

# Practical 6

## Use Case Diagrams



A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.


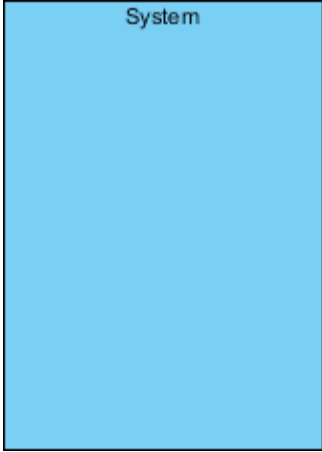
### Purpose of Use Case Diagrams

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Notation Description	Visual Representation
<b>Actor</b> <ul style="list-style-type: none"><li>• Someone interacts with a use case (system function).</li><li>• Named by noun.</li><li>• Actor plays a role in the business</li><li>• Similar to the concept of user, but a user can play different roles</li><li>• For example:<ul style="list-style-type: none"><li>o A prof. can be instructor and also researcher</li><li>o plays 2 roles with two systems</li></ul></li><li>• Actor triggers use case(s).</li><li>• Actor has a responsibility toward the system (inputs), and Actor has expectations from the system (outputs).</li></ul>	
<b>Use Case</b> <ul style="list-style-type: none"><li>• System function (process - automated or manual)</li><li>• Named by verb + Noun (or Noun Phrase).</li></ul>	

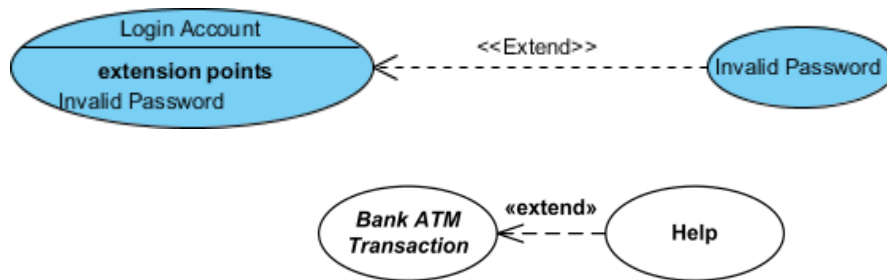
<ul style="list-style-type: none"> <li>• i.e. Do something</li> <li>• Each Actor must be linked to a use case, while some use cases may not be linked to actors.</li> </ul>	
<b>Communication Link</b> <ul style="list-style-type: none"> <li>• The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link.</li> <li>• Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.</li> </ul>	
<b>Boundary of system</b> <ul style="list-style-type: none"> <li>• The system boundary is potentially the entire system as defined in the requirements document.</li> <li>• For large and complex systems, each module may be the system boundary.</li> <li>• For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.</li> <li>• can form a system boundary for use cases specific to each of these business functions.</li> <li>• The entire system can span all of these modules depicting the overall system boundary</li> </ul>	

## Structuring Use Case Diagram with Relationships

Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modeling the dependency between the two use cases. The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system. Use case relationships are listed as the following:

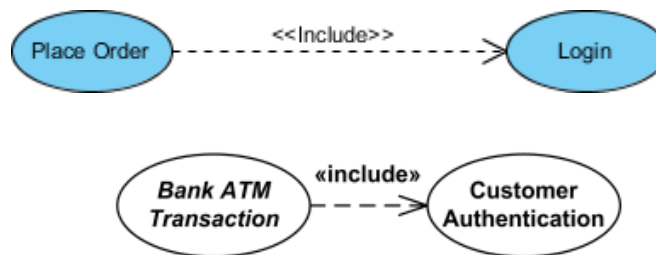
Use Case Relationship
<b>Extends</b> <ul style="list-style-type: none"> <li>• Indicates that an <b>"Invalid Password"</b> use case may include (subject to specified in the extension) the behavior specified by base use case <b>"Login Account"</b>.</li> <li>• Depict with a directed arrow having a dotted line. The tip of arrowhead points to the base use case and the child use case is connected at the base of the arrow.</li> </ul>

- The stereotype "<<extends>>" identifies as an extend relationship



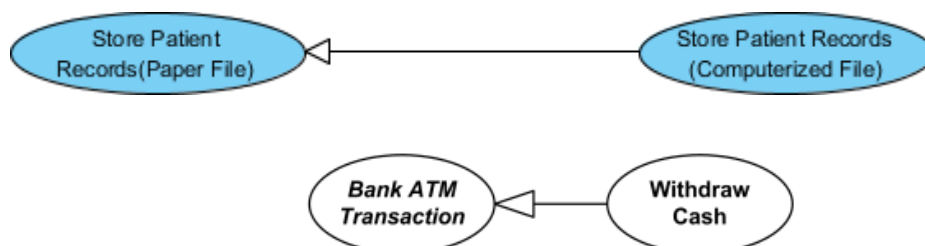
## Include

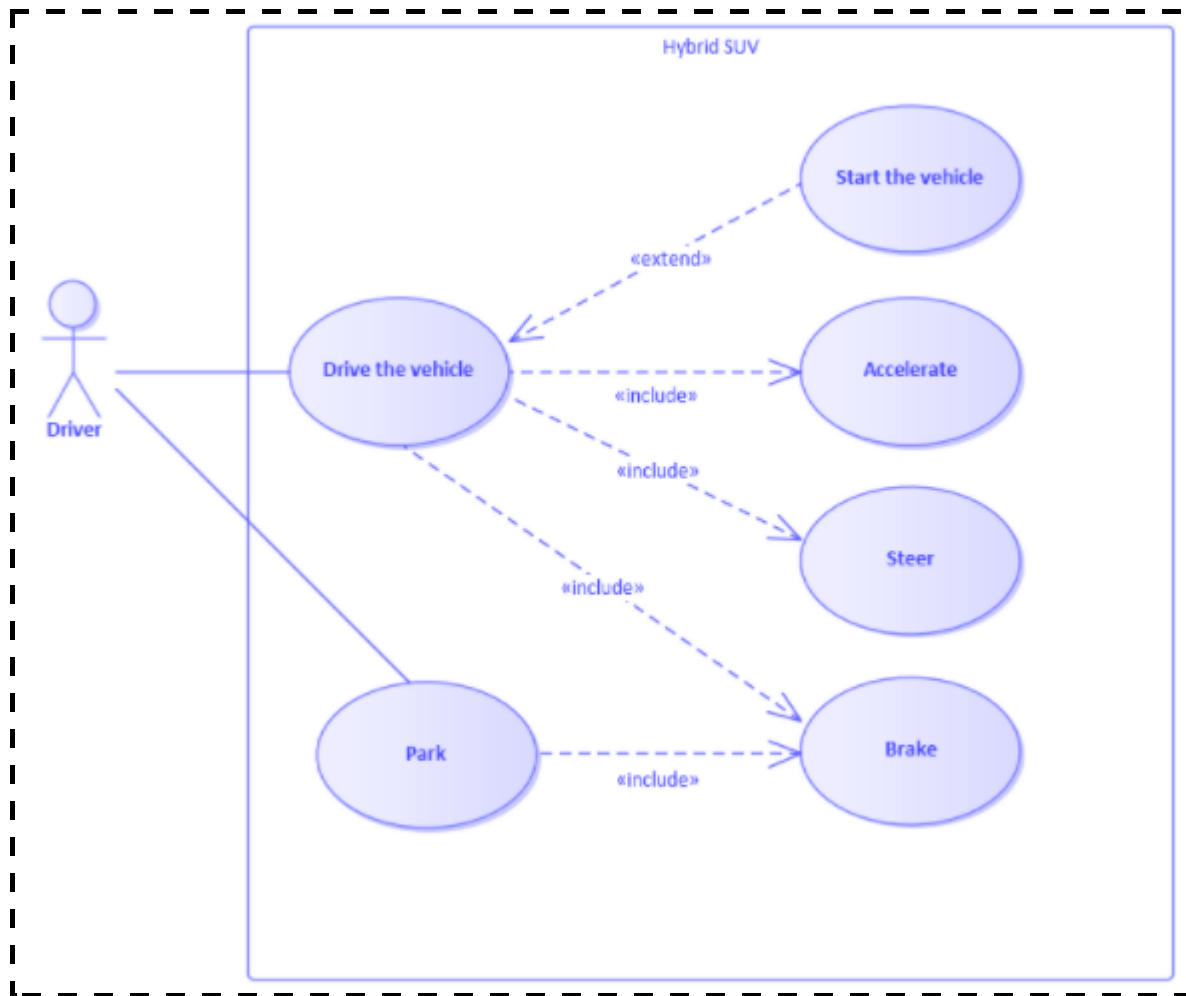
- When a use case is depicted as using the functionality of another use case, the relationship between the use cases is named as include or uses relationship.
- A use case includes the functionality described in another use case as a part of its business process flow.
- A uses relationship from base use case to child use case indicates that an instance of the base use case will include the behavior as specified in the child use case.
- An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at the base of the arrow.
- The stereotype "<<include>>" identifies the relationship as an include relationship.



## Generalization

- A generalization relationship is a parent-child relationship between use cases.
- The child use case is an enhancement of the parent use case.
- Generalization is shown as a directed arrow with a triangle arrowhead.
- The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.





## How to draw a Use Case diagram?

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

Once both the actors and use cases are enlisted, the relation between the actor and use case/system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

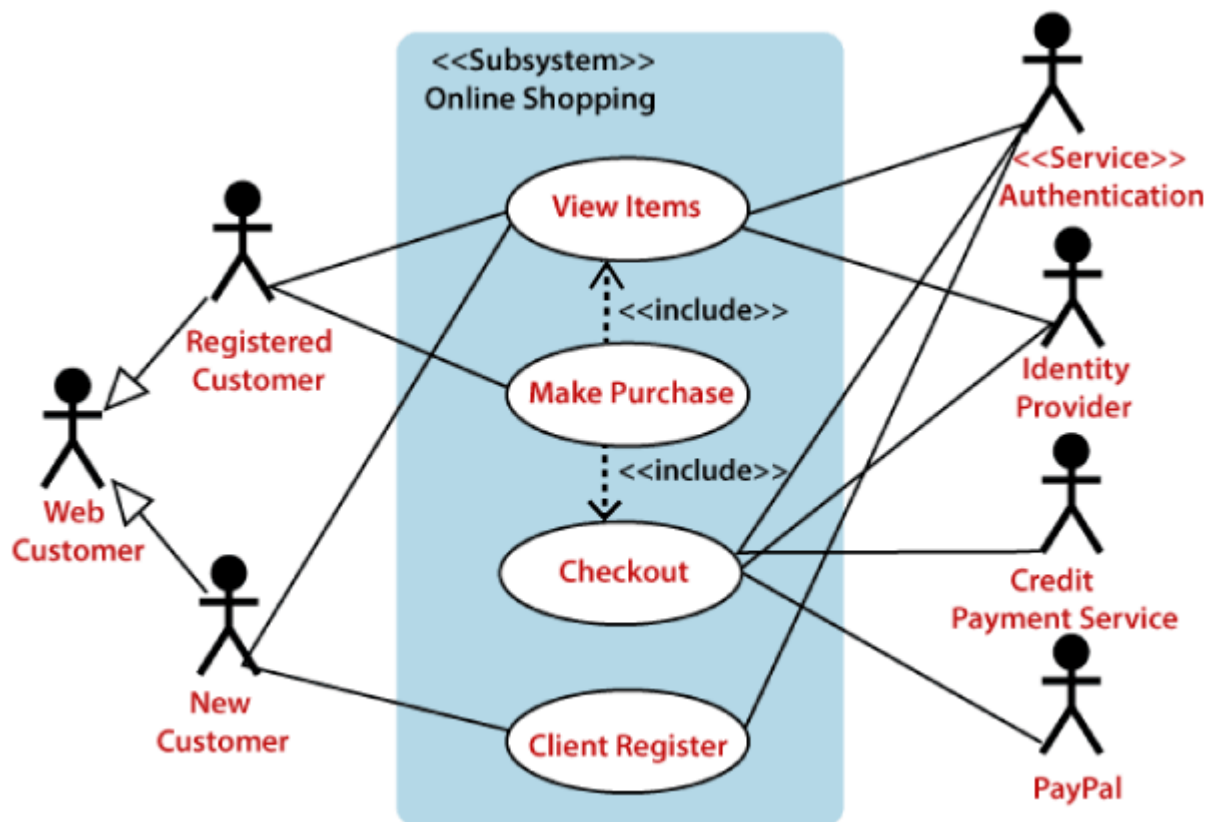
Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

## Example of a Use Case Diagram

A use case diagram depicting the Online Shopping website is given below.

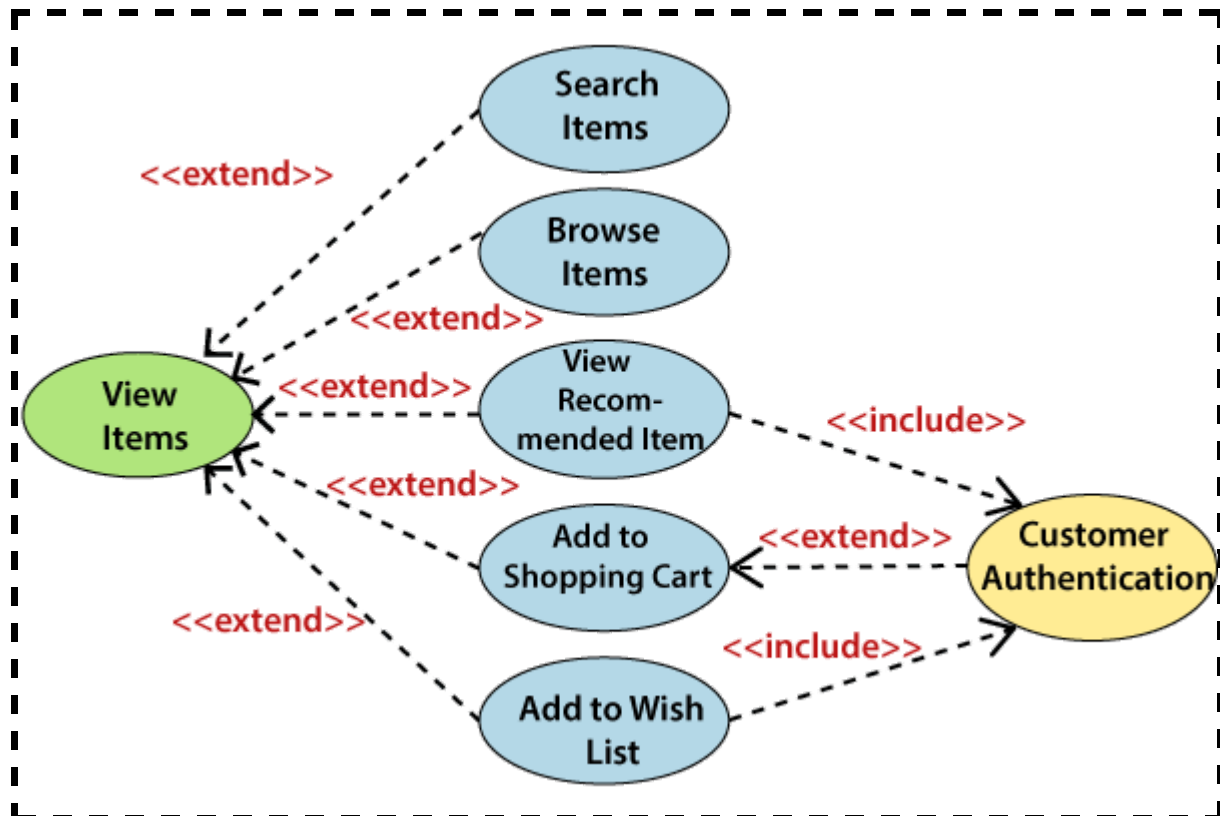
Here the Web Customer actor makes use of any online shopping website to purchase online. The top-level uses are as follows; View Items, Make Purchase, Checkout, Client Register. The **View Items** use case is utilized by the customer who searches and views products. The **Client Register** use case allows the customer to register itself with the website for availing gift vouchers, coupons, or getting a private sale invitation. It is to be noted that the **Checkout** is an included use case, which is part of **Making Purchase**, and it is not available by itself.



The **View Items** is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these extended use cases provide some functions to customers, which allows them to search for an item. The View Items is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these

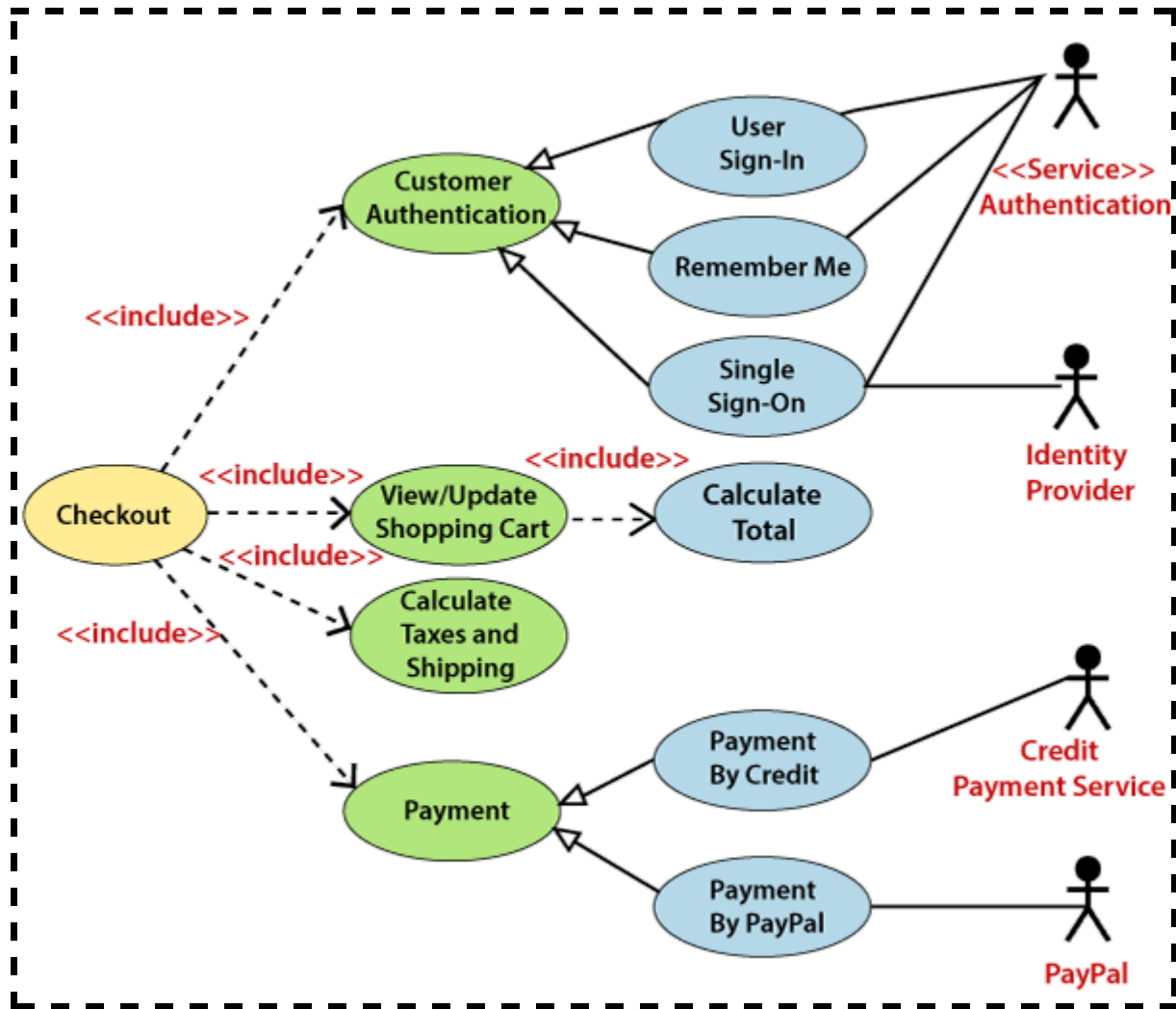
extended use cases provide some functions to customers, which allows them to search for an item.

Both **View Recommended Item** and **Add to Wish List** include the Customer Authentication use case, as they necessitate authenticated customers, and simultaneously item can be added to the shopping cart without any user authentication.



Similarly, the **Checkout** use case also includes the following use cases, as shown below. It requires an authenticated Web Customer, which can be done by login page, user authentication cookie ("Remember me"), or Single Sign-On (SSO). SSO needs an external identity provider's participation, while Web site authentication service is utilized in all these use cases.

The Checkout use case involves Payment use case that can be done either by the credit card and external credit payment services or with PayPal.



## Important tips for drawing a Use Case diagram

Following are some important tips that are to be kept in mind while drawing a use case diagram:

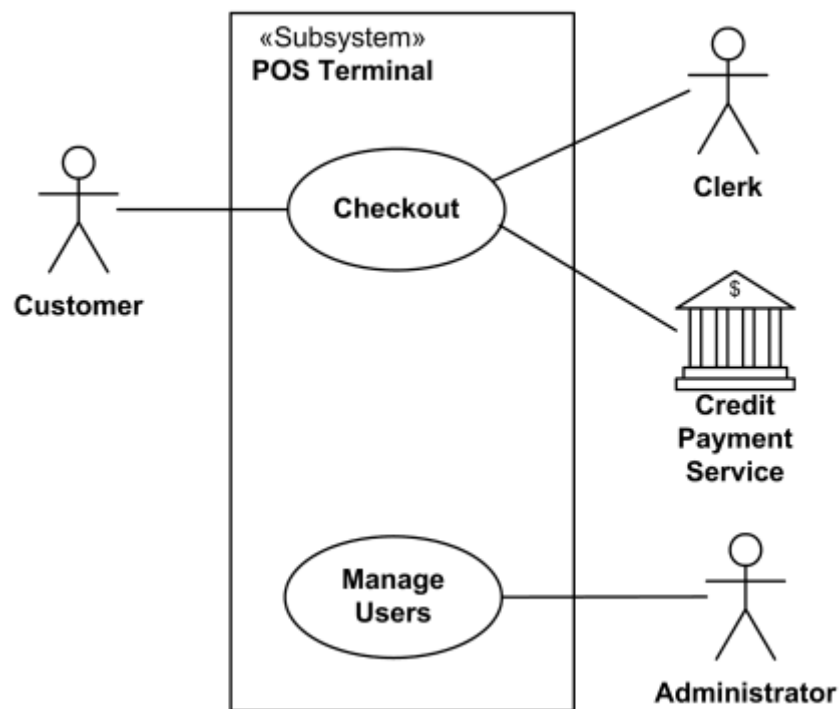
1. A simple and complete use case diagram should be articulated.
2. A use case diagram should represent the most significant interaction among the multiple interactions.
3. At least one module of a system should be represented by the use case diagram.
4. If the use case diagram is large and more complex, then it should be drawn more generalized.

## Example - Point of Sales Terminal

An example of UML use case diagram for Point of Sale (POS) Terminal or Checkout.

A retail POS system typically includes a computer, monitor, keyboard, barcode scanners, weight scale, receipt printer, credit card processing system, etc. and POS terminal software.

Checkout use case involves Customer, Clerk and Credit Payment Service actors and includes scanning items, calculating total and taxes, payment use cases.

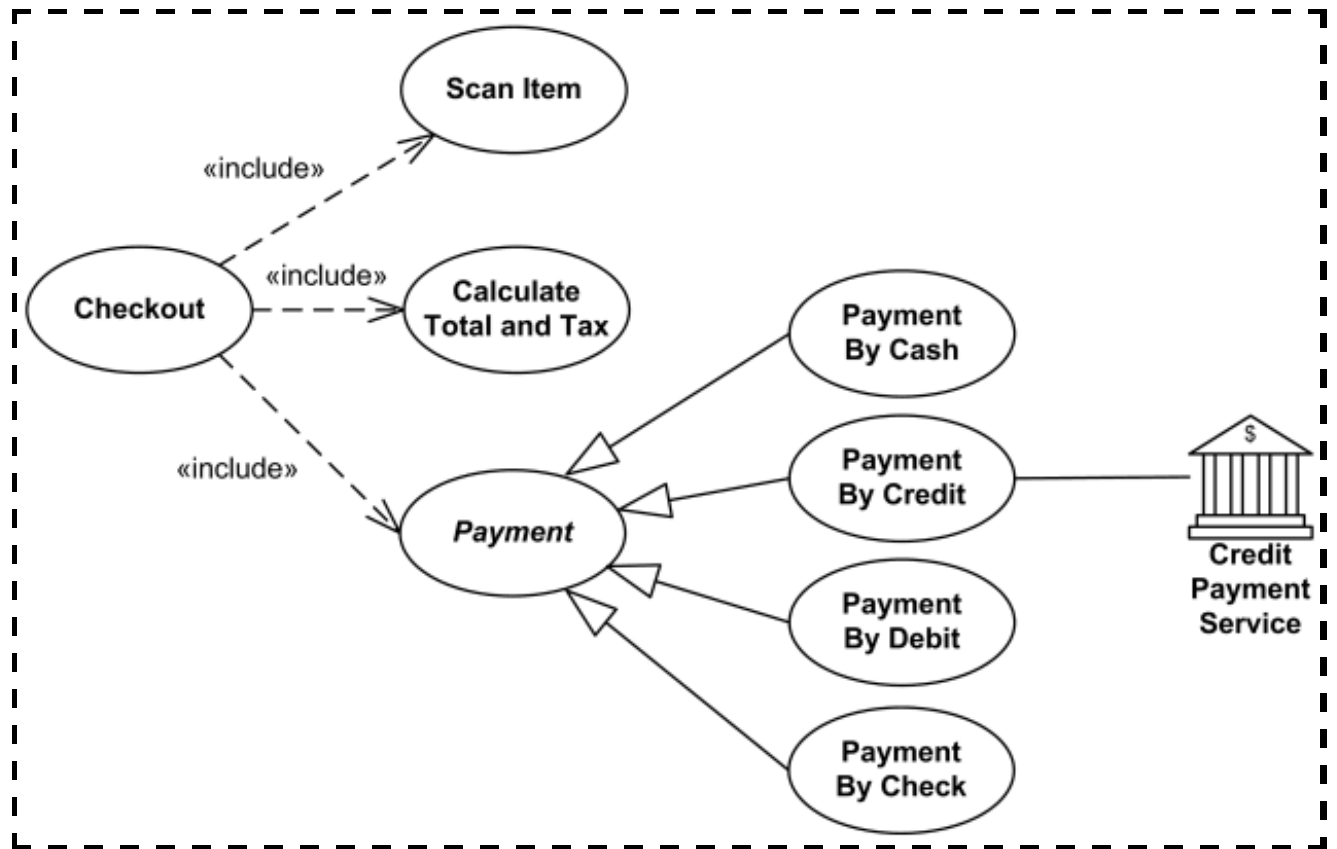


Checkout use case requires Customer actor, hence the 1 multiplicity of Customer. Clerk can only participate in a single Checkout use case. Credit Payment Service can participate with many Checkout use cases at the same time. Checkout use case may not need Credit Payment Service (for example, if payment is in cash), thus the 0..1 multiplicity.

Checkout use case is an example of a large and complex use case split into several use cases each describing some logical unit of behavior. Note, that including use case becomes incomplete by itself and requires the included use cases to be complete.

Payment use case is represented using generalization relationship. It means that only one specific type of payment is accepted - either by cash, or by credit, debit, or with check. An alternative to such representation could be to use include relationship so that not just single but several forms of payment could be accepted from the same client during checkout.

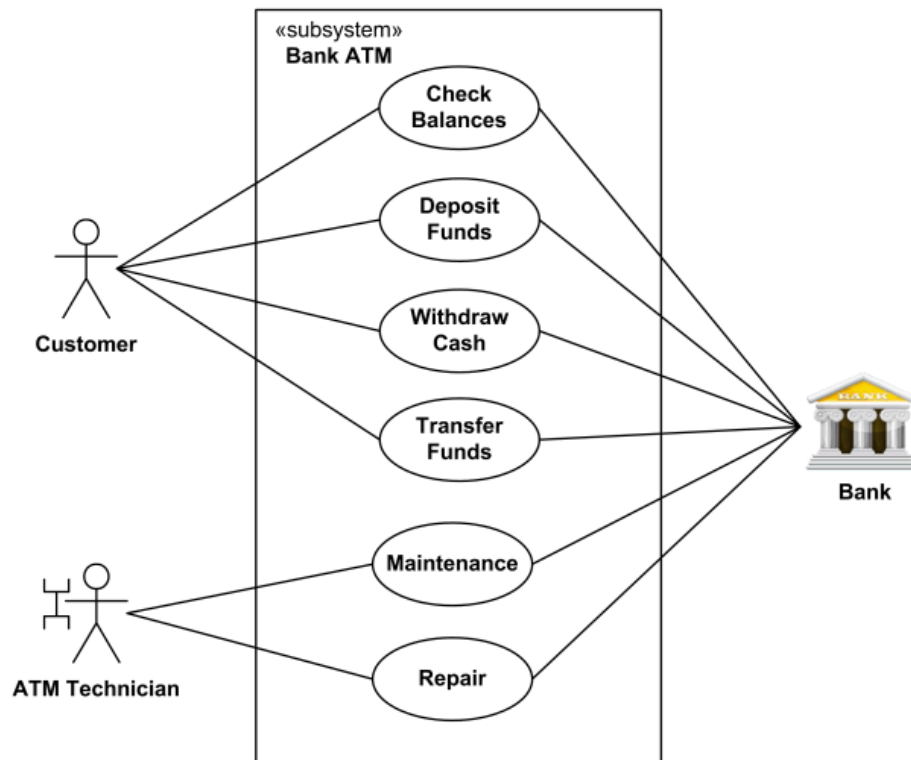




## Example - Bank ATM

An automated teller machine (ATM) is a banking subsystem (subject) that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller.

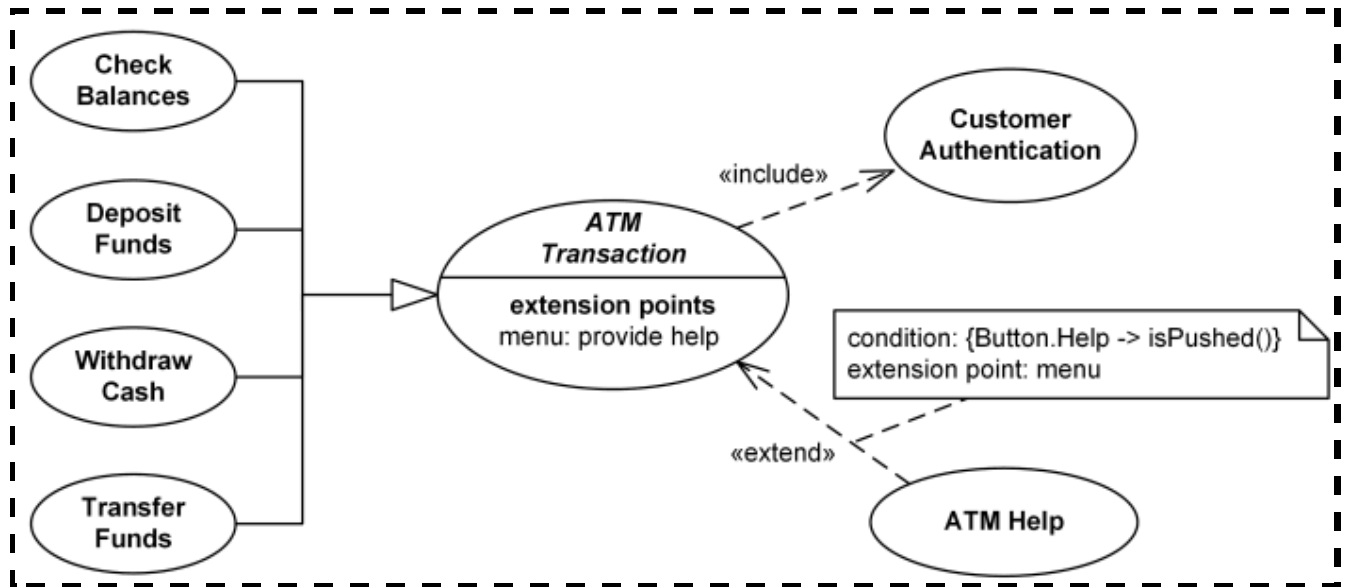
Customer (actor) uses bank ATM to Check Balances of his/her bank accounts, Deposit Funds, Withdraw Cash and/or Transfer Funds (use cases). ATM Technician provides Maintenance and Repairs. All these use cases also involve Bank actor whether it is related to customer transactions or to the ATM servicing.



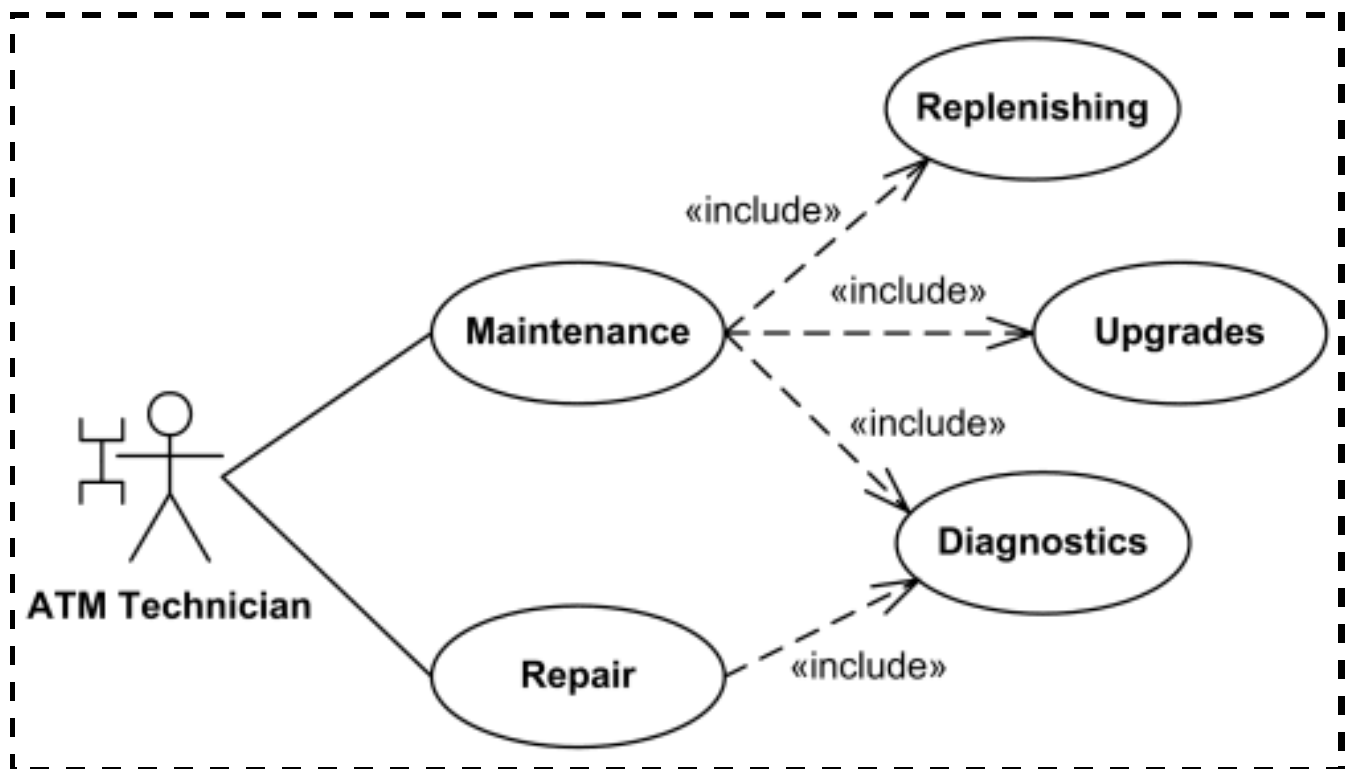
On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN). Customer Authentication use case is required for every ATM transaction so we show it as include relationship. Including this use case as well as transaction generalizations make the ATM Transaction an abstract use case.

Customer may need some help from the ATM. ATM Transaction use case is extended via extension point called menu by the ATM Help use case whenever ATM Transaction is at the location specified by the menu and the bank customer requests help, e.g. by selecting Help menu item.

ATM Technician maintains or repairs Bank ATM. Maintenance use case includes Replenishing ATM with cash, ink or printer paper, Upgrades of hardware, firmware or software, and remote or on-site Diagnostics. Diagnostics is also included in (shared with) Repair use case.



*Bank ATM Transactions and Customer Authentication Use Cases Example.*



*Bank ATM Maintenance, Repair, Diagnostics Use Cases Example.*

## Example - Online Shopping - Credit Cards Processing

This UML use case diagram example shows some use cases for a system which processes credit cards.

- Credit Card Processing System (i.e. Credit Card Payment Gateway) is a subject, i.e. system under design or consideration. Primary actor for the system is a Merchant's Credit Card Processing System. The merchant submits some credit card transaction request to the credit card payment gateway on behalf of a customer. Bank which issued the customer's credit card is an actor which could approve or reject the transaction. If a transaction is approved, funds will be transferred to the merchant's bank account.
- Authorize and Capture use case is the most common type of credit card transaction. The requested amount of money should be first authorized by Customer's Credit Card Bank, and if approved, is further submitted for settlement. During the settlement funds approved for the credit card transaction are deposited into the Merchant's Bank account.
- In some cases, only authorization is requested and the transaction will not be sent for settlement. In this case, usually if no further action is taken within some number of days, the authorization expires. Merchants can submit this request if they want to verify the availability of funds on the customer's credit card, if the item is not currently in stock, or if the merchant wants to review orders before shipping.
- Capture (request to capture funds that were previously authorized) use case describes several scenarios when a merchant needs to complete some previously authorized transaction - either submitted through the payment gateway or requested without using the system, e.g. using voice authorization.
- Credit use case describes situations when a customer should receive a refund for a transaction that was either successfully processed and settled through the system or for some transaction that was not originally submitted through the payment gateway.
- Void use case describes cases when it is needed to cancel one or several related transactions that were not yet settled. If possible, the transactions will not be sent for settlement. If the Void transaction fails, the original transaction is likely already settled.
- Verify use case describes zero or small amount verification transactions which could also include verification of some client's data such as address.

