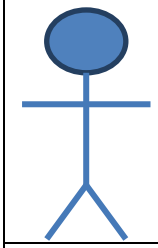
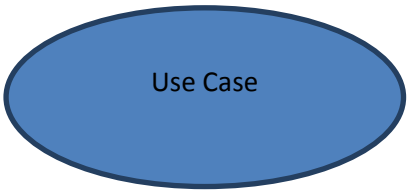
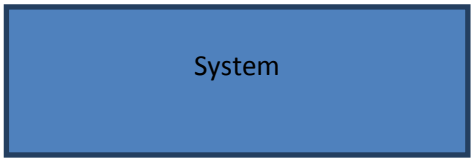
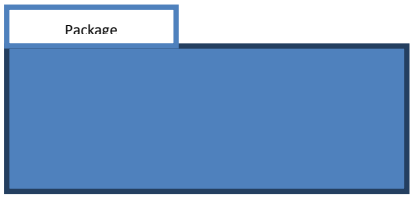
How to create a use case diagram

Use case diagram is very helpful to see the big picture of the usage of the application you are creating. For example if you are making a log in system then we need to know how the user will interact with the system and what the system should provide to the user. This also helps us to find any fault with the logic before continuing further.

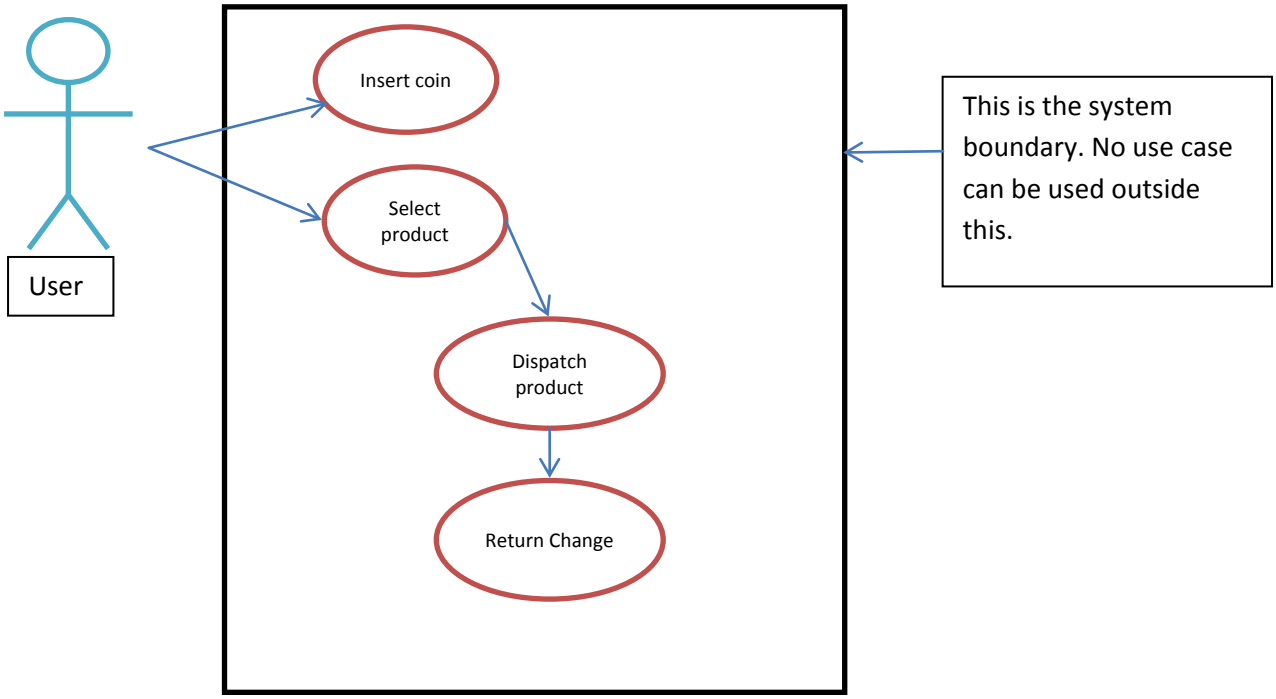
Use case diagrams are unique because they separate the role of the user, administrator and the system itself. It becomes easy for us to see what they system will do for each user whether they are the customer, employee or IT administrator.

Use case diagrams have 4 important parts

1. ACTOR
2. USE CASE
3. SYSTEM
4. PACKAGE

Actor 	This is a real person who will potentially use the application. It can be the Administrator, user or employee.
Use Case 	This round oval shape represents the use case of any software. This means the interaction or process which is happening between the actor and the system. It can be order food, make payments, select colour or click print. Any and all user interaction can be pointed in this shape.
System 	The system is visualised in the BOX. Basically all the use case will take inside this box. The actors are placed outside it.
Package 	Package is used to group classes together and show where it will be used. For example if we have separate class for easy, medium and hard enemies in a game we will group them using the package inside the system symbol. For the beginner level Use case diagrams we will not be using this. The first 3 are the ones we are going to explore in this tutorial; however it is good to know what they do.

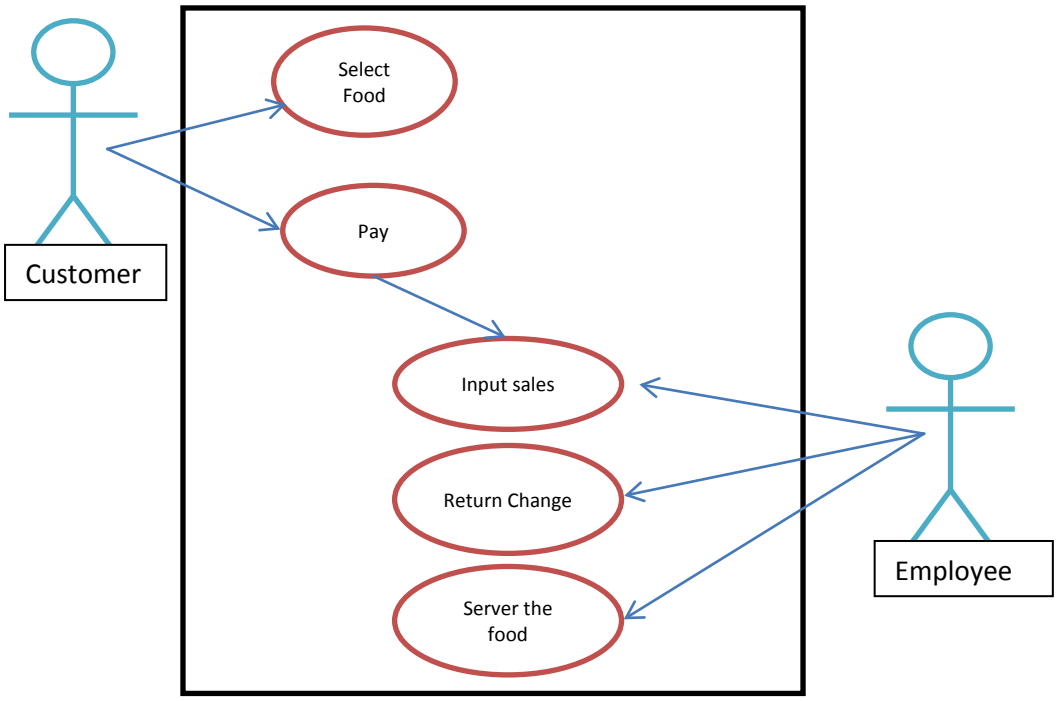
Example 1 – Vending Machine



Here we have a simple USE CASE DIAGRAM of a vending machine. The user here has an option to insert coin and then to choose a product. Once the machine verified the amount and the choice from the user it can dispatch the product and return any change left to the user. Since the system outputs the return to the user we won't need to show an arrow back to the user.

Now let's add have the user buy their food off a counter instead of a vending machine.

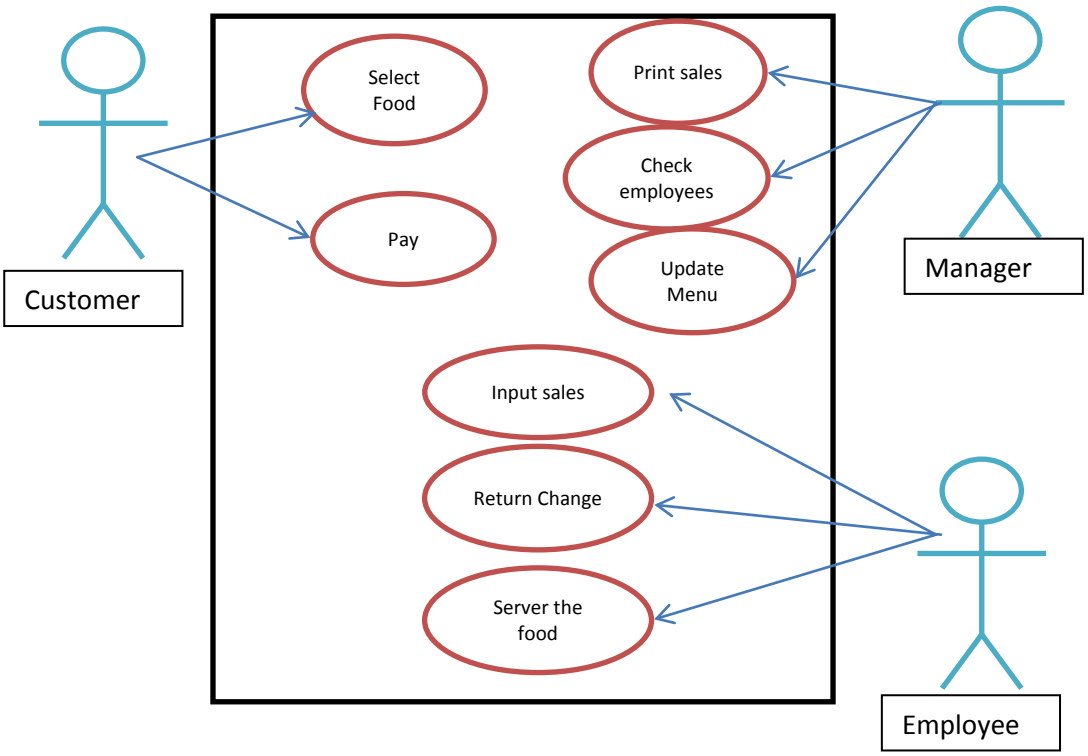
Example 2 – Buying off the counter.



In this example we have two actors using the system. 1 is the customer and 1 is the employee. They both have access the same system differently.

In this case the customer chooses the food he wants to buy and pays for it. The employee will input the sale in the system return any change back to the customer and then server the food. It's pretty simple right.

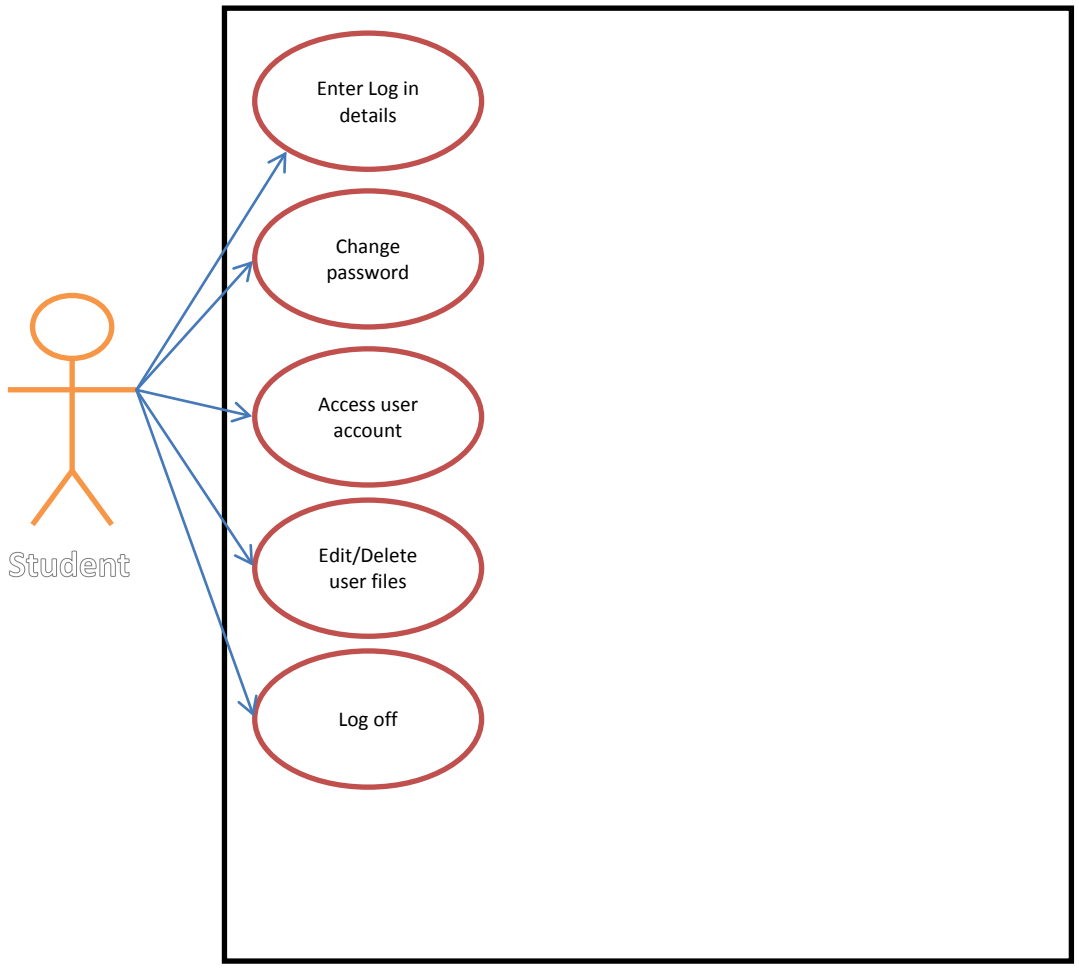
Let's add another actor to the system.



Now we added the manager to the system. The manager can have his own access and use case for this system. He can check and print sale records, check employee performance and update the restaurant menu.

In the example above we have looked at how we can have multiple users undertaking separate tasks in a single system. Now remember none of the use case should leave the system boundary.

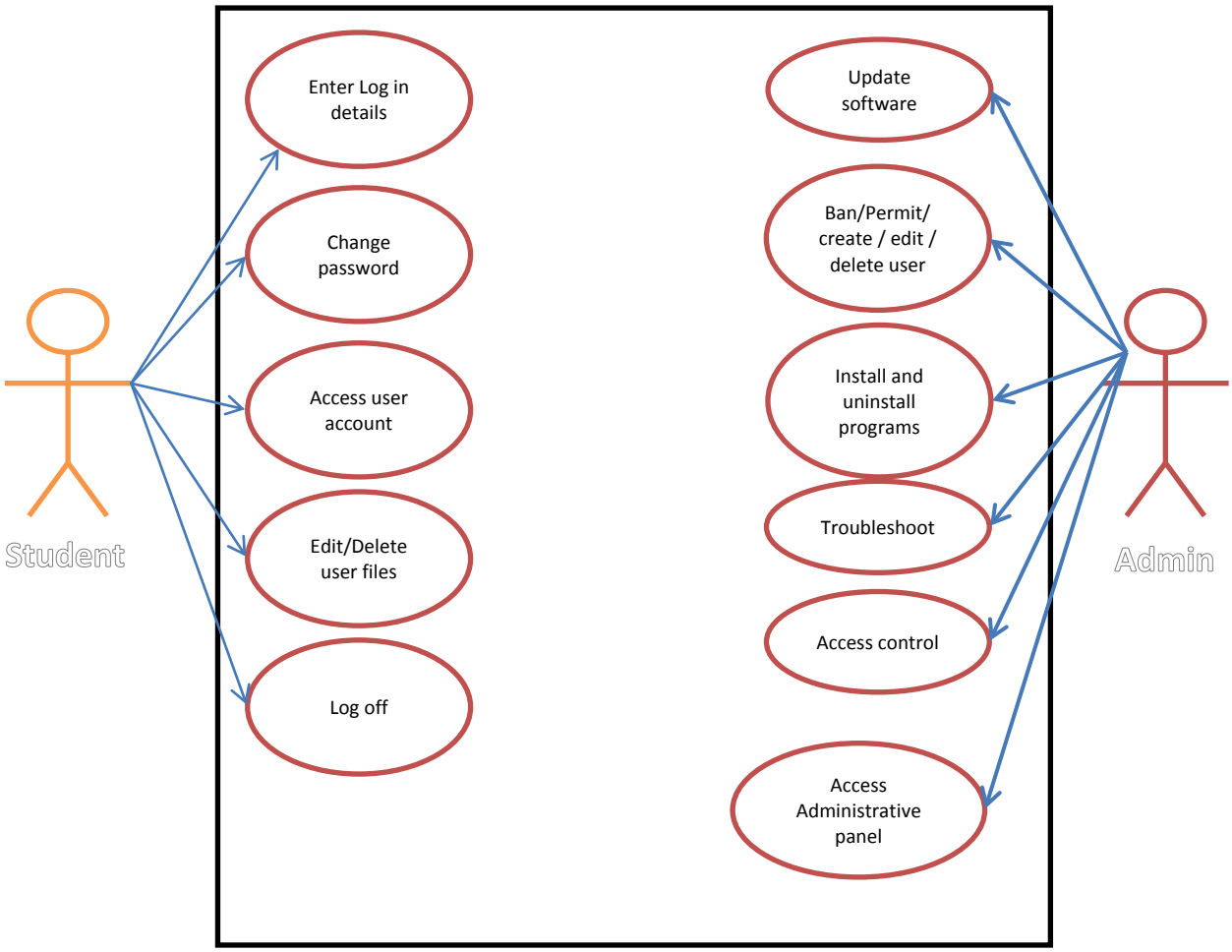
Example 3 – Student and Administrator



In this example we will look at a system which you might be familiar with. It's the school or college student account. We have a student that's accessing their account in this case the user has a bit more control that just pay and receive food.

The student can log in, change password, access their area, edit and delete files. Other than that we have restricted our use from accessing the system files or installing or uninstalling any software from the system.

Let's add the administrator to this system.



Now we have our administrator for the school system. Let's take a look at their responsibilities.

Admin can update the core software to a new version; they can edit the permissions for the user, can create or delete user accounts, install or uninstall programs, troubleshoot the system and have administrative access to other settings.

As you can see by doing UML diagrams it gives us a birds eye view of the system and we can trouble shoot any issues before starting to program it. Now it will be difficult if we need to add any features to the already coded system however if we can spot a feature that we want to add while doing the UML diagram then that can solve the problem with ease.

This tutorial is to cover the basics of UML diagrams. There will be more detailed tutorials on this later on.

Hope this helps ☺