System Documentation

Explains the design of the system

K.P.I. Shenesh Perera

IDM-Negombo  System Analysis & Design

**Contents**

[Overall system explained through Data Flow Diagram 2](#_Toc11778212)

[Access level grantor 3](#_Toc11778213)

[Ratelimiter 4](#_Toc11778214)

[Use case diagram to declare the interactions between **STOCK\_CONTROLLER**, **ASSISTANT\_STOCK\_CONTROLLER** and **SUPPLIER**. 5](#_Toc11778215)

[The entity relationship diagram 6](#_Toc11778216)

# Briefing

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements.

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

IFR Belts requires a stock control system that covers the following requirements:

* Receive information of customer orders
* Print reports of customer orders
* Create sales orders and send them to suppliers in order to satisfy the customer sales orders for the coming month
* Create lists of items that are required to complete a particular customer sale
* Create a daily report of customer orders that have been completed
* Delete customer orders from the system once they have been completed
* The new work flow system should have the following levels of access:
  + - * + Report and update - for the Assistant Stock Controller
        + Report, update and delete - for the Stock Controller
* The new work flow system should be able to print information of customer orders at a rate of 15 per hour

These requirements are taken as functions of the stock control system and are designed, step by step.

# Overall system explained through Data Flow Diagram

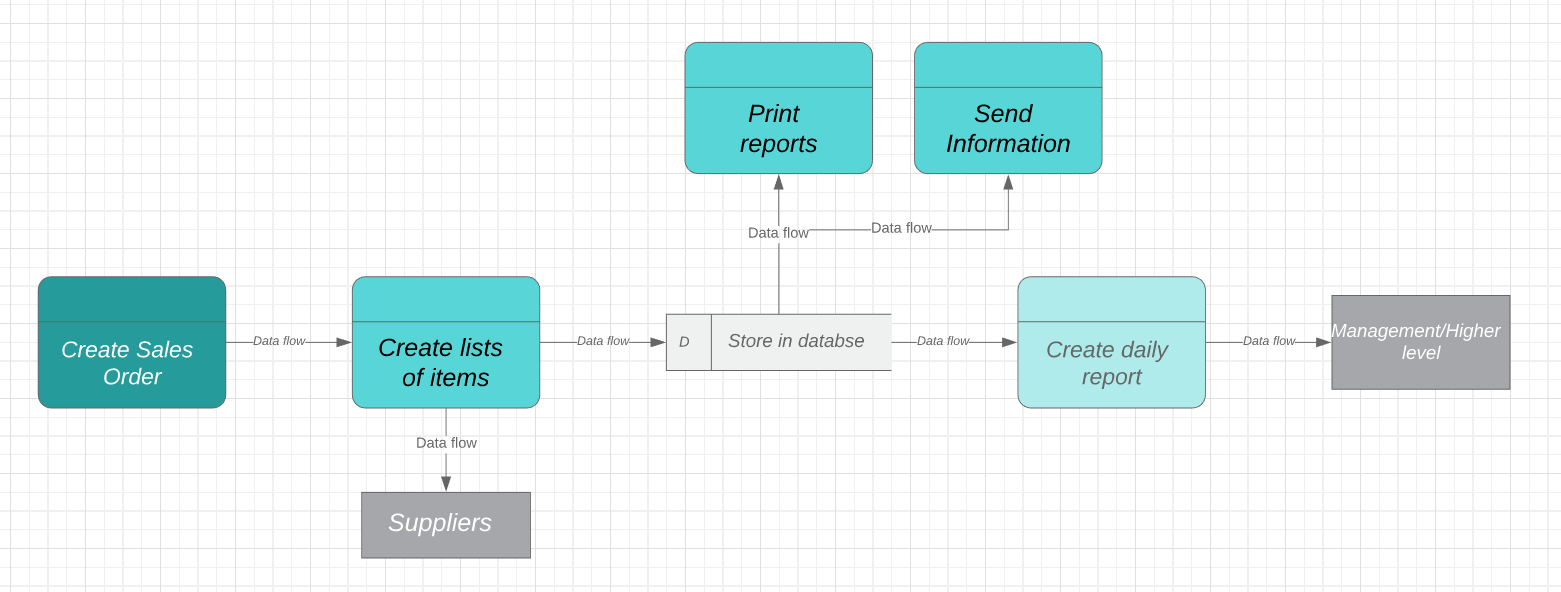


Figure 1.0, Shenesh Perera, 15/04/2019

Initially a sales order is created by the hands of a Stock Controller, this sales order is processed, validated and the data from this process flows to the next process in order to create a list of items that will add itself to the sales order, which is done by the assistant stock controller.

By now a successful sales order must be complete, then it can be sent to the suppliers as a satisfactory sales order. This data then flows to the database, and is stored there for later use.

Three of these later uses are printing reports, sending information and creating daily reports, which are passive tasks and must be performed only when initialized.

When the assistant stock controller wishes to print a report, data flows from the database and the data is processed to print. Then a report is printed.

When the assistant stock controller wants information to send, data flows from the database and through the means of a notification system the data will be processed and presented.

When the assistant stock controller wants to print a daily report a process similar to printing a report is performed, and if wanted the data will be sent to management. This will be a data flow that goes to an external entity, which is the management.

The data flow is sequential, there is no data that flows backwards. Flow is always consistently forward**; a unidirectional data flow is maintained**.

**This makes sure that data is not corrupted in any way sort or form,** this flow also makes it extremely fast and predictable as to where data is and makes the system very less perceptible to errors. As there is no checking involved during the processing of data and flowing of data, the tests that have to be done can be significantly reduced too which leads to a greater performance boost.

This is the complete idea of the design of the entire system put into place, the logic of checking for access level and rate limiting ( 15/hr ) will be expressed through the flow charts below.

# Access level grantor

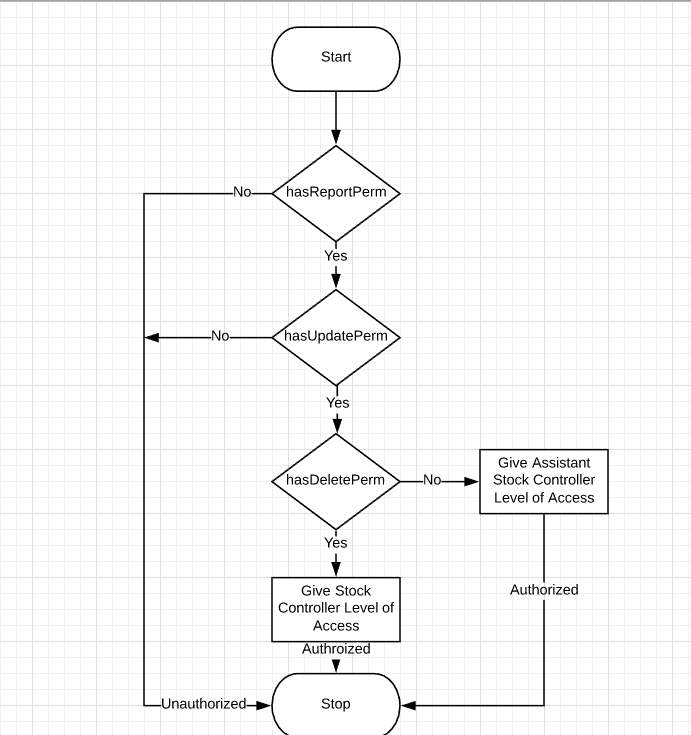


Figure 1.1, Shenesh Perera, 17/04/2018

This is the permission grantor or the access level checker.

1. First the user will be tested if they have the REPORT permission, if they have it then the user is allowed to goto the next level of checking, if not the program exits with an UNAUTHORIZED message.
2. Then the user will be tested if they have the UPDATE permission, if they have it then the user will be allowed to goto the next level of checking, if not the program exits with an UNAUTHORIZED message.
3. Then the user will be tested if they have the DELETE permission, if they have it then they will be granted the STOCK\_CONTROLLER level of access, if they don’t have it then they will be granted the ASSISTANT\_STOCK\_CONTROLLER level of access.
4. After all checks are completed, the user will have a level of access if authorized and the program exits.

# Ratelimiter

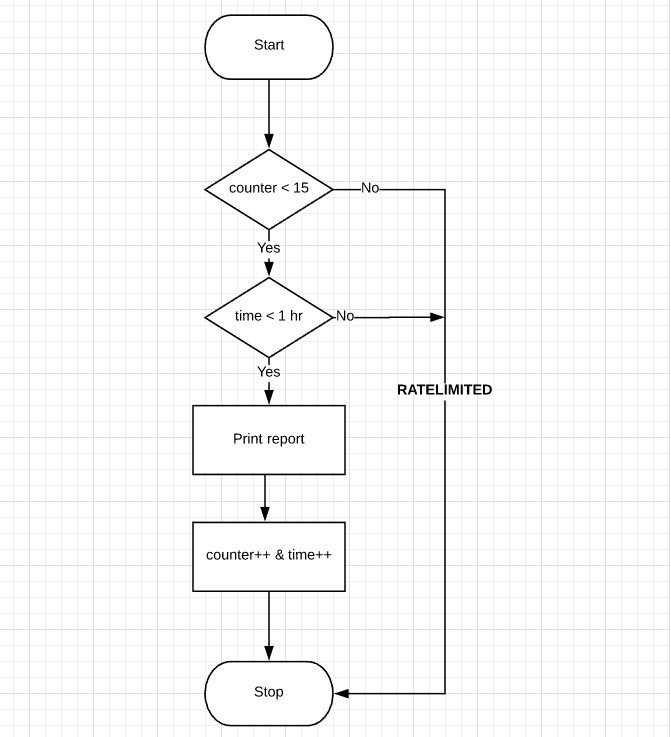


Figure 1.2, Shenesh Perera, 21/04/2018

This is the ratelimiter design, this will decide whether or not a report can be printed.

1. First, the print request is check against the COUNTER to see if the COUNTER value is smaller than 15. If the COUNTER is lesser than 15, then the print request is sent to the next check, if not, then immediately the print request will be rejected and the program exits with a RATELIMITED message.
2. Now the print request’s time is check against the TIME stored, to see if this print request has come within an hour of the first print request in that hour, if it has then the print request is allowed to the next process if not, the print request will be rejected and the program exits with a RATELIMITED message.
3. Since all checks have been passed, the print request will be authorized and the report is printed.
4. The COUNTER value is incremented and the TIME value is appropriately changed, after which the program exits.

# Use case diagram to declare the interactions between **STOCK\_CONTROLLER**, **ASSISTANT\_STOCK\_CONTROLLER** and **SUPPLIER**.

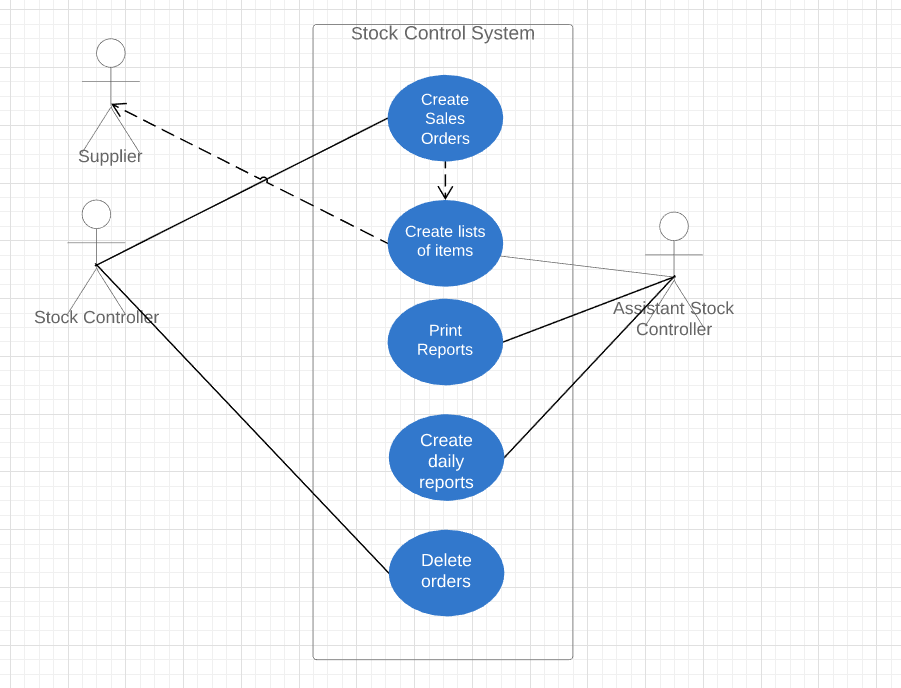


Figure 1.3, Shenesh Perera, 25/04/2019

This diagram will attempt to explain all the interactions between the users of this system. Although the supplier does not partake in any direct interaction with the system, it is a consumer and therefore an external user, which is why suppliers are in this use case diagram.

The actor ASSISTANT\_STOCK\_CONTROLLER is responsible for the following use cases:

1. Creating lists of items for completing custom orders.
2. Printing reports that have information about particular custom orders.
3. Creating daily reports of the custom orders that have been completed.

The ASSISTANT\_STOCK\_CONTROLLER is one of the main actors in the stock control system, because it effectively participates in the system.

The actor STOCK\_CONTROLLER is responsible for the following use cases:

1. Creating sales orders and checking if they meet the satisfactory standards.
2. Deleting already completed sales orders.

The STOCK\_CONTROLLER is one of the main actors in the stock control system, because it effectively participates in the system.

The actor SUPPLIER is only responsible for receiving sales orders from the STOCK\_CONTROLLER, this actor’s roles aren’t clearly scoped into the design as there is insufficient information to make assumptions whether or not they’re part of the stock control system as such it is an external entity is not a main actor of the stock control system.

# The entity relationship diagram

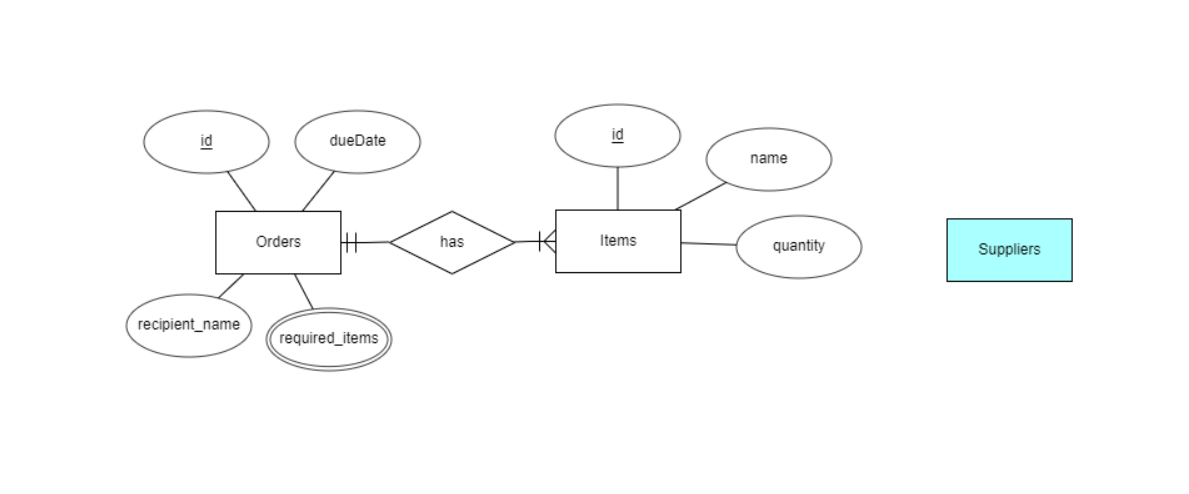


Figure 1.4, Shenesh Perera, 27/04/2019

In the context of this business problem, the database in of the stock control system must by the least have 3 main entities.

The orders entity, that will hold both complete and incomplete sales orders in order to provide the function of creating sales orders and retrieving them for printing or for analysis.

The Items entity, that will hold a list of all the items that the IFR Belts company provides gauged by quantity, so that the sales orders can be successfully created with item lists.

The supplier entity, who is an external entity, which is why it’s highlighted. This entity does not have to be directly stored in the database, as it’s attributes and the like aren’t specifically justified as it’s participation in the system is not directly justified.

The orders entity shares a 1 to Many relationship called [has] (read as Orders has Items), participation is mandatory in both sides. An order can not exist without items and an item can not exist without items.