ASSIGNMENT 2 : REFLECTIONS

EMAIL: GREGOIRE.COUSIN@UCDCONNECT.IE
GREGOIRE COUSIN STUDENT NUMBER: 18204188
COMP 47480 PRACTICAL

Work done:

A summary
In the work that my teammate and I have done, we have found that a lot of principles of UML consists of ways you organize your code. This theory implies that you are merely getting closer and closer to your goals as a developer or team. By sectioning off behaviors into boxes and relating them with other ones you can completely map out visually what you intend to accomplish in the end. UML can also be perceived as a method of communication, between you and a small team but also facilitate a large one. In our work, we have found that the more technical you get with the diagrams, the better you are able to gather the details needed at hand to complete the jobs. We were given a set of task to relate a machine with and a shopper and simulate what it would look like if you were to create a Case Modeling diagram of what that would look like. We then were instructed to find the positives and negatives of an example based class modeling diagram. Since UML is like a map to guide you from point a to the last step, it is essential to understand what your constraints are. We then dove into the different methods and approaches of sequence diagrams. This type of layout relates and concentrates on the timing of the project at hand based on behavior and functionality.

Reflections: Why, alternative, preferred, Linkages, areas of computer science, deeper study of some aspects

# Case Modeling
In Case Modeling, the structure is based on simple key points of interaction between two or more sources. It would be easy to take something that we interact with and thoroughly analyze its components, behavior, and functionality into everything it's capable of doing. However, this model gives us the ability to gather data more in an eagle eye point of view. Take the sequence of events that you would need to go through when interacting with an ATM. You could describe all of the things that go wrong along with representing all of the steps. However, with Case Modeling, you need to have the eye of someone external. Instead of explaining what would happen when you type a pin wrong, all you need to describe is that you typed in a pin. Case Modeling in its purest form, are the key of events that occur when dealing with interactions. As if you were someone from an external source looking in.

# Class Modeling
I find going through the Class Diagrams approach to be the most simplistic way to deal with an abstract problem. To me, Class Diagrams are clear and concise. It has the simplicity of working with small chunks as if each class was viewed like a posit note but also the complexity of joining them together to explain the relationship of what's inside a specific class. Another cool aspect of dealing with class diagrams is the fact that we can visually see what goes on inside of an application but also translate it to code through the use of specific software that is based on translating the diagram into code. Class diagrams have useful arrows and other visual tools making it easy for developers to associate classes together. This can facilitate the understanding of a multitude of complex concepts in the software such as inheritance, general relationships, aggregation and so on. Since we have found that object-oriented programming to be beneficial to organizing and building software, Class Modeling attempts to give you a visual way to work with objects, their attributes, and internal functions.

# Sequence Diagrams
In sequence Diagrams, we are less likely to focus on the internal side of who something works the way it does, or how behavior affects one particular item over another in the sequence of events that it would. Sequence Diagram focuses directly on the *WHAT* will happen throughout the behavior of an interaction. This means taking a whole organism and separating it into what it does over time. Take for instance starting a car. When we start an engine, we first have to turn the key to tell the vehicle to activate the motor. In this case, when drawing out what a decency diagram does; we would first need to turn the key and know that the motor will react to start the engine but then relate everything back to the key since it's the interacting tool that got us there in the first place.

# Conclusions
In my opinion, I believe that all three modeling practices are necessary and essential to use as a unit. It would depend on the application at hand or even what part of the application process would need what diagram the most. We have seen OOP be very useful. The class diagram method would resemble OOP perfectly but give you less focus on the sequence of the application. To accomplish this better, adding a sequence diagram would benefit your application. Since these are complex methods, we also need a way to communicate our ideas effectively. Which is where Case Modeling would strive.