**Modelling**

We model to reduce risk and complexity that comes with development.

Allows us to build more manageable representations of complex ideas and on the component base level.

**System Modelling**

We are building an abstract idea of the system.  
With each model represents a different view and perspective.

Modelling also helps analysts with functionality and \_\_\_\_\_.

Model of an existing system, which clarifies the existing system.

**UML (Unified Modeling Language)**

Visual language, object oriented.  
While OMG (object management group).

Use Case diagrams (These are the diagrams we need to know)  
Activity diagram  
…

Consistency and neatness is what is looked at for diagrams.

**Use Cases**

Model use cases based on requirements.

Actors: Entities who interact who use or consume the system.   
-Actors can be users, or even external systems or subsystems.

Use Cases: Individual parts that the system must **do.**  
-Every use case **must** be a verb.  
-Is at **most 3-4 words**.

Grouping: Relationships/associations and Subsystems (components of the main system, when combined together form a whole system).

Use cases do not describe the **how (behavior)**, but **what** is being done. Eg Manage inventory. (not detailed)

**Relationships: Actors and use cases**

Actor acting upon a use case  
-being acted upon

**Generalization: Actors**-Actor A can do whatever actor B can do.  
-The descendant inherits the role from the ancestor.

**Relationship: Use cases and use cases**

All use cases can be independent  
**Include**: to use one use case, another must also be used. (**In semester Test)**

A 🡨includes B :use case B depends on use case A. (for B to occur, A must occur. But A can occur without B occurring)  
 ie B elevator goes up  
ie A elevator button  
A must be pressed, for B elevator to go up.  
But A button can be pressed, and B doesn’t occur.

**Extends:** Optional (supplementary) behavior. Both cases can be independent.  
ie A 🡨extends B : A and **you can** do B (optional). B can be done independently

**Include** is mandatory, **Extend** is optional.

**Subsystems**

System is divided into multiple subsystems  
Contained in **packages**Each subsystem has its own set of actors (and use cases)

**Scenarios**

This focus more on the **how** it happens.  
We specify the Inputs and Outputs. From system and actors.

Things that can happen.

-Extension: This is when an error occurs.

For the Introduction.md (example – also see .svg in pracs):  
  
**#Introudction**  
  
**#Problem Statement**  
The stock management process is manual and it is very slow and we do not like it therefore we must change it.  
  
**#Our Team**  
**#Functionality requirements**

1. The system must be able to record the stock received using a 3rd normal form database setup.
2. The system must be able to track the delivery of the orders received.
3. The system should be able to manage user authentication of its users.
4. The system must generate reports on the amount of stock used.
5. The system must generate notifications when the stock count Is less than 500.

**## User Authentication**1. User must enter credentials that will be verified in the system  
1. The system must capture the user details and hash the passwords  
1. The system must log the user in once the user has been registered on the system.  
1. The system must notify the user that they have successfully logged into the system or that the user has successfully registered.  
1. Users must be given access based on their level of authority (type of user?).

**#Non Functional Requirements**

1. The system must be able to notify customers of a low stock count in order to ensure that slower delivery times are understood.

**Create new file UsecasesD1.md  
  
#Use Case**

**##Stock Management**

![](images/StockManagement.svg) … something about underscore

**### Use Case: Receive Stock**

\*\*input\*\*: Stock Manager receives stock

\*\*output\*\*: The system is loaded with the new stock.

1. The stock manager gets a deliver  
1. The stock is loaded in the system  
  
\*\*Extension\*\*

2a. The system rejects the new stock input  
 1. The system prompts an error message that the number entered was not a num  
 1. The user re enters the stock value

## **User Authentication**

![](images/UserAuthentication.svg)

\*\*input\*\*: Verify Credentials

\*\*output\*\*: The system should verify the user input

1. The user enters name surname, email, password and type  
1. The system will check if the user enters all fields  
  
\*\*Extension\*\*

2a. The system gives an error message  
 1. The system prompts the user to re-enter the name  
 1. The user re-enters the name  
  
2b. The system does not recognize the email and the password is too short.  
 1. The system gives an error message that the email is incorrect  
 1. The user re-enters info

**\*\*input\*\*:** Enter Credentials.

\*\***output\*\*:** User is registered and logged in.

1. User will enter their names, surnames, email and password  
1. The system checks if input is valid  
1. The system registers user as a driver or a client  
1. The user is logged in

**\*\*input\*\*:** User information validation

\*\***output\*\*:** The system will provide a notification message.

1. The user has entered their information and verification has been completed  
1. The system validates the information to the database  
1. The system gives the user a success or failure message.

\*\***Extension**\*\*

2a. The system is unable to validate the data  
 1. The system throws an error message that the information could not be validated  
 1. The system will prompt the user to re-enter their information  
 1. The user enters their information again.