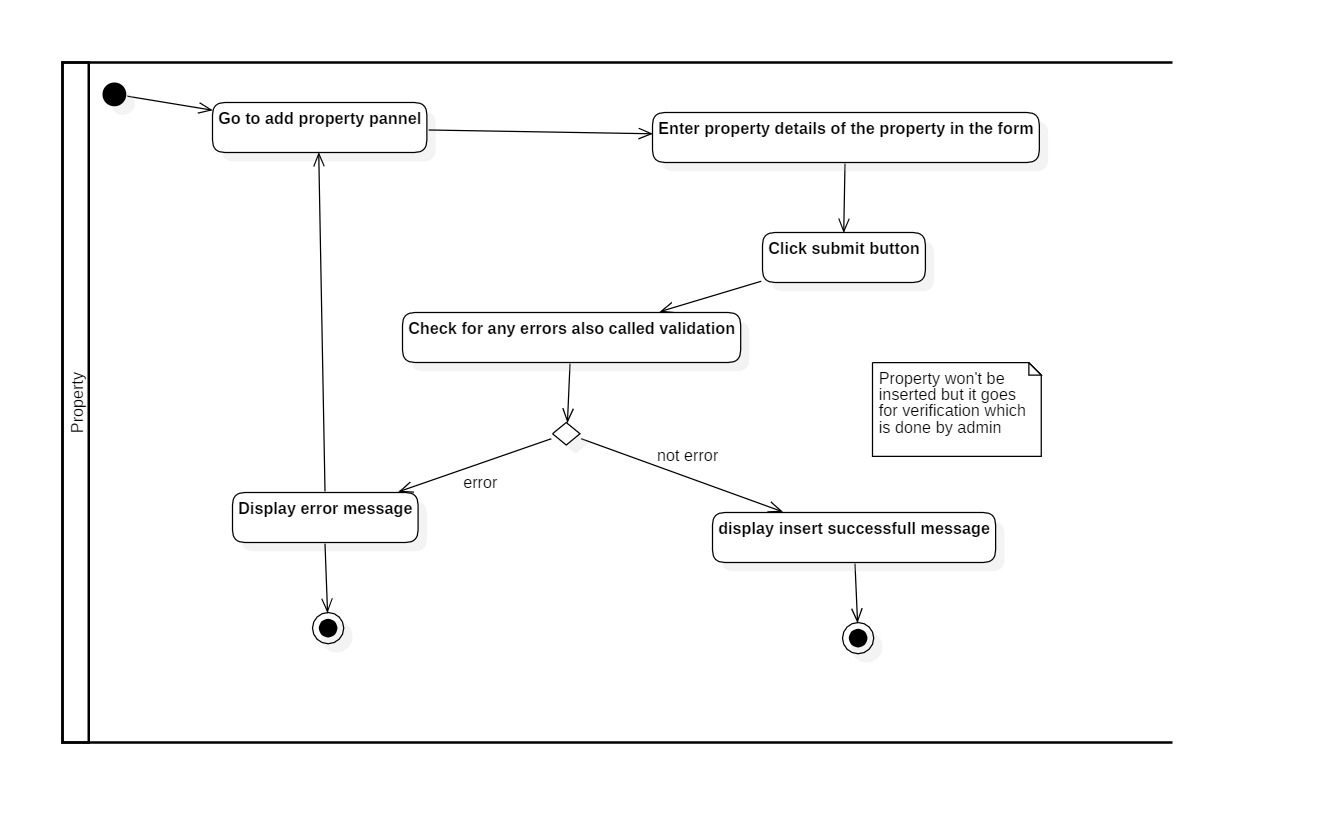


Figure Activity diagram to add property

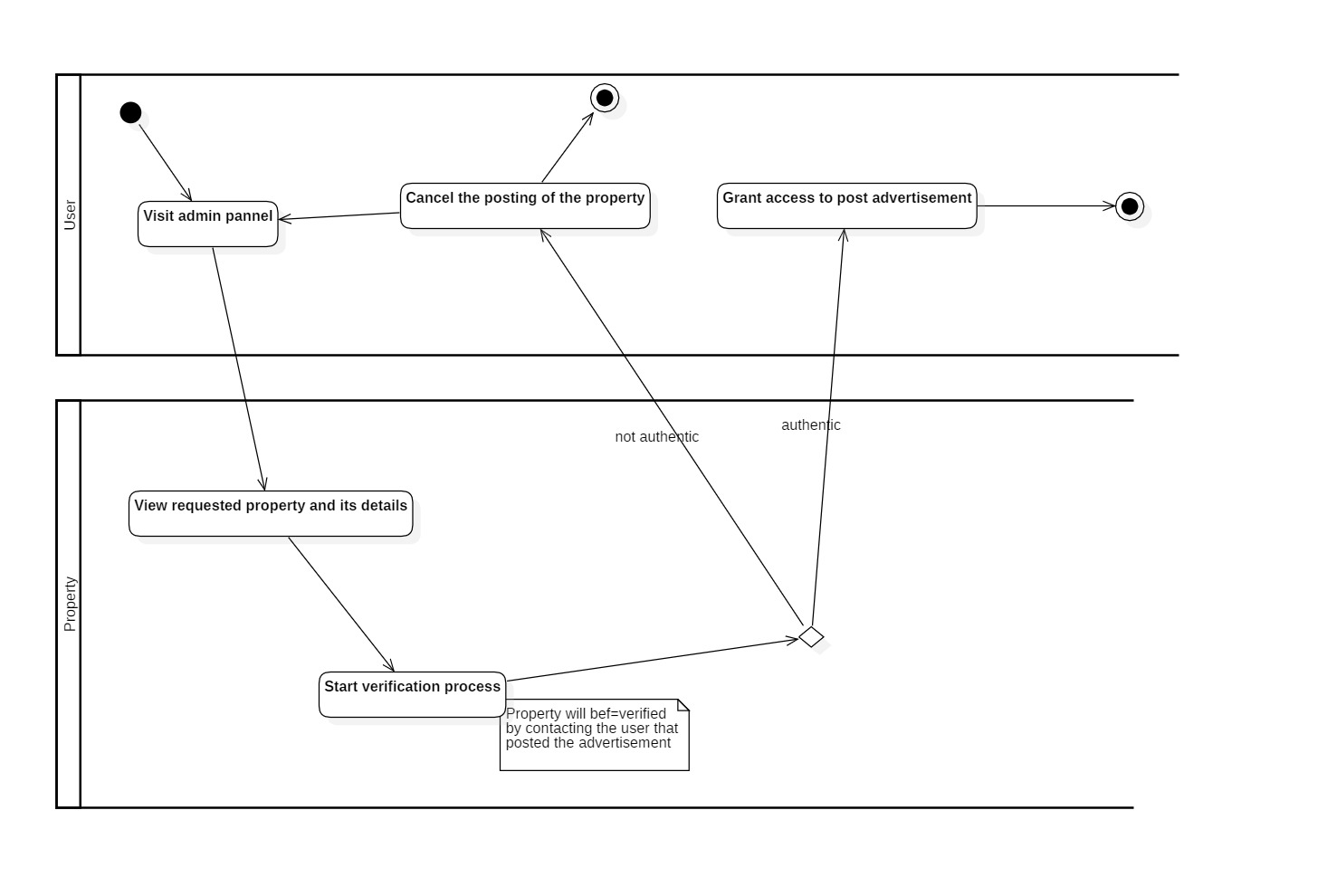


Figure Activity diagram for admin verification

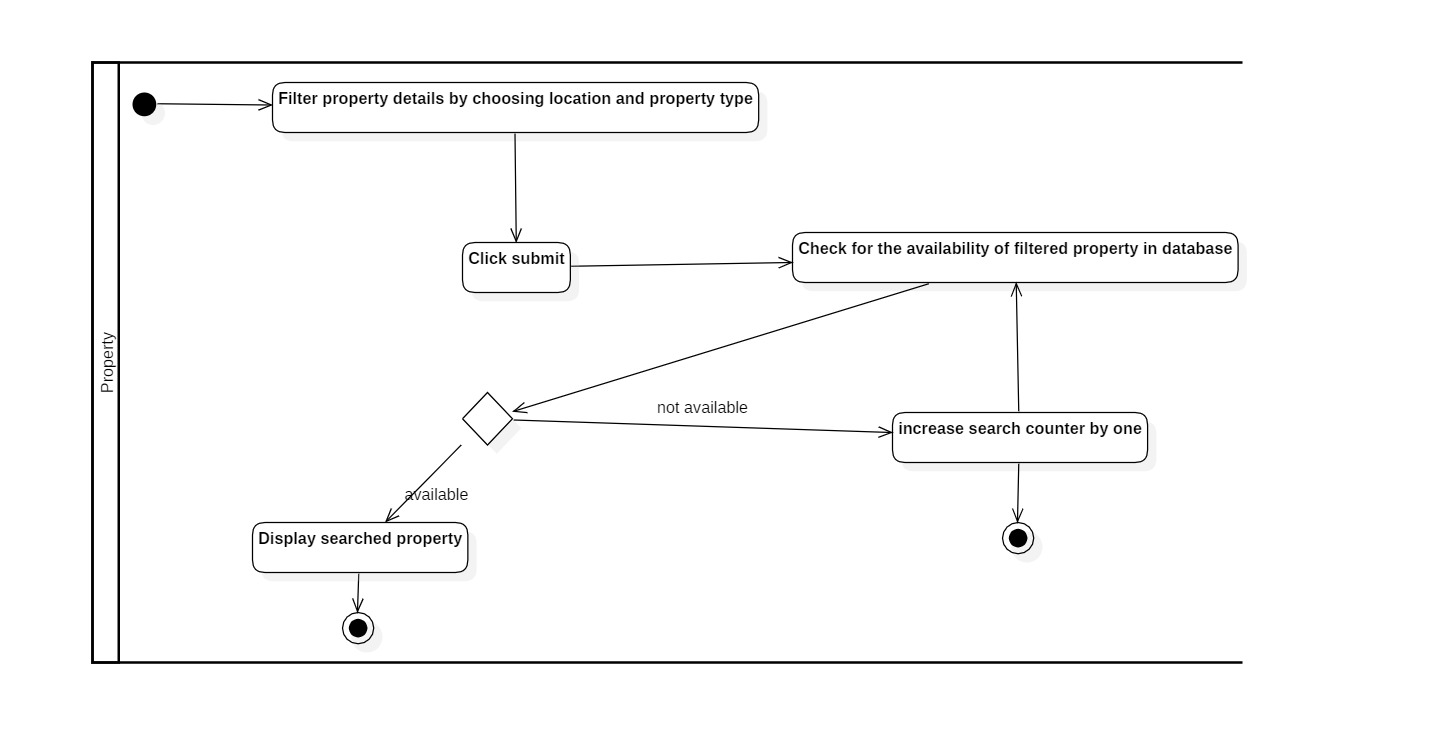


Figure Activity diagram to search property

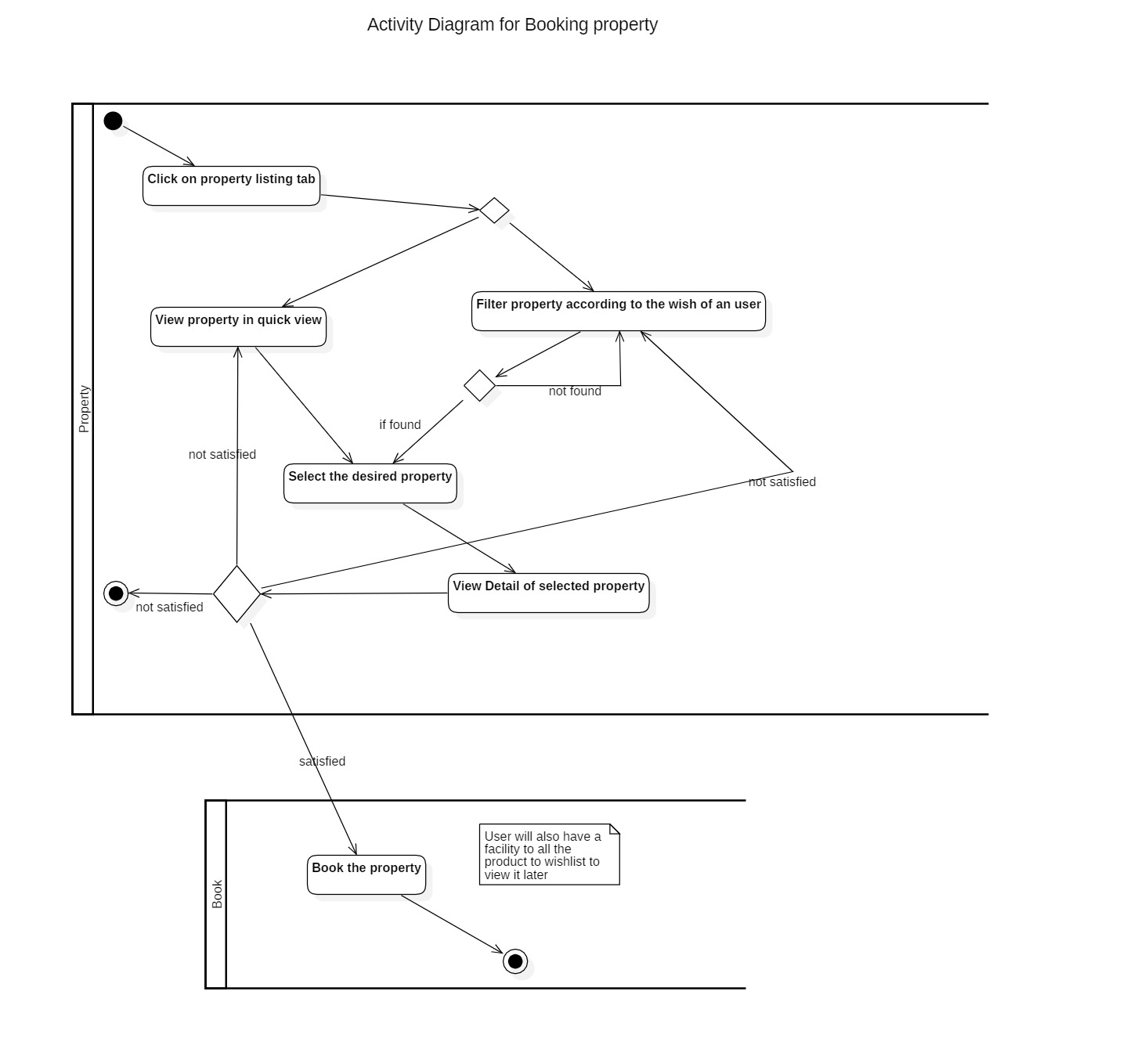


Figure Activity diagram to book property

Mainly activity diagram is used for two purpose:

* **For representing algorithm formally:** Each activity that are shown in the diagram have certain functionality in the real project.
* **For creating list of high-level activity:** This point will be shown in the diagram itself

Particularly by making activity diagram for the project it has helped me to understand my project even better as it allows to defragment the harsh scenario of any project.

# Conclusion

At last, all the necessary requirements are successfully identified and prioritized accordingly to know their priorities. Both functional and Non-Functional requirements was identified which helped to clarify the requirement further more. Use case diagram was created to know the responsibility of different actors involved in an application. Initial class diagram was created to know the structure of the application regarding classes.ER-Diagram helped to identified the relationship between different entities.