**FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR**

**DEPARTMENT OF COMPUTER SCIENCE**

**CL2002 – SOFTWARE DESIGN AND ARCHITECTURE LAB**



**LAB MANUAL # 02**

**Visual Paradigm and Use Case diagram**

**Instructor: Mazhar Iqbal**

**SEMESTER SPRING 2024**

Contents

[Visual Paradigm 2](#_Toc137469855)

[Installation of Visual Paradigm 2](#_Toc137469856)

[UML – Use Case Diagram 16](#_Toc137469857)

[How To Write a Use Case 16](#_Toc137469858)

[Use Case Diagram objects 16](#_Toc137469859)

[Actor 17](#_Toc137469860)

[Use Case 17](#_Toc137469861)

[System 17](#_Toc137469862)

[Relationships in Use Case Diagrams 17](#_Toc137469863)

[How to Create a Use Case Diagram: 17](#_Toc137469864)

[Bank ATM 19](#_Toc137469865)

[UML Use Case Diagram Examples 19](#_Toc137469866)

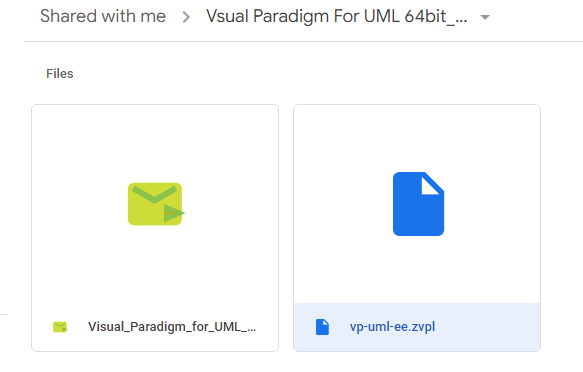
[Use Case Diagram from Case Study: 21](#_Toc137469867)

# Visual Paradigm

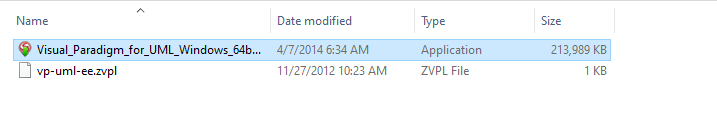
A visual Paradigm is a tool used for creating & managing the UML or the unified modeling language cases,

## Installation of Visual Paradigm

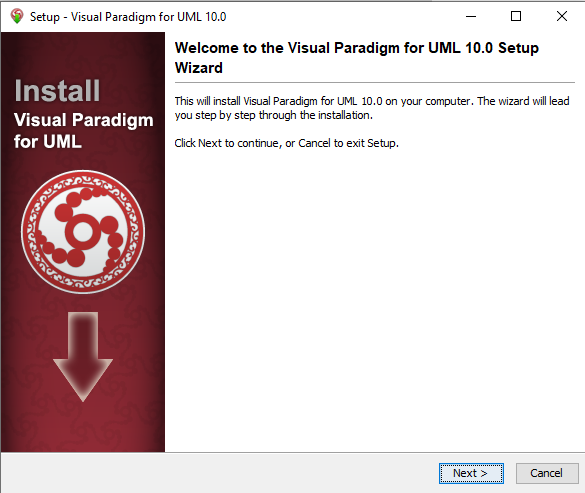
Open the provided folder shared on google drive and download the folder.



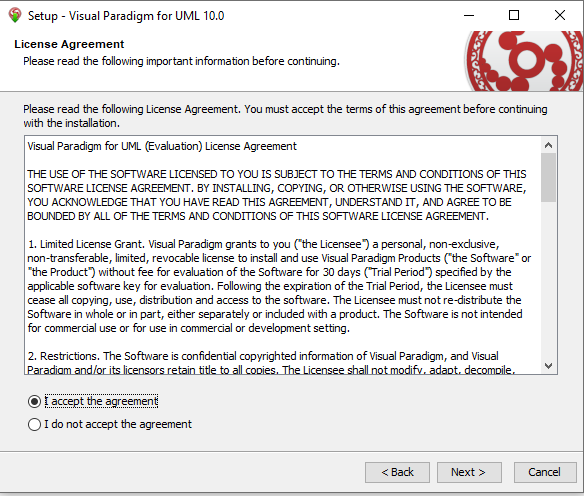
Click on the Installer .exe file.



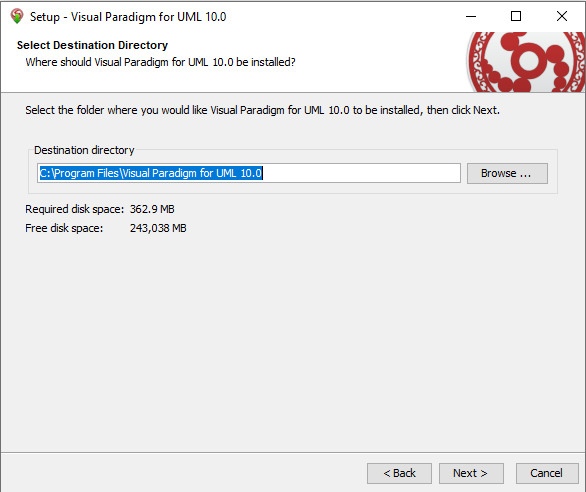
Click Next.



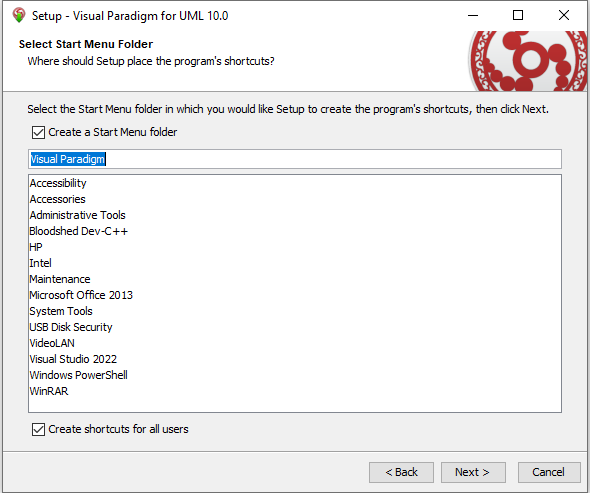
Accept the license agreement.



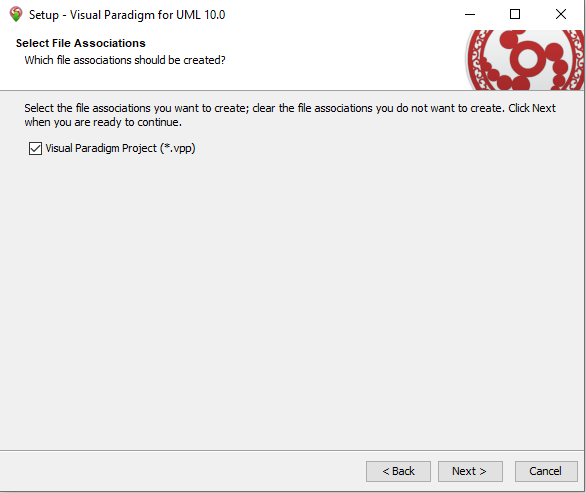
You can change the default installation directory if you want. If not proceed with the default one and click next.

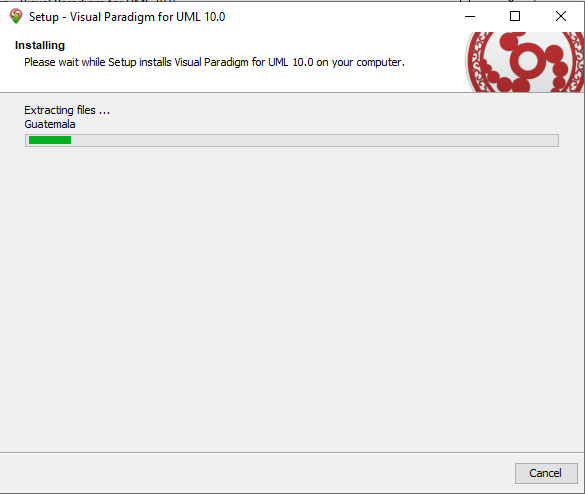


Click Next.



Make sure to check the checkbox and then click Next to proceed the installation.

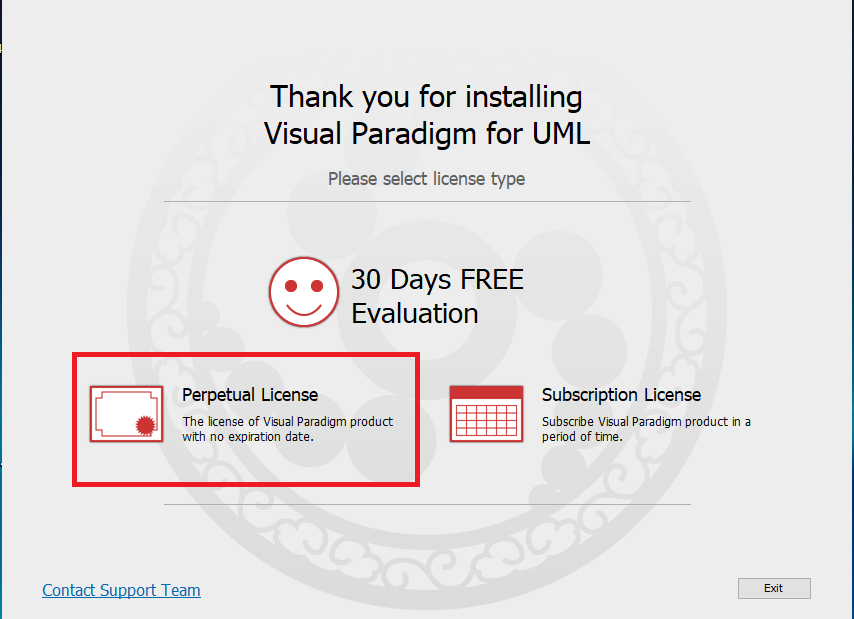




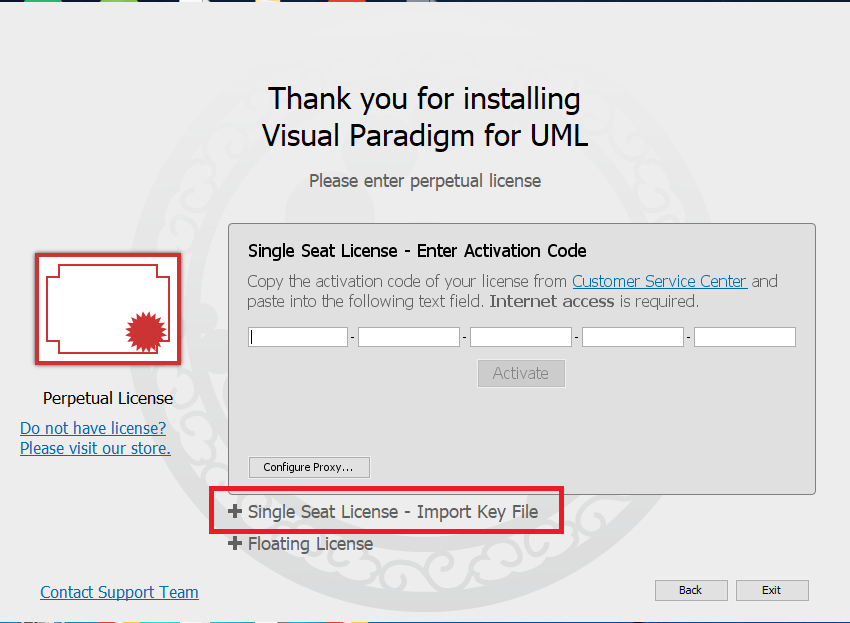
Start the visual Paradigm by clicking on Finish.



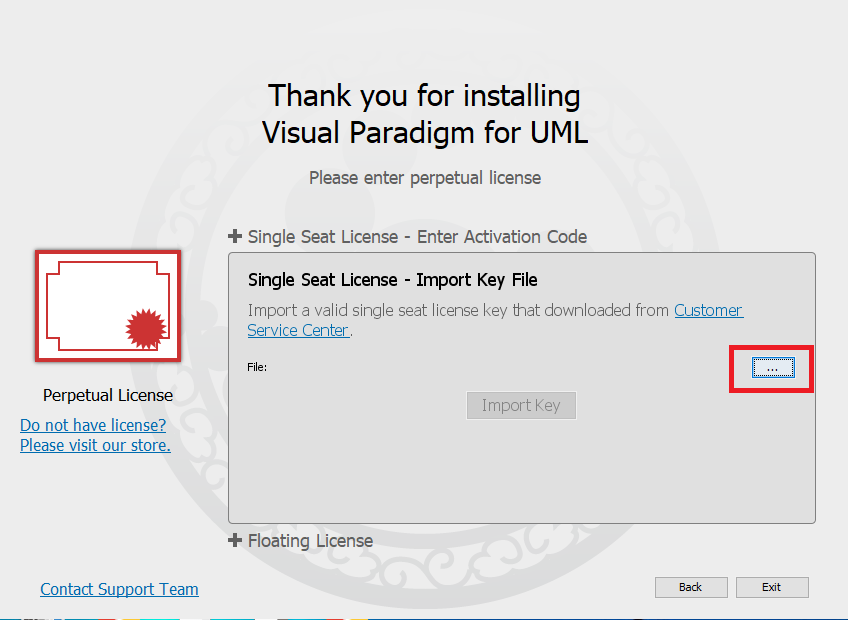
Now activate the product by clicking on Perpetual License.



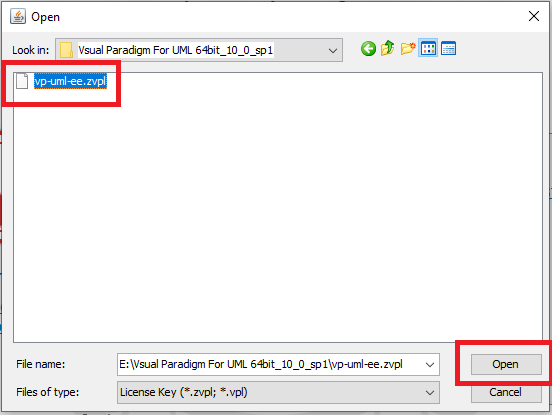
Click on Single Seat License to import the license file.



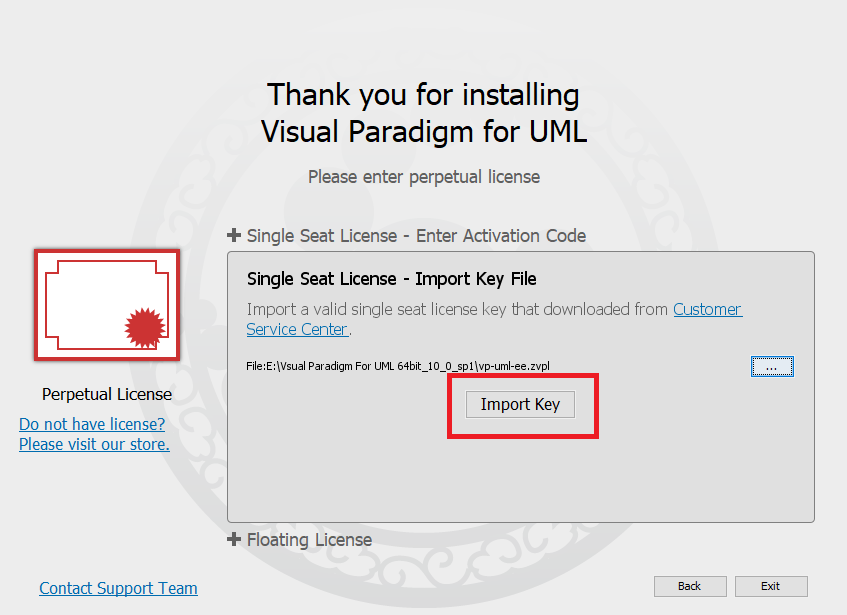
Click on the three dots to locate the license file.



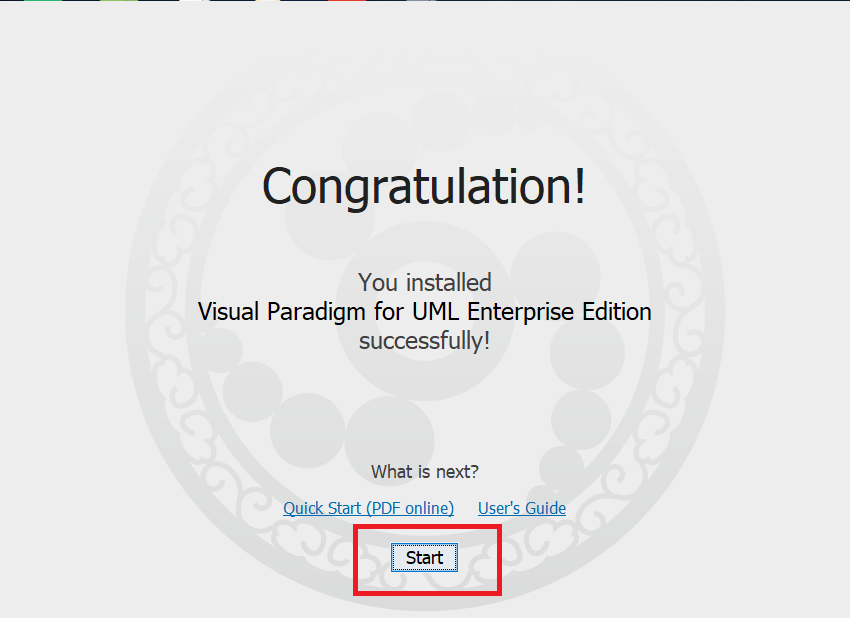
Browse and open the license file.



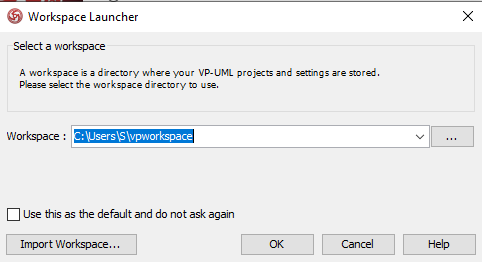
Click import key.



Installation is successfully finished. Now start the Visual Paradigm.



By default, all the visual paradigm projects will be saved in the following default workspace. If you want to change, select a different directory.



# UML – Use Case Diagram

A [**UML**](https://en.wikipedia.org/wiki/Unified_Modeling_Language) use case diagram is the visual form of the expected behavior (what), users, and their interactions with the use case. Each use case depicts an expected behavior/feature of the system, and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram is usually simple. It does not show the detail of the use cases:

* It only summarizes **some of the relationships** between use cases, actors, and systems.
* It does **not show the order** in which steps are performed to achieve the goals of each use case.

## How To Write a Use Case

Identify who is going to be using the System.

1. Pick one of those users.
2. Define what that user wants to do on the site. Each thing the use does on the site becomes a use case.
3. For each use case, decide on the normal course of events when that user is using the site.
4. Describe the basic course in the description for the use case. Describe it in terms of what the user does and what the system does in response that the user should be aware of.
5. When the basic course is described, consider alternate courses of events and add those to "extend" the use case.
6. Look for commonalities among the use cases. Extract these and note them as common course use cases.
7. Repeat the steps 2 through 7 for all other users.

## ****Use Case Diagram objects****

Use case diagrams consist of 3 objects.

* Actor
* Use case
* System

### Actor

Actor in a use case diagram is **any entity that performs a role** in one given system. This could be a person, organization or an external system and usually drawn like skeleton.

### Use Case

A use case **represents a function or an action within the system**. It’s drawn as an oval and named with the function.

### System

The system is used to **define the scope of the use case** and drawn as a rectangle. This an optional element but useful when you’re visualizing large systems. For example, you can create all the use cases and then use the system object to define the scope covered by your project.

## Relationships in Use Case Diagrams

There are five types of relationships in a use case diagram. They are

* Association between an actor and a use case
* Generalization of an actor
* Extend relationship between two use cases
* Include relationship between two use cases
* Generalization of a use case

## How to Create a Use Case Diagram:

**Identifying Actors**

Actors are external entities that interact with your system. It can be a person, another system or an organization. In a banking system, the most obvious actor is the customer. Other actors can be bank employee or cashier depending on the role you’re trying to show in the use case.

An example of an external organization can be the tax authority or the central bank. The loan processor is a good example of an external system associated as an actor.

##### **Identifying Use Cases**

Now it’s time to identify the use cases. A good way to do this is to identify what the actors need from the system. In a banking system, a customer will need to open accounts, deposit and withdraw funds, request check books and similar functions. So all of these can be considered as use cases.

Top level use cases should always provide a complete function required by an actor. You can extend or include use cases depending on the complexity of the system.

Once you identify the actors and the top level use case you have a basic idea of the system. Now you can fine tune it and add extra layers of detail to it.

##### Look for Common Functionality to use Include

Look for common functionality that can be reused across the system. If you find two or more use cases that share common functionality you can extract the common functions and add it to a separate use case. Then you can connect it via the include relationship to show that it’s always called when the original use case is executed.

##### Is it Possible to Generalize Actors and Use Cases

There may be instances where actors are associated with similar use cases while triggering a few use cases unique only to them. In such instances, you can generalize the actor to show the inheritance of functions. You can do a similar thing for use case as well.

One of the best examples of this is “Make Payment” use case in a payment system. You can further generalize it to “Pay by Credit Card”, “Pay by Cash”, “Pay by Check” etc. All of them have the attributes and the functionality of payment with special scenarios unique to them.

##### **Optional Functions or Additional Functions**

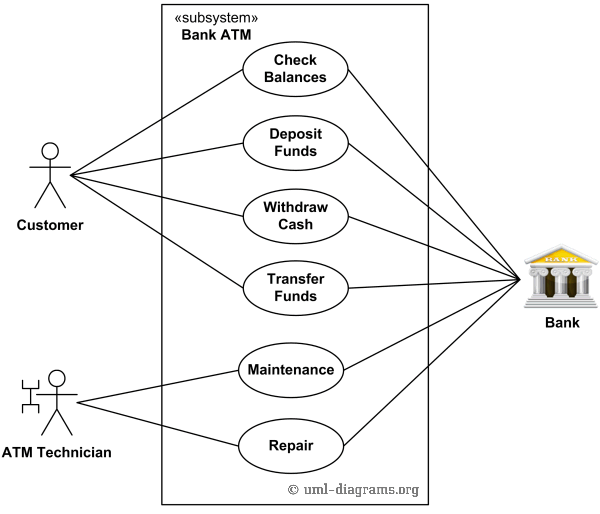
There are some functions that are triggered optionally. In such cases, you can use the extend relationship and attach an extension rule to it. In the below banking system example “Calculate Bonus” is optional and only triggers when a certain condition is matched.

## Bank ATM

## UML Use Case Diagram Examples

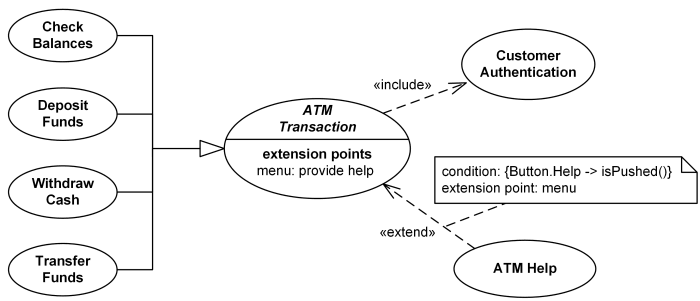
An automated teller machine (**ATM**) or the automatic banking machine (**ABM**) is a banking subsystem ([subject](https://www.uml-diagrams.org/use-case-subject.html)) that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller.

*Customer* ([actor](https://www.uml-diagrams.org/use-case-actor.html)) uses bank ATM to *Check Balances* of his/her bank accounts, *Deposit Funds*, *Withdraw Cash* and/or *Transfer Funds* ([use cases](https://www.uml-diagrams.org/use-case.html)). *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.



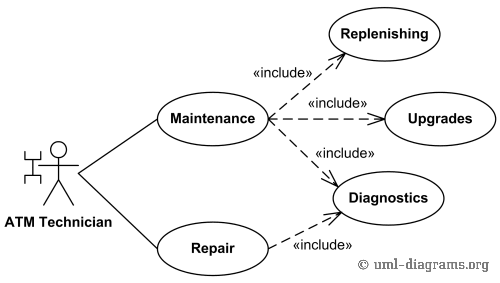
An example of use case diagram for Bank ATM subsystem - top level use cases.

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](https://www.uml-diagrams.org/use-case-include.html) relationship. Including this use case as well as transaction [generalizations](https://www.uml-diagrams.org/use-case.html#generalization) make the *ATM Transaction* an [abstract use case](https://www.uml-diagrams.org/use-case.html#abstract-use-case).



Bank ATM Transactions and Customer Authentication Use Cases Example.

Customer may need some help from the ATM. *ATM Transaction* use case is [extended](https://www.uml-diagrams.org/use-case-extend.html) via [extension point](https://www.uml-diagrams.org/use-case-extend.html#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.  **Replenishing** (to make full or complete again)

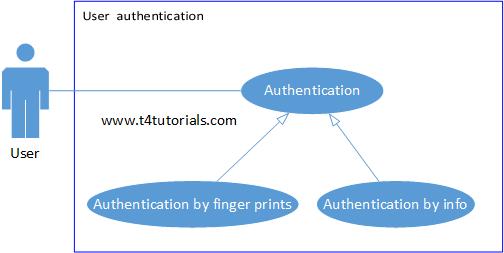


Bank ATM Maintenance, Repair, Diagnostics Use Cases Example.

*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](https://www.uml-diagrams.org/use-case-include.html) in (shared with) *Repair* use case.

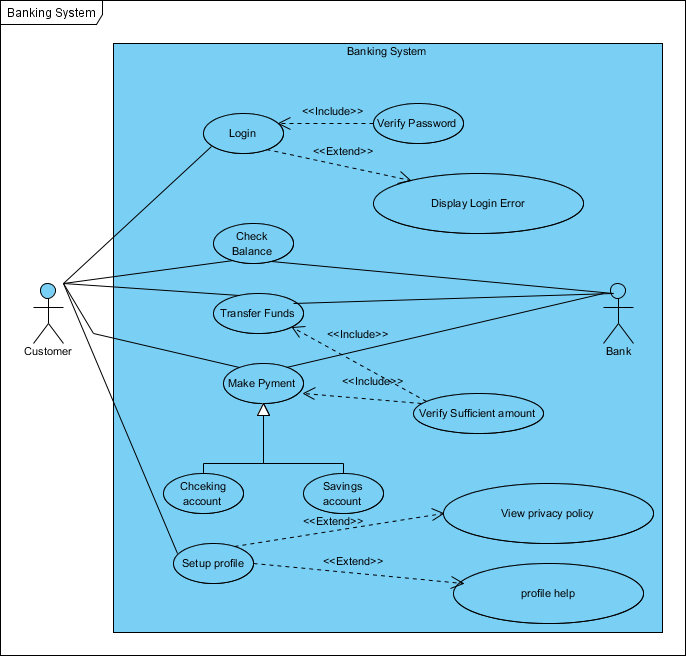
Example 3:

This diagram represents inheritance among use cases. authentication is the parent use case and authentication by finger authentication by info are child use cases. It means that some of the variables, functions are something else is inherited from parent to child.



Use Case Diagram from Case Study:  
You are required to create a Use-case diagram for a banking app. Where a customer can login, check balance, transfer funds, and make payment (from checking account or savings account). After the login, the password should verify. In case of incorrect password, an error message will be displayed. Make sure that when making payments, or transferring funds, the sufficient amount must be present for which the system should verify the sufficient amount. From the setup profile page, the user can navigate to profile help or view privacy policy page.

Furthermore, the bank reacts (give response) to the actions of customer like checking balance, transfer funds, and make payment



Here’s a helpful material if want to know the common mistakes that people make while designing the use case diagram. <https://t4tutorials.com/common-mistakes-of-use-case-diagrams/>