**PKFOKAM INSTITUTE OF EXCELLENCE**

**Faculty of Science and Technology**

**Department of Computing and Software Engineering**

By Freshman II Students (2021)

**SUPERMARKET BILLING SOFTWARE**

***Project Manager:***

Mr. Herman MEKONTSO

***Project Member Staff:***

SOKOUDJOU GATSING Leopold

TAKOU TENE AURELIEN Fredy

Table Of Content

[PART I: PROJECT REQUIREMENTS DOCUMENTATION 3](#_Toc91493436)

[I- Introduction 4](#_Toc91493437)

[II- Project Overview 4](#_Toc91493438)

[III- Modules 5](#_Toc91493439)

[a) Admin Module: 5](#_Toc91493440)

[b) Cashier Module: 5](#_Toc91493441)

[c) Customer Module: 6](#_Toc91493442)

[IV- Assumption and Dependencies 6](#_Toc91493443)

[V- General Constraints 6](#_Toc91493444)

[VI- Specific Requirements 6](#_Toc91493445)

[VII- Glossary 6](#_Toc91493446)

[PART II: ANALYSIS DOCUMENT 7](#_Toc91493447)

[I- Introduction 8](#_Toc91493448)

[1. Description and Objectives 8](#_Toc91493449)

[2. Purpose of the document 8](#_Toc91493450)

[II- Project from a Functional Aspect 9](#_Toc91493451)

[1. Actors of the project 9](#_Toc91493452)

[a) List of actors 9](#_Toc91493453)

[b) Relationship between actors 9](#_Toc91493454)

[2. Use cases 9](#_Toc91493455)

[a) Identification of use cases grouped by actors 10](#_Toc91493456)

[b) Relationship between use cases 10](#_Toc91493457)

[3. Use Case Diagram 11](#_Toc91493458)

[4. Some overviews 12](#_Toc91493459)

[a) In Online Mode 12](#_Toc91493460)

[b) In Offline Mode 14](#_Toc91493461)

[5. Some Sequence System diagrams 15](#_Toc91493462)

[a) For Products Management 16](#_Toc91493463)

[b) For Users Management 18](#_Toc91493464)

[c) For Modules 19](#_Toc91493465)

# PART I: PROJECT REQUIREMENTS DOCUMENTATION

# Introduction

A supermarket billing system or software is **used to maintain a healthy relationship between the customer and the store managers** by providing transparency in the billing system which validates that exact price has been quoted for the purchases. Managing it manually could be very challenging because one will have to make on a daily basis and physically an inventory of the supermarket, and certainly maintain some books which hold information about the status of the resources in the supermarket. Maintaining the supermarket using only those books can be very cumbersome and studious. Therefore, getting a solution allowing to efficiently manage a supermarket can really be beneficial to an Organization. The purpose of this project is to provide an organization with a software allowing an efficient management and billing of this supermarket.

# Project Overview

This section will provide the reader with an overview of the project. To do it, we first present an overview of the different modules which will compose our application.

Some advantages of this application are:

* The system reduces much of human efforts in calculating bill especially for huge products.
* Saves money and resources of organization and excludes of use of paper or sheets in making bill.
* Saves time.
* It provides accuracy and faultless in billing calculations.
* It is flexible and user-friendly.

To achieve these, we separated the application into the following modules: **Admin module**, **Cashier Module** and **Customer Module**. These modules are implemented under different modes: **online mode** and **offline mode**. The online mode makes use of the Admin and Customer while the offline mode makes use of the Cashier and Admin Module.

# Modules

This section will give the reader more details about each module of our project.

### Admin Module:

General manager of the supermarket stock. He is the person who has the right to:

* Add a new product to stock.
* Set the quantity of a product.
* Set the price of a product.
* Access the list and references of all the cashier except their password.
* Access the list of all customer and their references except their password.
* Set discount on product.
* Delete/remove a product from the stock.

He will own an ID and a password and will be able to access his references once connected and modify them.

### Cashier Module:

He is in charge of:

* Registering the customer product with the quantity purchased by the customer.
* Generate bill. The bill will contain the following info:
* Cashier name.
* Date and time.
* Different items, their quantity and unit price.
* Supermarket name and address.
* Greeting word.
* Number of the bill (ID).
* Display the bill to the customer.

He will own an ID and a password and will be able to access his references once connected and modify them.

### Customer Module:

He will sign in by submitting his information. He will have the possibility to once connected to still access his references and to modify them if he wants. He will be able to view all the products under each category.

He can perform the following action:

* Purchase an item (add to Cart).
* Delete item from the card.
* Access the description of a product.
* Access his account information.
* Generate bill.
* Access his bill information through his ID.

# Assumption and Dependencies

We did not make any assumption nor dependencies.

# General Constraints

It is not possible to add nor delete more than one kind of items at a time from the cart or from the database.

# Specific Requirements

This section of the document lists specific requirements for the Supermarket Billing Software. Requirement: System requirement

System requirement:

To be able to use this software, the computer in which it will be executed should have:

* Windows OS/ Linux OS
* Java (Java 5 at least)
* An Anti-virus to ensure the security of the software (if the OS used is Windows)

# Glossary

ID: Identifier

OS: Operating System

# PART II: ANALYSIS DOCUMENT

# Introduction

## Description and Objectives

A Supermarket Billing software is a software that helps to make purchases, create bills more efficiently and more accurately for customers, manage products in stock, manage customer and cashier information.

The objectives of the product we are proposing are:

* Facilitate the management of stock.
* Facilitate the management of customers.
* Create bills faster and accurately.
* To offer the opportunity for a customer to buy a product without being in the supermarket (Online mode).
* Save money and resources of organization and exclude the use of paper or sheets in making bill.
* The system reduces much of human efforts in calculating bill especially for huge products.

## Purpose of the document

The purpose of this document is to present our product from a functional viewpoint. It gives us all the information necessary to design, develop and test the software. It mainly helps answering to the following questions: what the software will do, that is what are the functionalities that will be offered by the software? Who will be the people who will be interacting with the software and how they will be doing it? We will provide a use case diagram, some diagrams for the project overview and some sequence system diagrams to answer to these questions.

# Project from a Functional Aspect

## Actors of the project

An Actor is a person or group of person physic or moral which interact with a use case (system function). Generally, an actor is a person who behaves as a participant, who takes part of a process. For our case, an actor is a person or a group of persons that represent a role that interacts with the system. An entity is considered as an actor if and only if he executes at least one use case (to be defined in the next paragraphs).

### List of actors

In our project, we have identified 3 primary actors:

* Admin: General manager of the supermarket and present in both online and offline mode, is responsible for managing the customer and cashier, manage products in stock and manage prices of products.
* Cashier: present only in the offline mode, is responsible for registering the customer products and creating the bill.
* Customer: present in the online mode only, is responsible for making purchases and creating his on bill.

### Relationship between actors

There is no direct relationship between actors of the System.

## Use cases

In software and systems engineering, a **use case** is an action or an event step, typically defining the interactions between a role (known in UML as an actor) and a system to achieve a goal**.**  A **use case** describes a functional requirement of a software. A functional requirement is an operation that the software has to provide. Therefore, providing and explaining all the use cases is the same thing as exhaustively answering to the question: what the software does? From our point of view, the best way to present them is to group them by actors when there are many actors involved.

### Identification of use cases grouped by actors

This section will provide the user with the different functionalities offered by the software (representing as use case) grouped in functions of actors and modules. There is a functionality which is transversal to all the roles: login. This is just to note that each actor of our future system will need to authenticate before proceeding.

|  |  |  |
| --- | --- | --- |
| Modules | Actors | Use cases |
| Admin Module | Administrator | -Chose mode  -Login  -Manage product  -Set Up profile |
| Cashier Module | Cashier | -Chose mode  -Login  -Register product  -Generate bill  -Print bill  -Set Up profile |
| Customer Module | Customer | -Chose mode  -Login or Sign Up  -Generate bill  -Print bill  -View Products  -Add Item to Cart  -Delete item from cart |

### Relationship between use cases

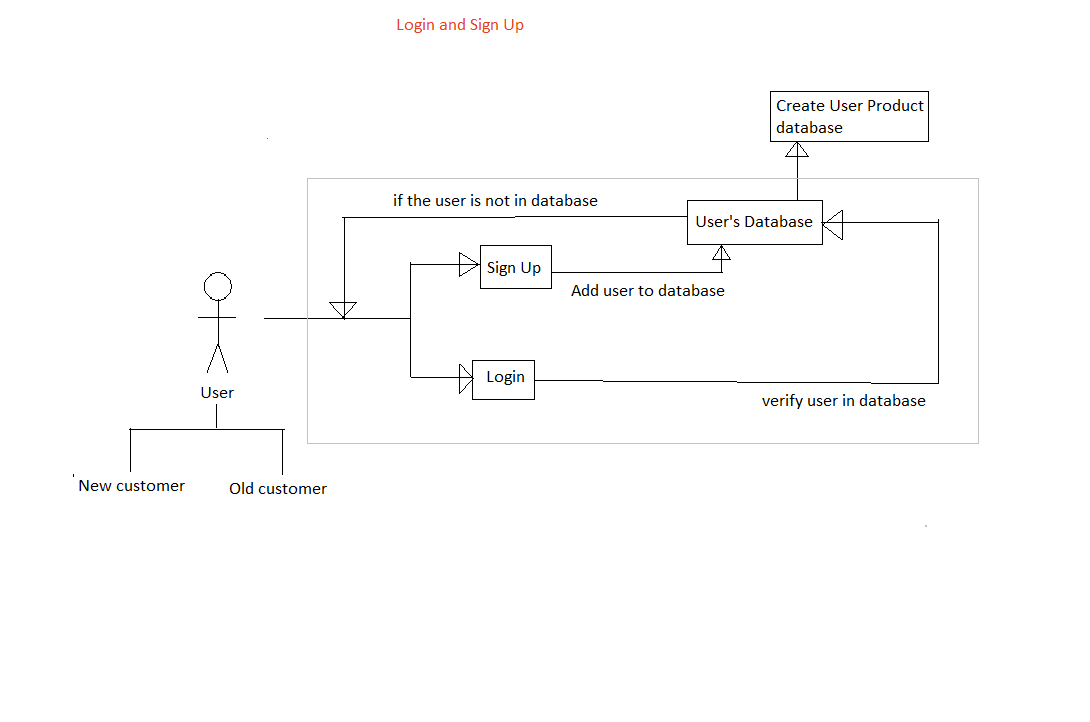
For our project, we will just consider two types of relationships between use cases:

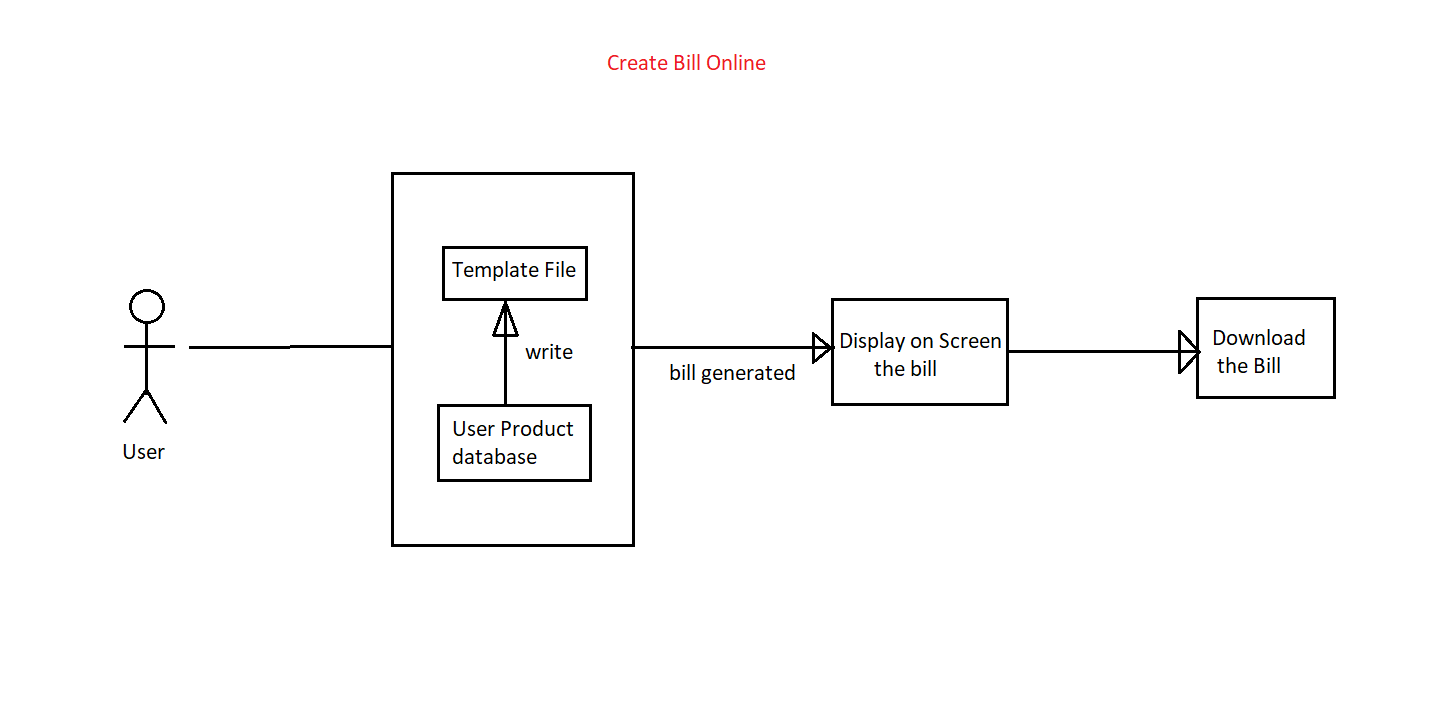
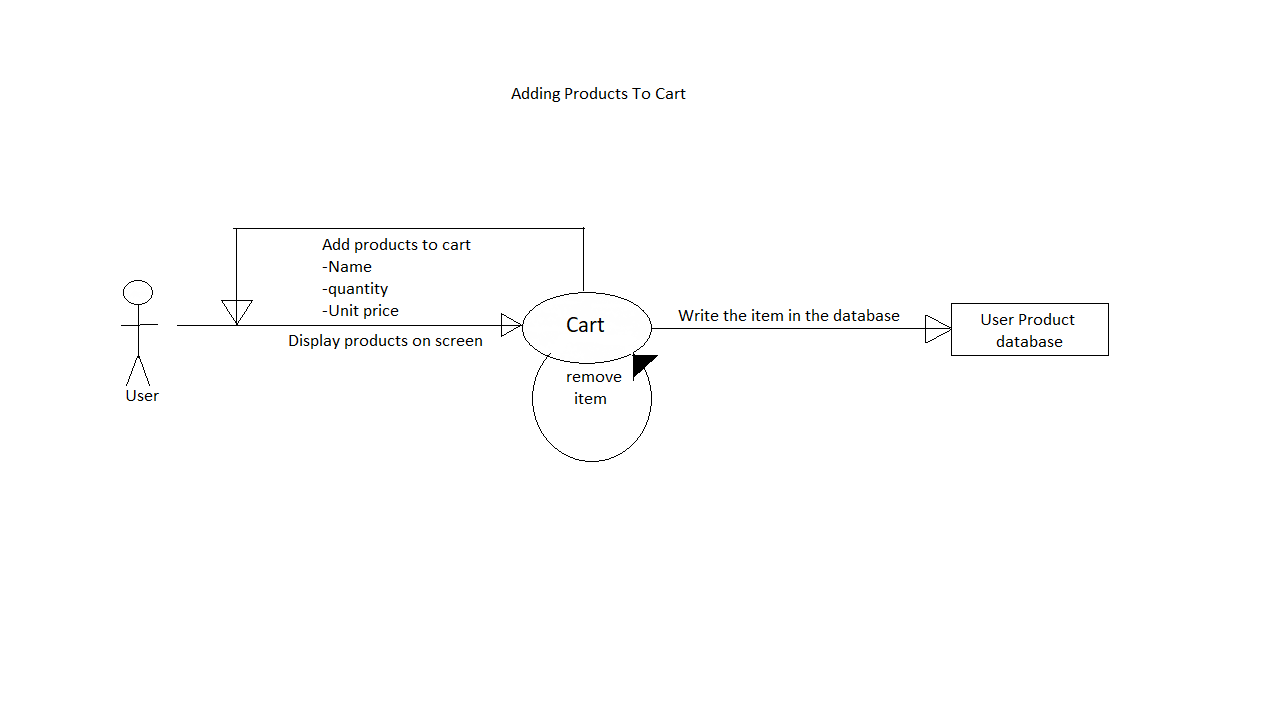
* Include: this relation means that for a use **case A** which **includes** a use **case B**, the use **case B** is **always performed** when the use **case A** is executed.
* Extends: this relation means that for a use **case A** which **extends** a use **case B**, when **certain conditions** match on use **case A**, the use **case B** is executed.

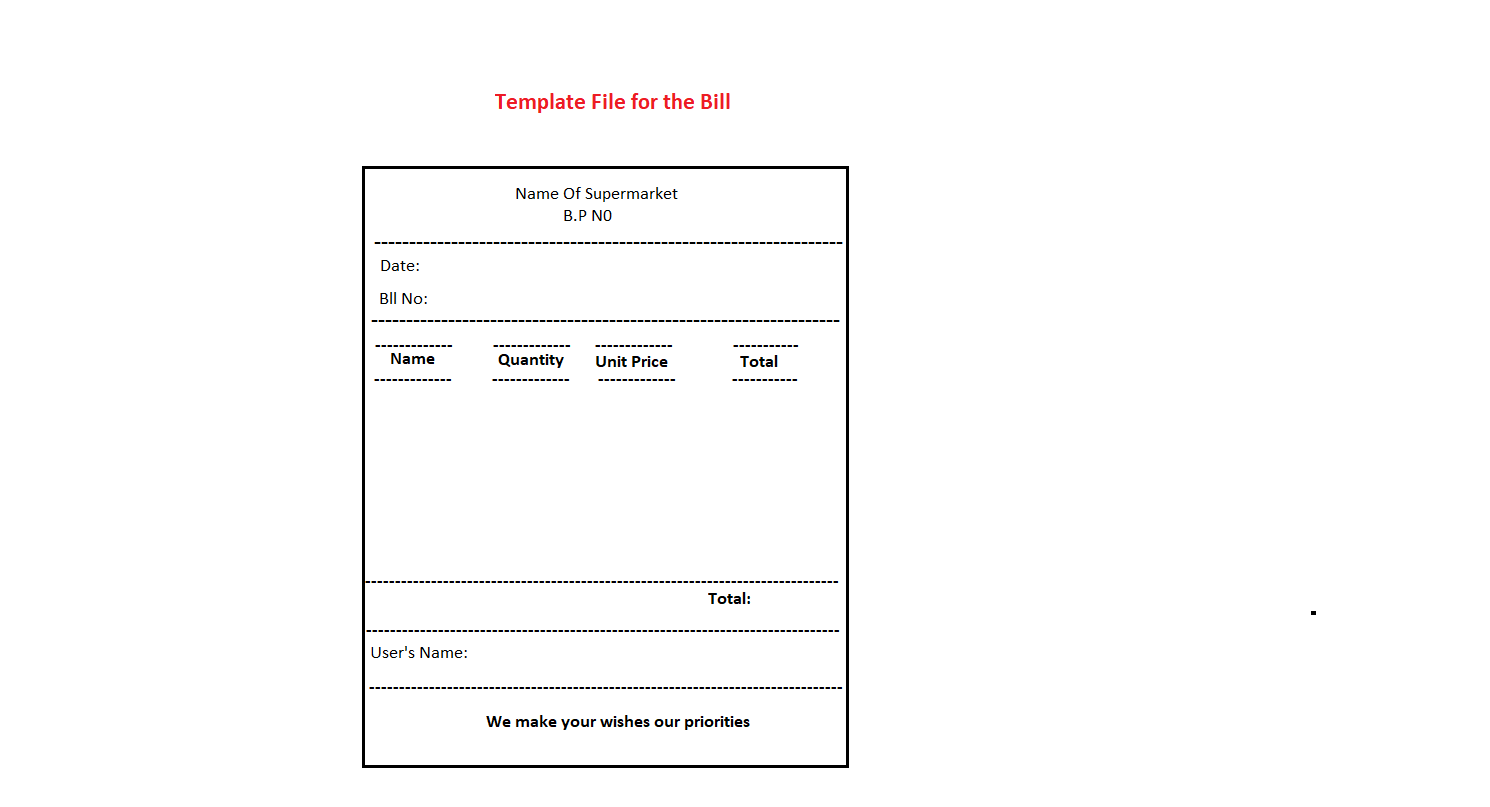
## Use Case Diagram

## Some overviews

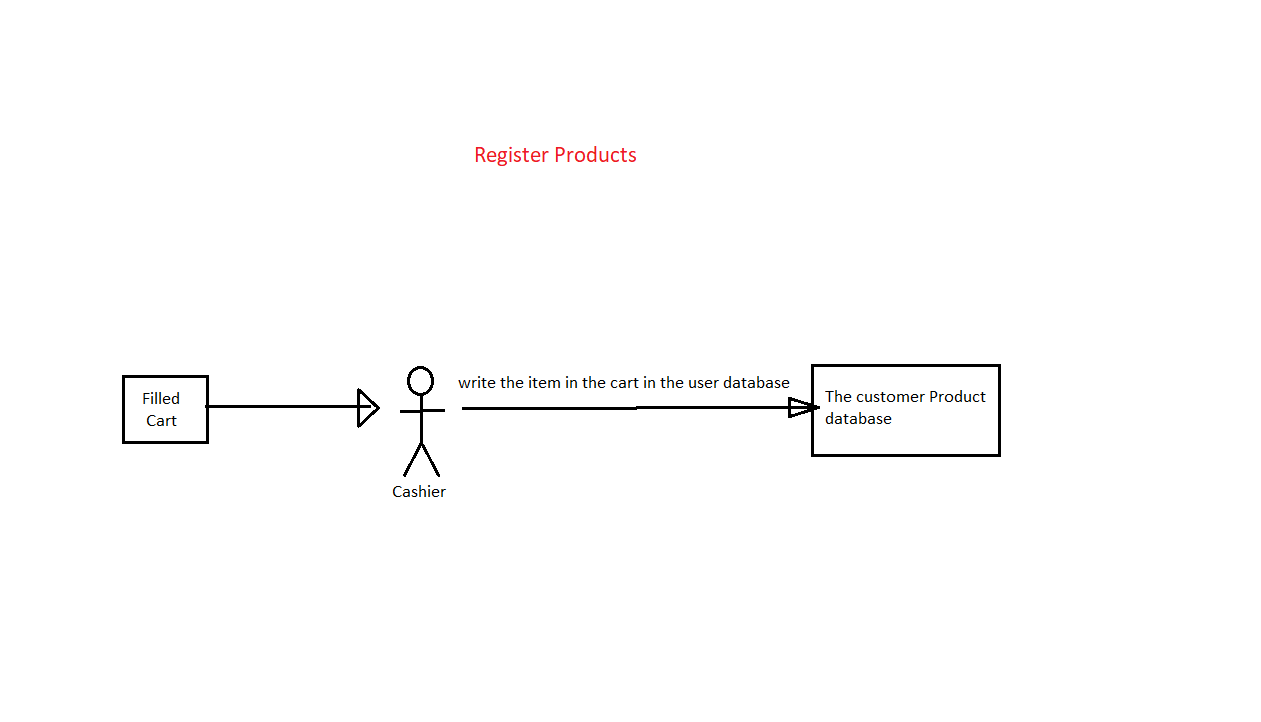
### In Online Mode

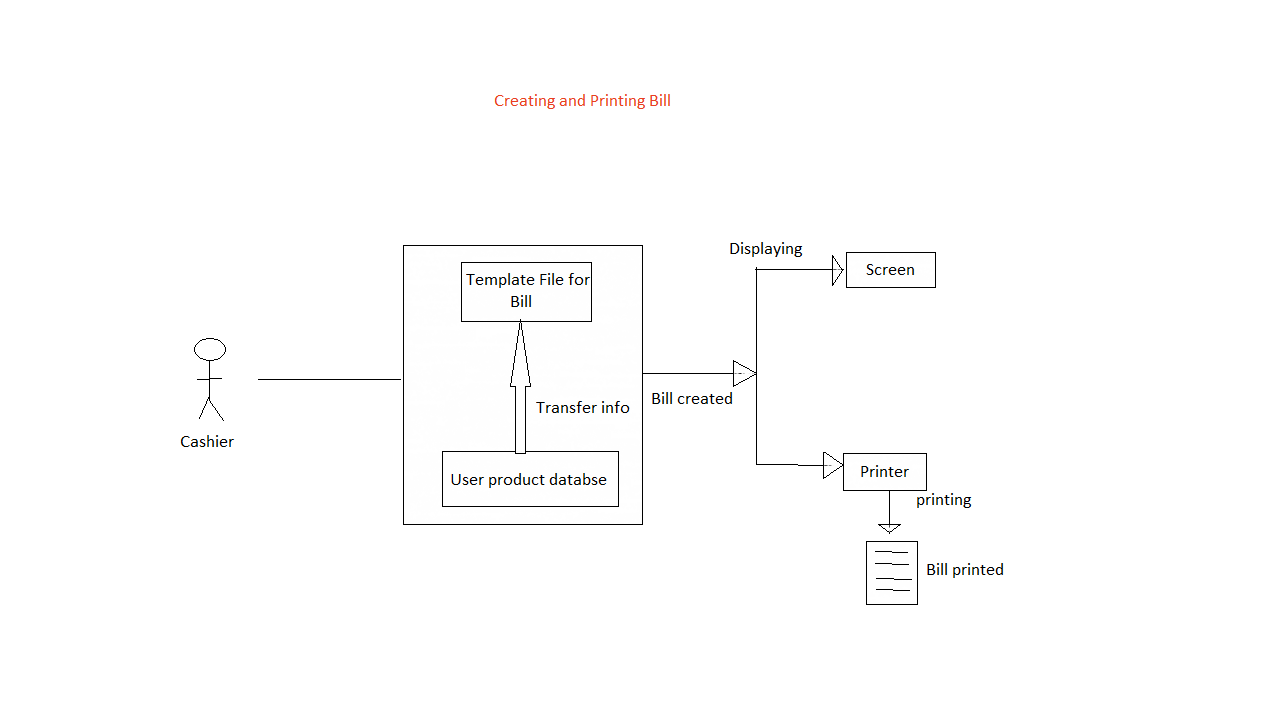
****

****

****

### In Offline Mode

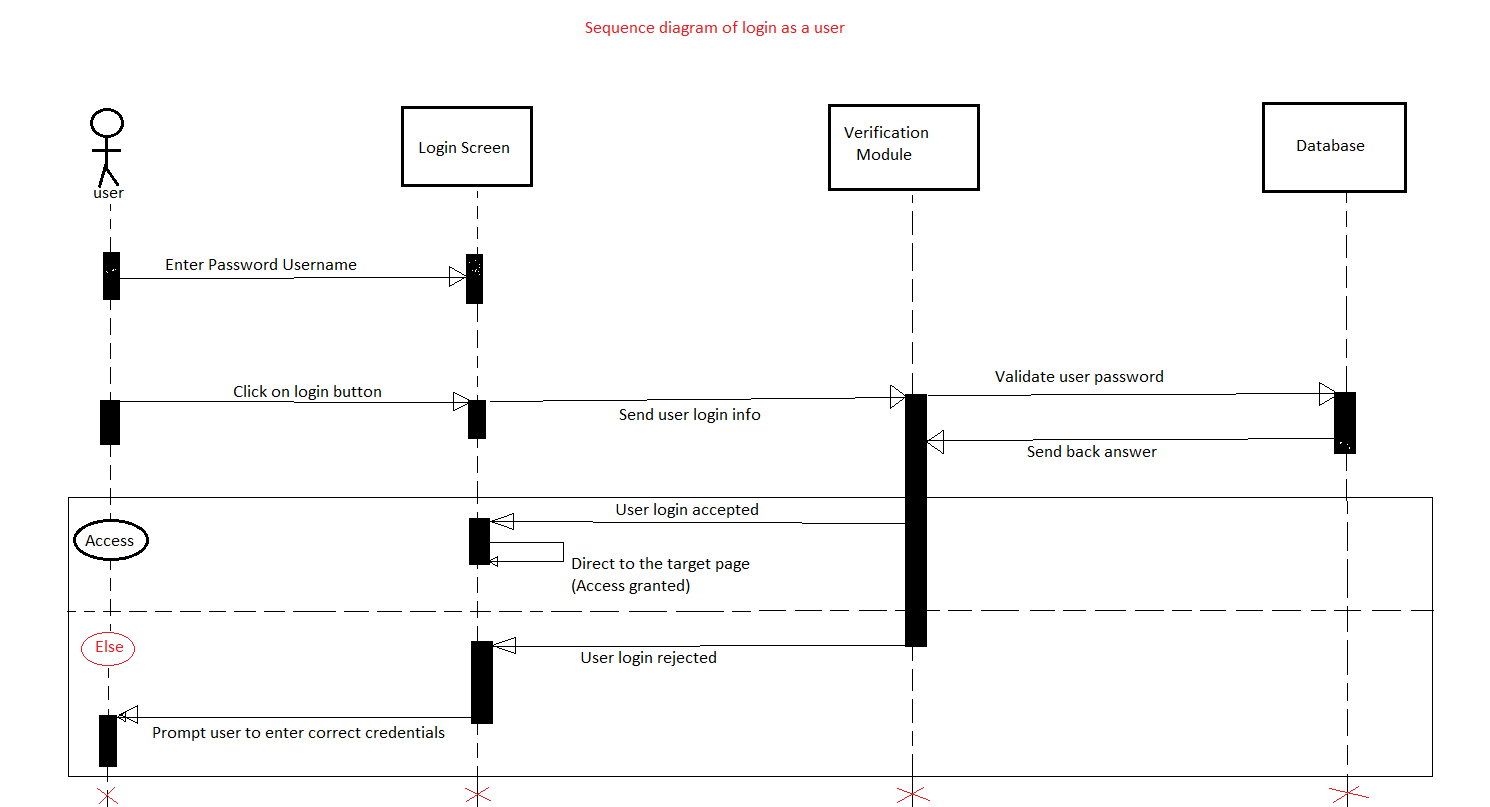
****



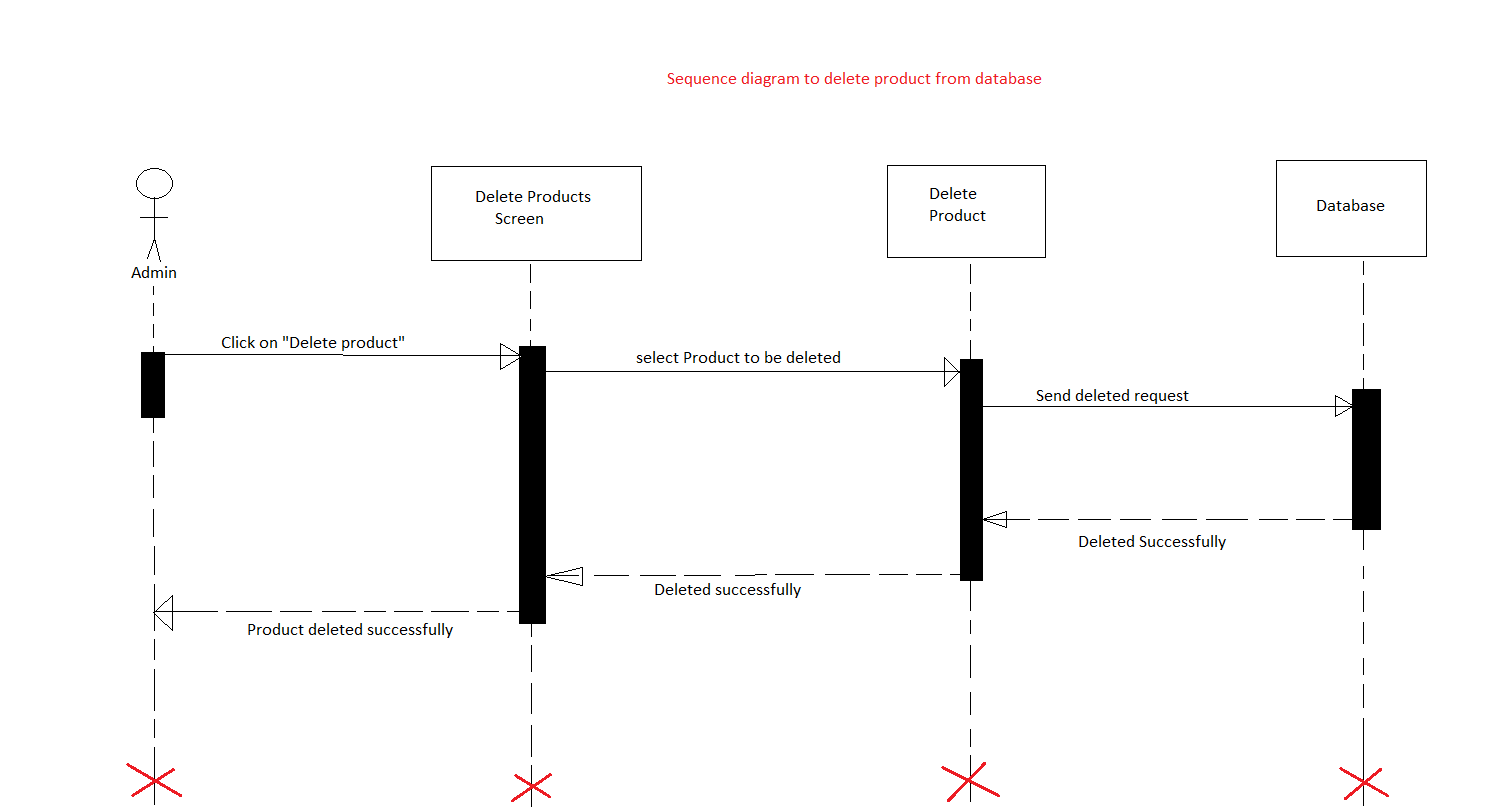
## Some Sequence System diagrams

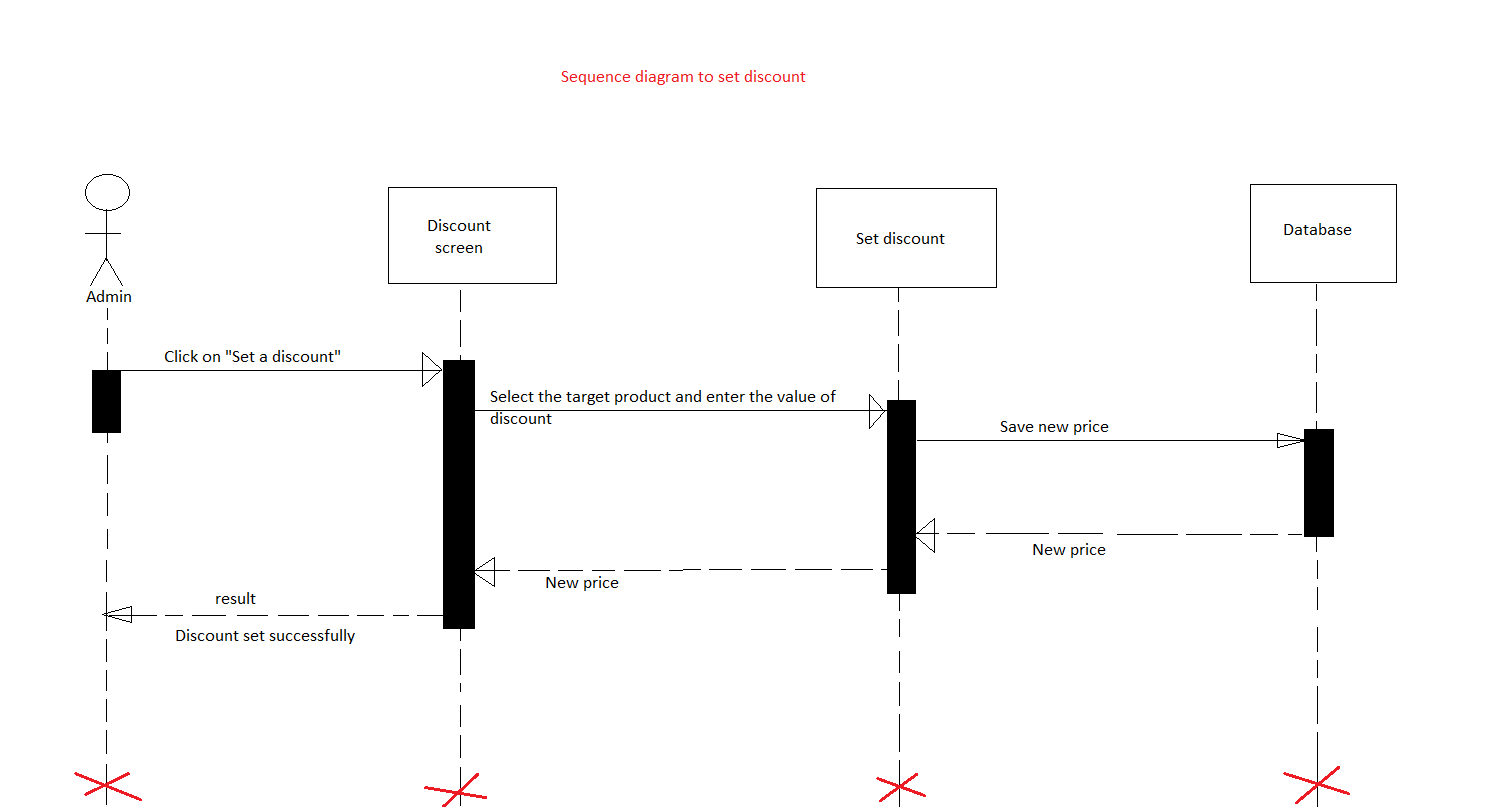
A **sequence diagram** or **system sequence diagram (SSD)** shows object interactions arranged in time sequence in the field of **software engineering**. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the **logical view** of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a **black box**; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

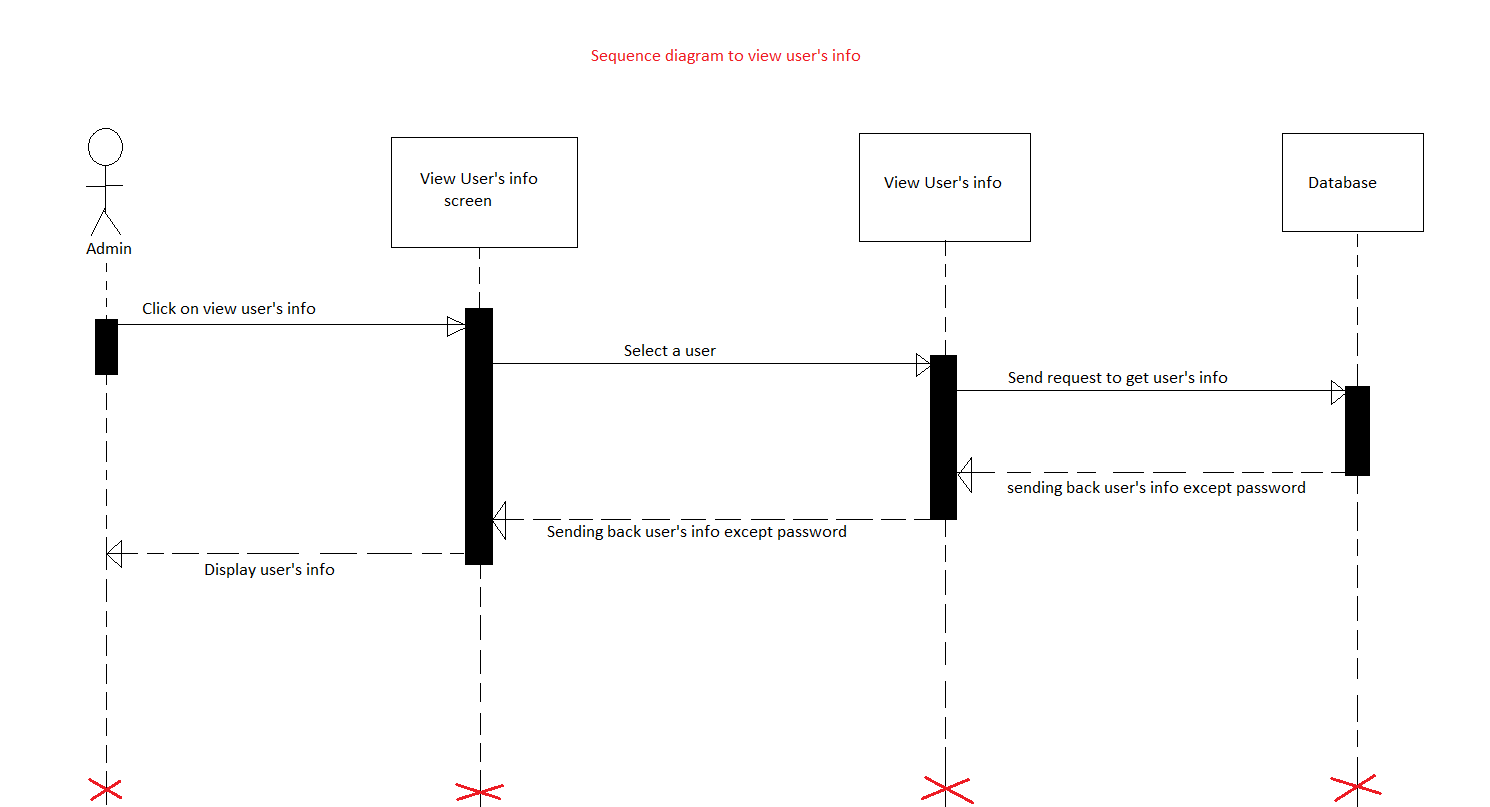


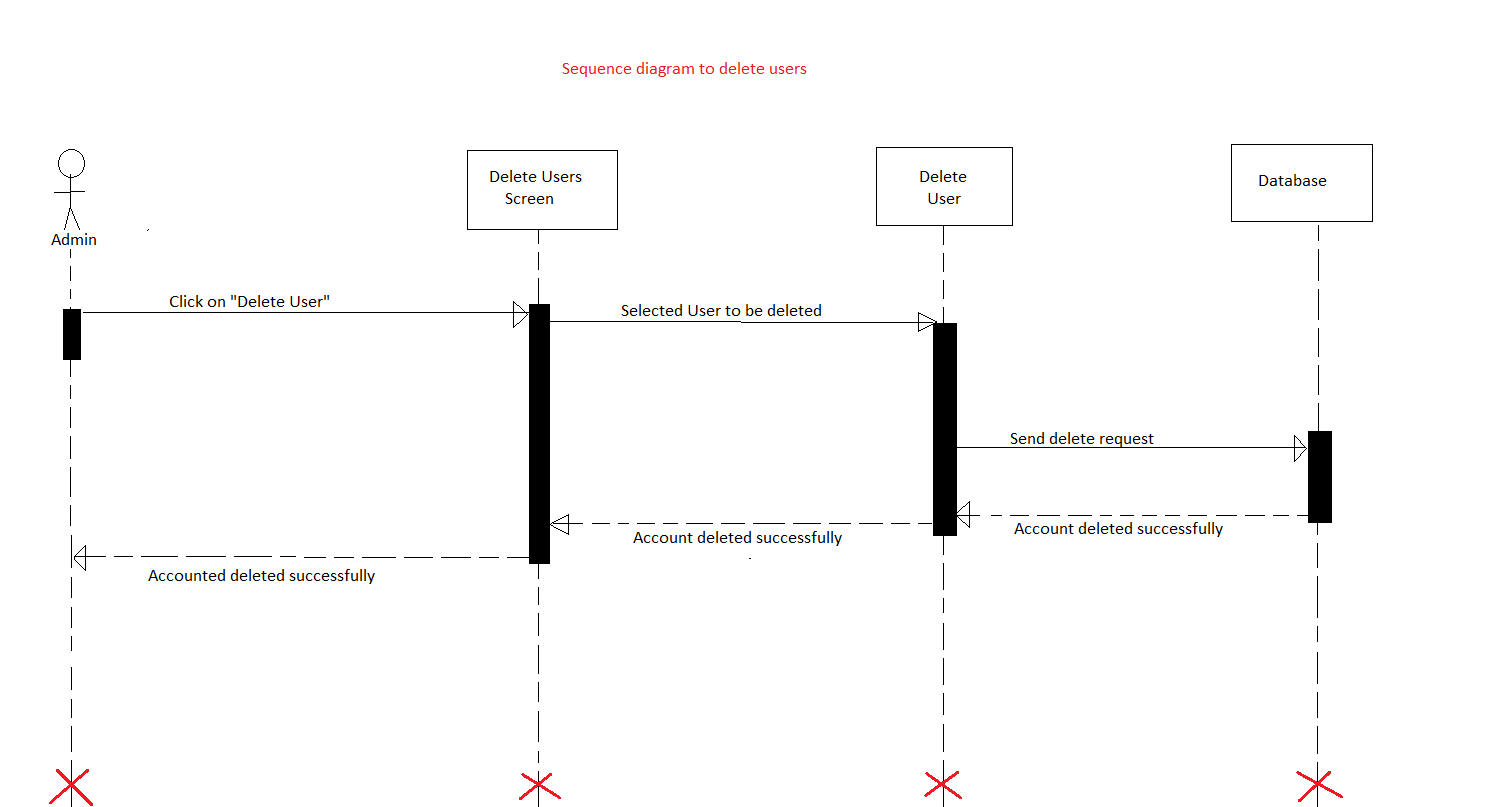
### For Products Management

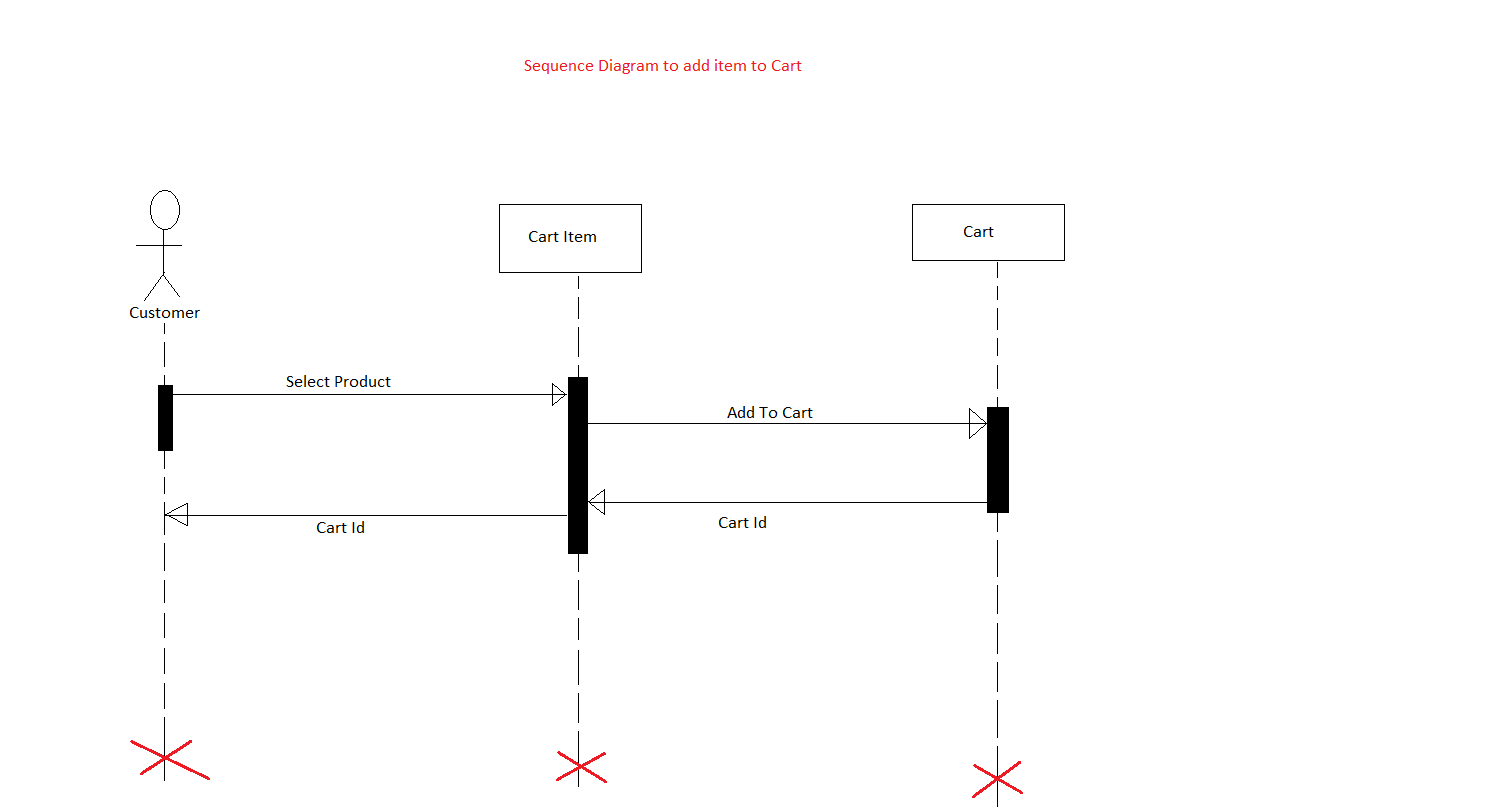




### For Users Management







### For Modules

