

Store Management System

(USER MODULE AND REPORT)

Table of Contents

1.Introduction

- 1.1 Project Summary
- 1.2 Purpose of the Document
- 1.3 Scope
- 1.4 System Purpose
- 1.5 UserModule
- 1.6 Technologies to be used
- 1.7 Functional Requirements
- 1.8 Non-functional Requirements

2.SYSTEM DESIGN

1. Introduction

1.1 Project Summary

The Store Management System is a Application is intended to provide complete solutions for customers. It will enable a customer to browse the products and purchase them online without having to visit store physically. We have five modules those are

- * User Module
- * Inventory Module
- * Payment Module
- * Invoice Module
- * Reports and sales dashboard

1.2 Purpose of the Document

The purpose of this document is to make easy to understand more about this web application and features of this application. In this document everything is in-detail about this application and it acts like a guide to the customer and also developer.

1.3 Scope

In this project user module having the user and admin and guest are the three modules.in this the number of users can be registered by using the valid user name and password.And all the members are viewing the products and purchasing the products and also give a rating to the products.Admin can having the all the rights in the project.like adding ,deleting,updating products etc..

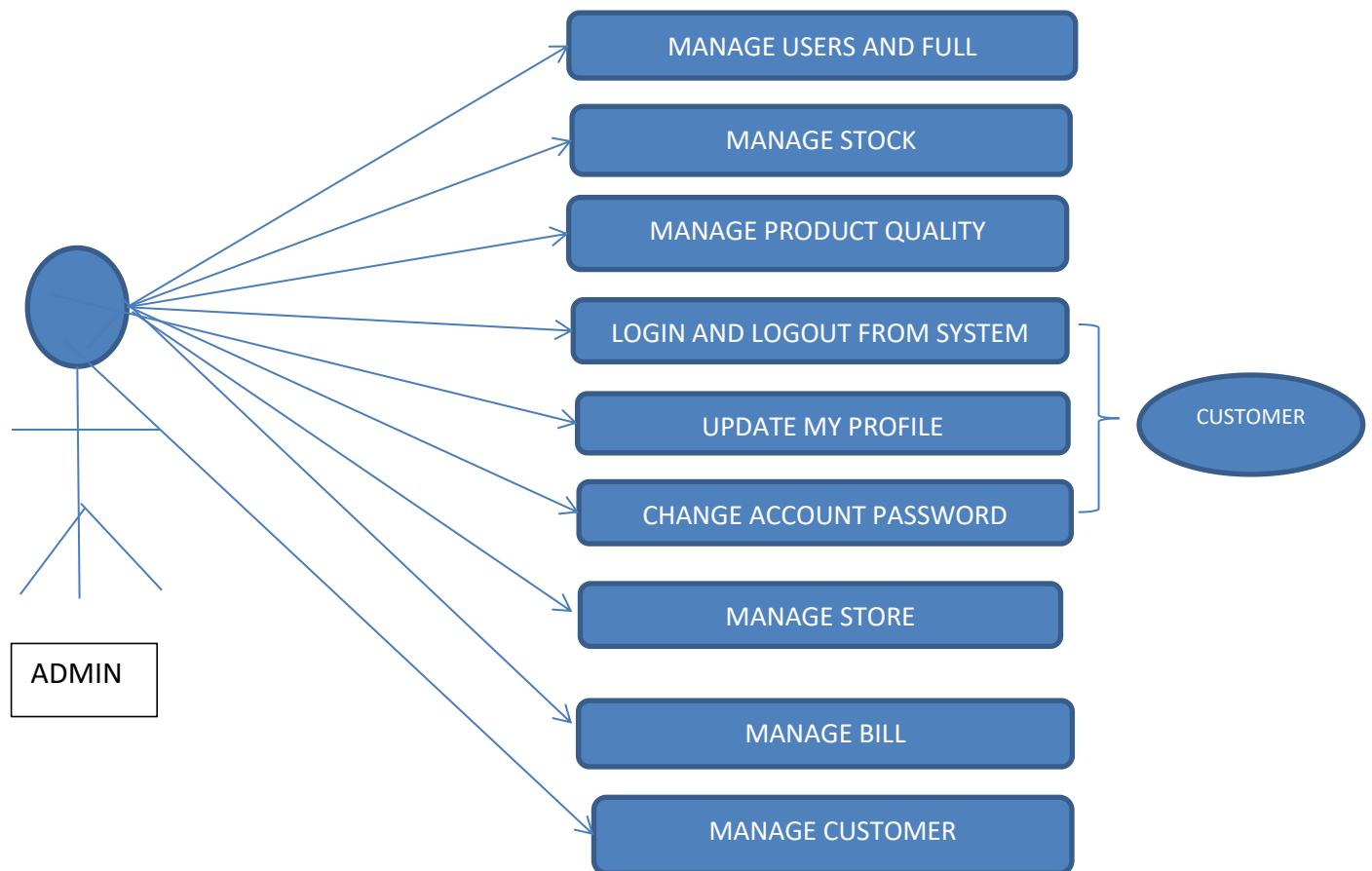
1.4 System Purpose

This Application is a virtual store where visitors come to this website and pick the product by their choice and if they want purchase them. This application is promising and trustworthy towards the customer and also online sellers can get wide range of customers. It saves time for the customers, no waiting in traffic to buy small things, no need to go to market. In this application everything in one place, user can purchase products online.

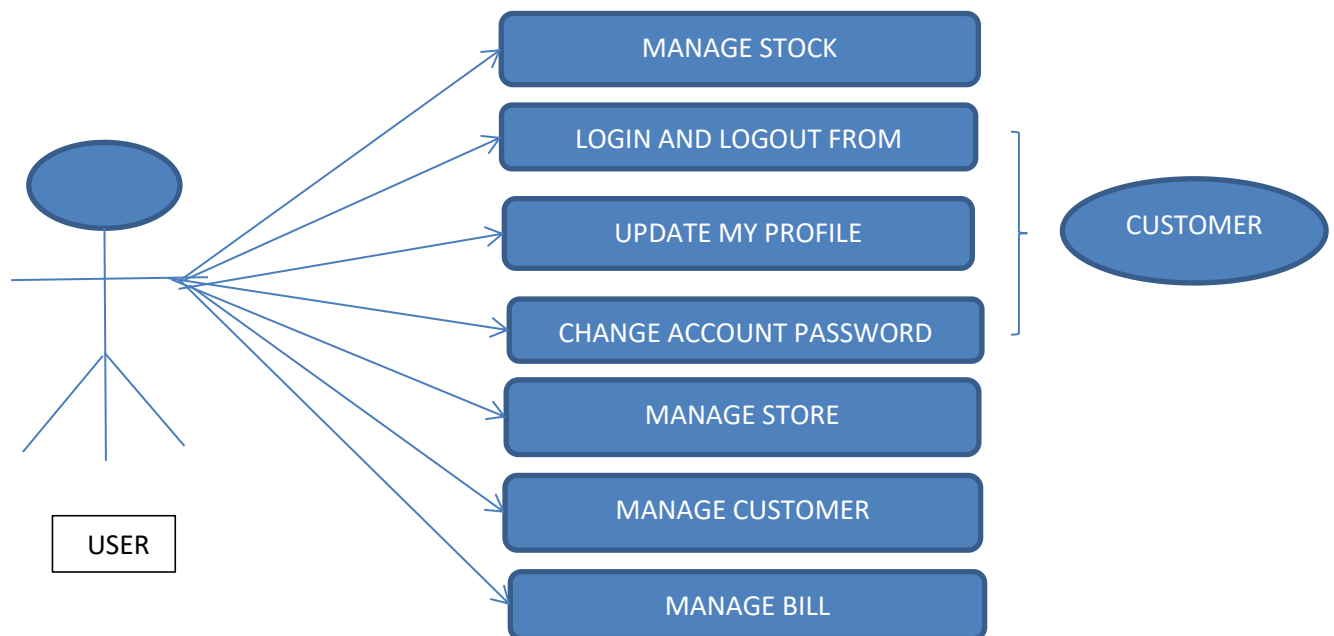
1.5 USER MODULE

In this application we have 3 types of users

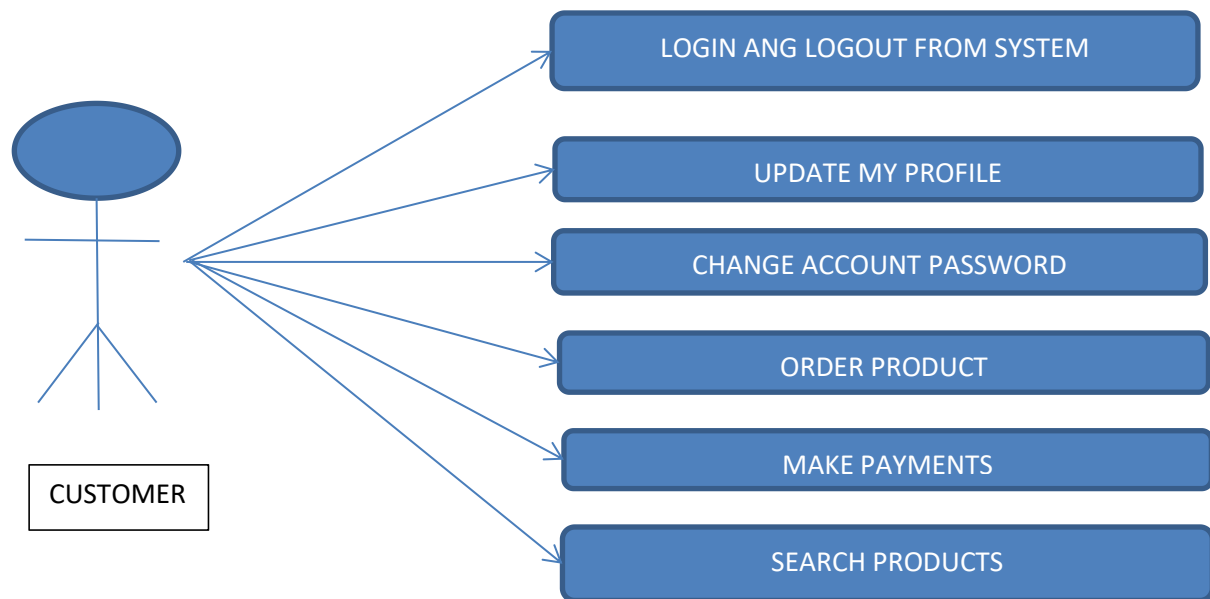
Admin: Admin logs in to the system and manage all the functionalities of store management system.admin can add,edit,delete and view the records of stock ,product quality,customer ,login. Admin can manage all the details of product bill,store.Admin can also generate reports of stock product,product quality,bill,customer,store,admin can search the details of product customer ,store.Admin can apply different level of filters on report of stock bill,customer,Admin can traks the detailed information of product,product quality ,bill,customer.



User: Many number of users can be registered. And log in by using the valid username and password. A registered user can browse the products and purchase products. Registered user get availability on offers and user can change their details.



Guest: Guest also register by valid user name and password. And viewing the all the products in the store. An unregistered user can also browse the products and purchase the products but unregistered user can't avail the offers because he/she using guest mode.



1.6 Technologies to be used:

- * Windows 10
- * Java
- * Apache Tomcat Web Server
- * Oracle

1.7 Functional Requirements:

- * If any product returned within the provided time then refund initiated.
- * User can also view the History.

1.8 Non Functional Requirements:

- * Payment Process is secured to purchasing the product.
- * User Bank Account details maintained securely.

2.0 SYSTEM DESIGN

1. Class Diagram
2. UseCaseDiagram
3. Dataflowdiagram

1.ClassDiagram:

Store management system class diagram describes the structure of a SMS(store management system) classes,their attributes and operations or methods and the relationship among the objects.the main classes of the SMS are stock,product,product quality,bill,customer,store.

Classes of SMS class diagram:

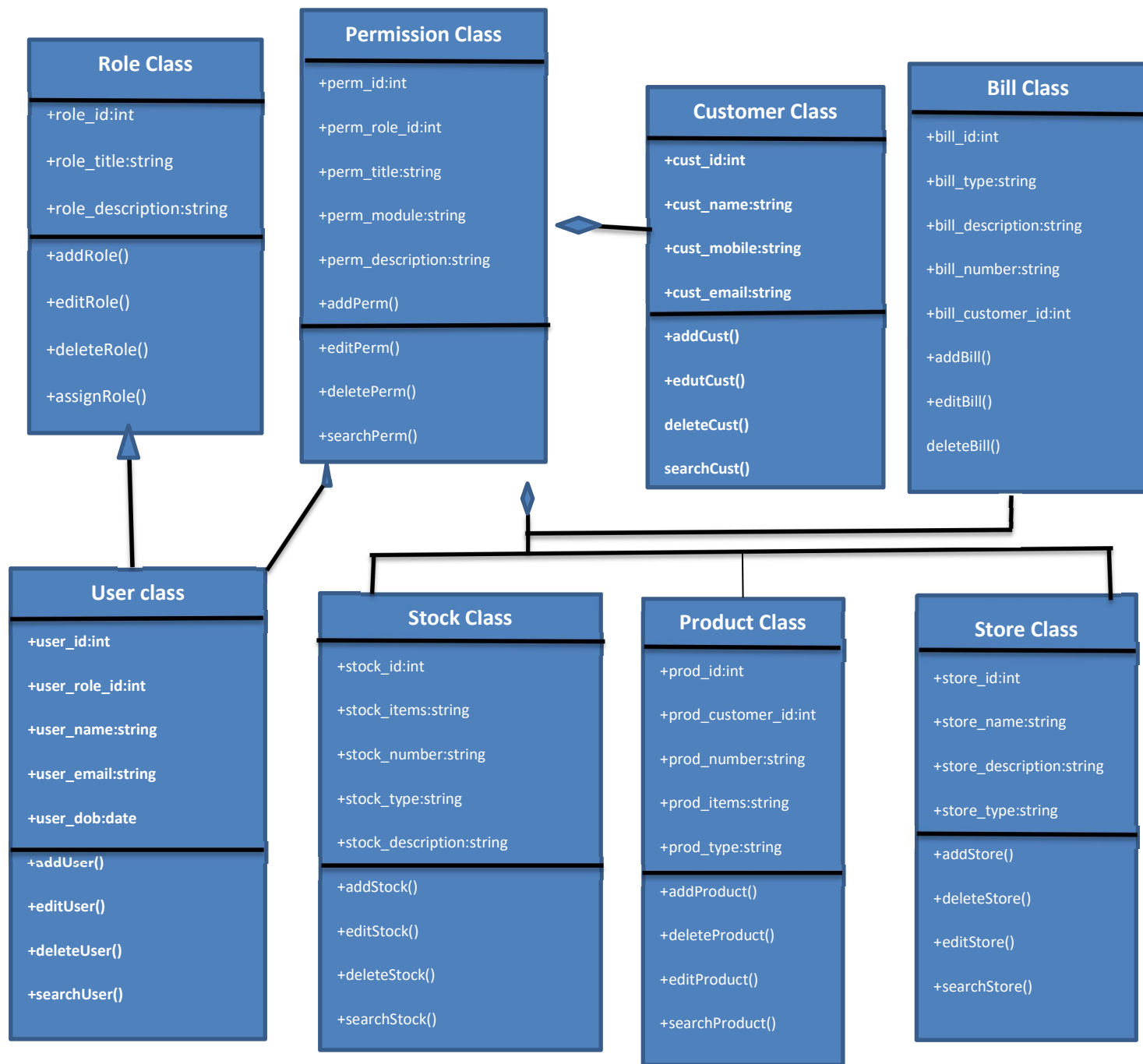
- **Stock class:** manage all the operations of stock
- **Product class:**manage all the operations of the product.
- **Product quality class:**manage all the operations of product quality.
- **Bill class:**manage all the operations of bill.
- **Customer class:**manage all the operations of customer.
- **Store class:**manage all the operations of the store.

Classes and their attributes of SMS class diagram:

- **StockAttributes:**stock-id,stock-items,stock-number,stock-type,stock-description.
- **ProductAttributes:**product id,product customer id product items,product number,product type,product description.
- **Product quality Attributes:**product quality id,quality name,quality type, quality description.
- **Bill Attributes:**bii id,bill customer id,bill number,bill type,bill receipt,bill description.
- **Customer Attributes:**customer id,name,mobile,email,username.
- **Store Attribute:**store id,name,type,description.

Classes and their methods of SMS class diagram:

- **Stock methods:** addStock(), editstock(), deletestock(), updatestock(), save stock(), searchstock().
- **Product methods:** add product(), edit product(), delete product(), update product(), save product(), search product().
- **Productquality methods:** add product quality(), edit product quality(), delete product quality(), update product quality(), save product quality(), search product quality().
- **Bill methods:** add bill(), edit bill(), delete bill(), update bill(), save bill(), search bill().
- **Customer methods:** addcustomer(), edit customer(), delete customer(), update customer(), save customer(), search customer().
- **Store methods:** add store(), edit store(), delete store(), update store(), save store(), search store().

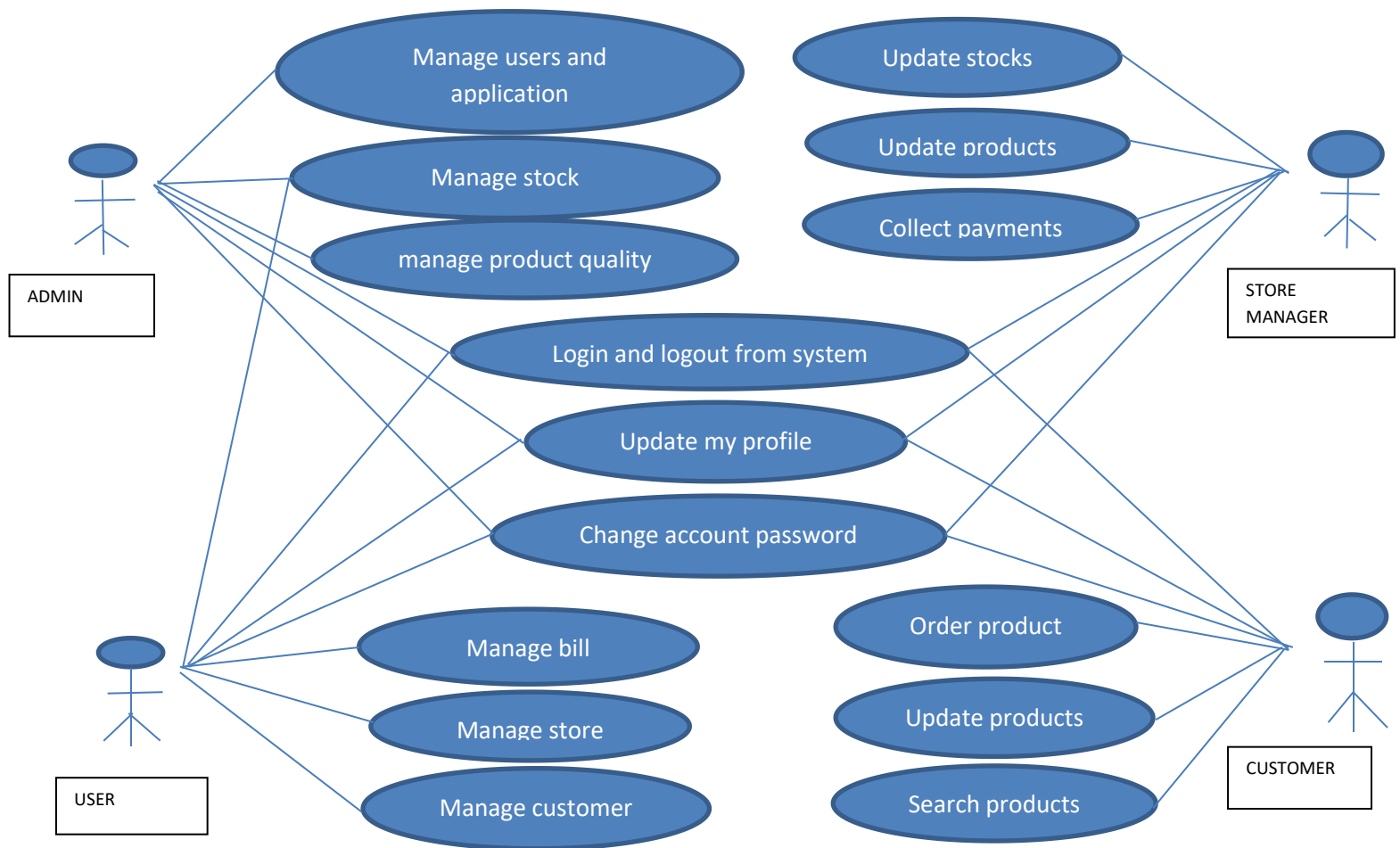


2.USECASEDIAGRAM:

The use case diagram is graphic depiction of the interactions among the elements of Store Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of Store Management System. The main actors of Store Management System in this Use Case Diagram are: Super Admin, System User, Store Manager, Customer, who perform the different type of use cases such as Stock, Manage Product, Manage Product Quality, Manage Bill, Manage Customer, Manage Store, Manage Users and Full Store Management System Operations. Major elements of the UML use case diagram of Store Management System are shown on the picture below.

The relationships between and among the actors and the use cases of Store Management System:

- **Super Admin Entity** : Use cases of Super Admin are Stock, Manage Product, Manage Product Quality, Manage Bill, Manage Customer, Manage Store, Manage Users and Full Store Management System Operations
- **System User Entity** : Use cases of System User are Stock, Manage Product, Manage Product Quality, Manage Bill, Manage Customer, Manage Store
- **Store Manager Entity** : Use cases of Store Manager are Update Products, Update Stocks, Create Invoices, Collect Payments
- **Customer Entity** : Use cases of Customer are Search Products, View Stocks, Order Products, Make Payment, Check Order History.

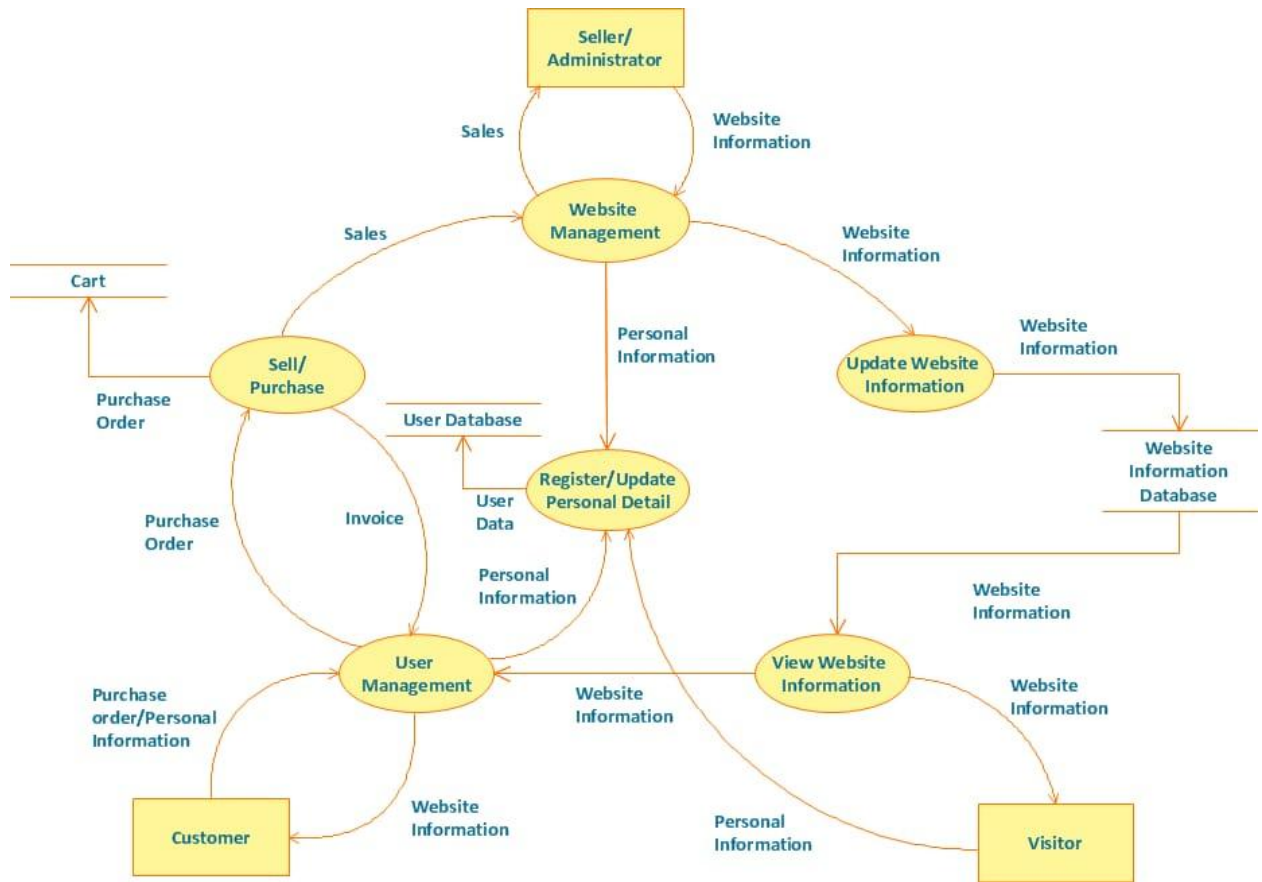


3.DataFlowDiagram:

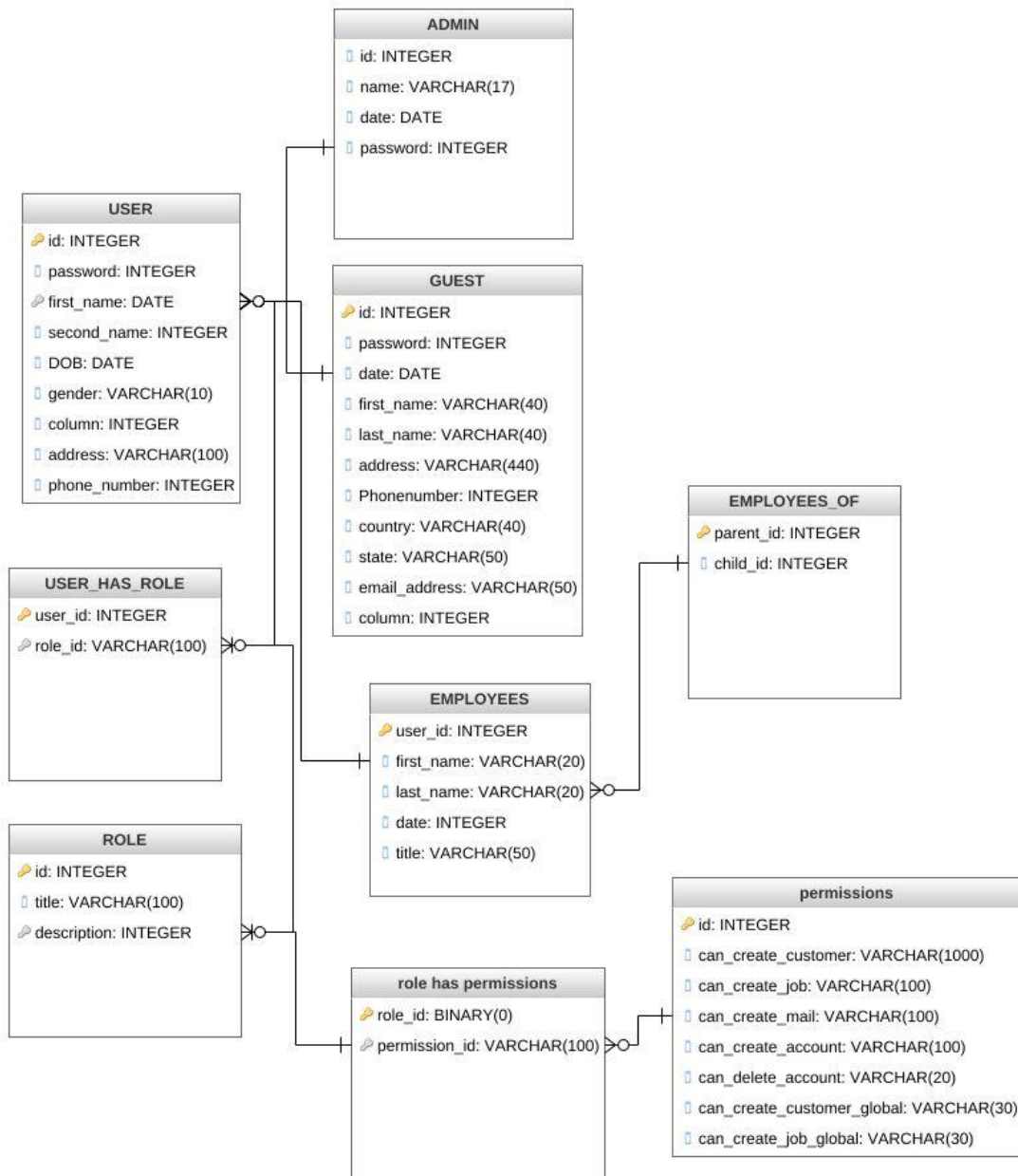
Structured-systems analysis and design method uses data flow diagrams to represent the process of data flowing through a system. Talking about this might be useless without a proper example of DFD for online store (Data Flow Diagram).

This DFD example shows all the distinctness that a diagram can bring into a scattered data structure. Data flow diagrams are used to show how data is processed within some business processes. Making DFD is a common practice for business process modeling and analysis. This diagram represents the online store business flow. It describes inputs and outputs within online selling process and depicts the interactions between its participants.

This DF diagram can be used by system analysts to create an overview of a business, to study and discover its inherent strengths and weak points.



DATABASE DIAGRAM:



Project summary

This project is about creating a sales and dashboard for a SMS application. It shows the catalogue of various things available for purchase in the store, and all the sales report.

Purpose

To show the whole store sales report at detail. Where one can know how much sales has happened till date. To make easy business solutions.

Goals

- To offer all sales reports with a high degree of accuracy. To offer all sales reports with a high degree of accuracy.
 - To figure out how many sales were made based on the number of customers that came in and placed orders.
1. List of products sold
 2. List of quantity sold against each product.
 3. List of quantity and total sales against each product
 4. List of quantity sold against each product and against each store.
 5. List of quantity sold against each Store with total turnover of the store.
 6. List of products which are not sold
 7. List of customers who have not purchased any product.

Here we will take customers, products, sales tables and use above operations to know the information.

```
CREATE TABLE IF NOT EXISTS `customers` (  
  `c_id` int(1) DEFAULT NULL,  
  `customer` varchar(4) DEFAULT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
--  
-- Dumping data for table `customers`  
--
```

```
INSERT INTO `customers` (`c_id`, `customer`) VALUES
(1, 'Rabi'),
(2, 'Raju'),
(3, 'Alex'),
(4, 'Rani'),
(5, 'King'),
(7, 'Ronn'),
(8, 'Jem'),
(9, 'Tom');
```

```
-- -----
```

```
--
```

```
-- Table structure for table `products`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `products` (
  `p_id` int(1) DEFAULT NULL,
  `product` varchar(12) DEFAULT NULL,
  `price` int(2) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
--
```

```
-- Dumping data for table `products`
```

```
--
```

```
INSERT INTO `products` (`p_id`, `product`, `price`) VALUES
(1, 'Hard Disk', 80),
(2, 'RAM', 90),
(3, 'Monitor', 75),
(4, 'CPU', 55),
(5, 'Keyboard', 20),
(6, 'Mouse', 10),
(7, 'Motherboard', 50),
(8, 'Power supply', 20);
```



```
-- -----  
  
--  
-- Table structure for table `sales`  
--  
  
CREATE TABLE IF NOT EXISTS `sales` (  
  `sale_id` int(1) DEFAULT NULL,  
  `c_id` int(1) DEFAULT NULL,  
  `p_id` int(1) DEFAULT NULL,  
  `product` varchar(7) DEFAULT NULL,  
  `qty` int(1) DEFAULT NULL,  
  `store` varchar(3) DEFAULT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
--  
-- Dumping data for table `sales`  
--
```

```
INSERT INTO `sales` (`sale_id`, `c_id`, `p_id`, `product`, `qty`, `store`) VALUES  
(1, 2, 3, 'Monitor', 2, 'ABC'),  
(2, 2, 4, 'CPU', 1, 'DEF'),  
(3, 1, 3, 'Monitor', 3, 'ABC'),  
(4, 4, 2, 'RAM', 2, 'DEF'),  
(5, 2, 3, 'Monitor', 3, 'ABC'),  
(6, 3, 3, 'Monitor', 2, 'DEF'),  
(7, 2, 2, 'RAM', 3, 'ABC'),  
(8, 3, 2, 'RAM', 2, 'DEF'),  
(9, 2, 3, 'Monitor', 2, 'ABC');
```

```
-- -----  
Now we will use some operations of sql and fetch output:
```

1.List of products sold

```
SELECT product,p_id FROM `sales` group by product
```

product	p_id
CPU	4
Monitor	3
RAM	2

2. List of quantity sold against each product.

```
SELECT product,p_id,sum(qty) FROM `sales` group by product
```

product	p_id	sum(qty)
CPU	4	1
Monitor	3	12
RAM	2	7

3. List of quantity and total sales against each product

```
SELECT a.product,a.p_id,sum(qty), sum(qty*price) FROM `sales`
```

```
a LEFT JOIN products b on a.p_id = b.p_id group by product
```

product	p_id	sum(qty)	sum(qty*price)
CPU	4	1	55
Monitor	3	12	900
RAM	2	7	630

4. List of quantity sold against each product and against each store.

```
SELECT product , store , sum(qty) FROM sales group by product, store
```

product	store	sum(qty)
CPU	DEF	1
Monitor	ABC	10
Monitor	DEF	2
RAM	ABC	3
RAM	DEF	4

5. List of quantity sold against each Store with total turnover of the store.

```
SELECT a.store, sum(qty) total_qty,  
  
sum(b.price * ( a.qty )) AS total_price FROM sales a  
  
LEFT JOIN products b ON a.p_id = b.p_id GROUP BY store
```

store	total_qty	total_price
ABC	13	1020
DEF	7	565

6. List of products which are not sold

```
SELECT a.product , a.p_id FROM products a  
  
LEFT JOIN sales b on a.p_id=b.p_id WHERE b.sale_id is null
```

product	p_id
Hard Disk	1
Keyboard	5
Mouse	6
Motherboard	7
Power supply	8

7. List of customers who have not purchased any product.

```
SELECT a.customer, a.c_id from customers a  
  
LEFT JOIN sales b on a.c_id=b.c_id WHERE b.sale_id IS NULL
```

customer	c_id
King	5
Ronn	7
Jem	8
Tom	9

Here we will calculate monthly sales using SQL

```
create table sales(order_date date,sale int);
```

```
mysql>select * from sales;
```

```
+-----+-----+  
| order_date | sale |  
+-----+-----+  
| 2020-01-01 | 10 |  
| 2020-01-02 | 12 |  
| 2020-01-03 | 15 |  
| 2020-01-04 | 11 |  
| 2020-01-05 | 13 |  
| 2020-01-06 | 9 |  
| 2020-01-07 | 21 |  
| 2020-01-08 | 10 |
```

2020-01-09	10
...	...

```
insert into sales(order_date,sale)
  values('2020-01-01',10),('2020-01-02',12),('2020-01-03',15),
  ('2020-01-04',11),('2020-01-05',13),('2020-01-06',9),
  ('2020-01-07',21),('2020-01-08',10),('2020-01-09',10),
  ('2020-01-10',2),('2020-01-11',16),('2020-01-12',12),
  ('2020-01-13',10),('2020-01-14',18),('2020-01-15',15),
  ('2020-01-16',12),('2020-01-17',10),('2020-01-18',18),
  ('2020-01-19',14),('2020-01-20',16),('2020-01-21',12),
  ('2020-01-22',21),('2020-01-23',13),('2020-01-24',15),
  ('2020-01-25',20),('2020-01-26',14),('2020-01-27',16),
  ('2020-01-28',15),('2020-01-29',10),('2020-01-30',18);
```

```
select year(order_date),month(order_date),sum(sale)
  from sales
 group by year(order_date),month(order_date)
 order by year(order_date),month(order_date);
```

year(order_date)	month(order_date)	sum(sale)
2020	1	408
2020	2	320
2020	3	540

The above query uses SUM function which will help you sum the total sales every month.

```
select year(order_date),month(order_date),count(sale)
      from sales
      group by year(order_date),month(order_date)
      order by year(order_date),month(order_date);
```

```
+-----+-----+-----+
| year(order_date) | month(order_date) | sum(sale) |
+-----+-----+-----+
|          2020    |          1        |      18   |
|          2020    |          2        |      10   |
|          2020    |          3        |      21   |
|          ...     |          ...       |      ...   |
```

```
create table sales(product varchar(255),order_date date,sale int);
```

```
mysql> insert into sales values('A','2020-01-01',20),('B','2020-01-02',25),
('B','2020-01-03',15),('A','2020-01-04',30),('A','2020-01-05',20);
```

```
mysql> select * from sales;
```

```
+-----+-----+-----+
| product | order_date | sale |
```

```

+-----+-----+-----+
|  A    | 2020-01-01 |   20 |
|  B    | 2020-01-02 |   25 |
|  B    | 2020-01-03 |   15 |
|  A    | 2020-01-04 |   30 |
|  A    | 2020-01-05 |   20 |
+-----+-----+-----+

```

```

mysql> select product, year(order_date),month(order_date),sum(sale)
        from sales
        group by product, year(order

```

```

select * from sales;

```

```

+-----+-----+-----+
| product | order_date | sale |
+-----+-----+-----+
|  A    | 2020-01-01 |   20 |
|  B    | 2020-01-02 |   25 |
|  B    | 2020-01-03 |   15 |
|  A    | 2020-01-04 |   30 |
|  A    | 2020-01-05 |   20 |
+-----+-----+-----+

```

```

select product, year(order_date),month(order_date),sum(sale)
        from sales
        group by product, year(order_date),month(order_date)
        order by product, year(order_date),month(order_date);

```

```

+-----+-----+-----+-----+
| product | year(order_date) | month(order_date) | sum(sale) |

```


+-----+-----+-----+-----+				
A		2020		1 70
B		2020		1 40
...	

You can calculate percentage growth month by month using the following SQL.

```
create table monthly_sales(month int,sale int);
mysql> insert into monthly_sales(month,sale) values(1,20),
(2,30),(3,25),(4,45),(5,25);
mysql> select * from monthly_sales;
```

+-----+-----+		
month	sale	
+-----+-----+		
1	20	
2	30	
3	25	
4	45	
5	25	
+-----+-----+		

```
select month, sale,
       if(@last_entry = 0, 0, round(((sale - @last_entry) / @last_en
try) * 100,2)) "growth rate",
       @last_entry := sale
from
(select @last_entry := 0) x,
```

```
(select month, sum(sale) sale
from   monthly_sales
group by month) y;
```

month	sale	growth rate	@last_entry := sale
1	20	0	20
2	30	50.00	30
3	25	-16.67	25
4	45	80.00	45
5	25	-44.44	25

SQL Query to Compare Product Sales By Month

```
create table sales(product varchar(255),order_date date,sale int);
```

```
mysql> insert into sales values('Pen','2020-01-01',20),('Paper','
2020-01-02',25),
('Paper','2020-01-03',15),('Pen','2020-01-04',30),('Paper','2020-
01-05',20)
...;
```

```
mysql> select * from sales;
```

product	order_date	sale
Pen	2020-01-01	20

Paper	2020-01-02	25	
Paper	2020-01-03	15	
Pen	2020-01-04	30	

```

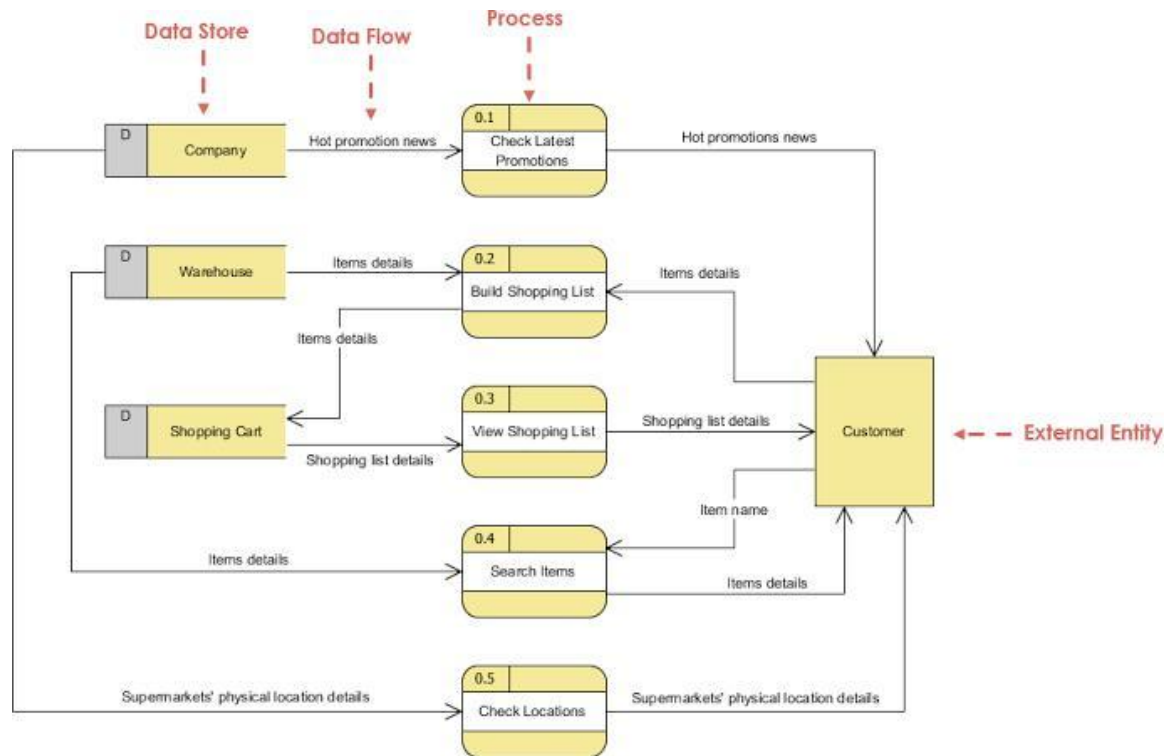
SELECT date_format(order_date, '%b-%y') as order_date,
       sum(IF(product='Pen', sale, NULL)) AS Pen,
       sum(IF(product='Paper', sale, NULL)) AS Paper
FROM sales
      GROUP BY year(order_date),month(order_date),date_format(or
der_date, '%b-%y')
      ;

```

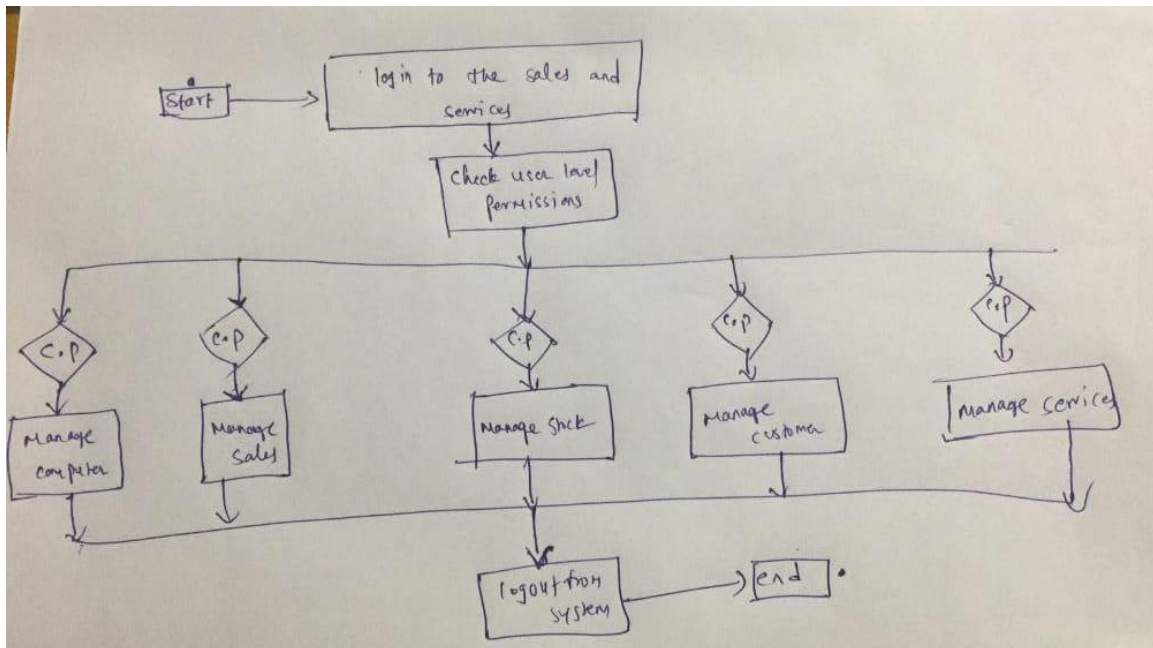
+-----+-----+-----+
order_date Pen Paper
+-----+-----+-----+
Jan-20 200 320
Feb-20 250 220
Mar-20 230 290
Apr-20 190 210
May-20 210 230
Jun-20 320 120
Jul-20 330 220
Aug-20 210 260
Sep-20 120 220
Oct-20 280 120
Nov-20 290 280
Dec-20 200 320

SYSTEM DESIGN:

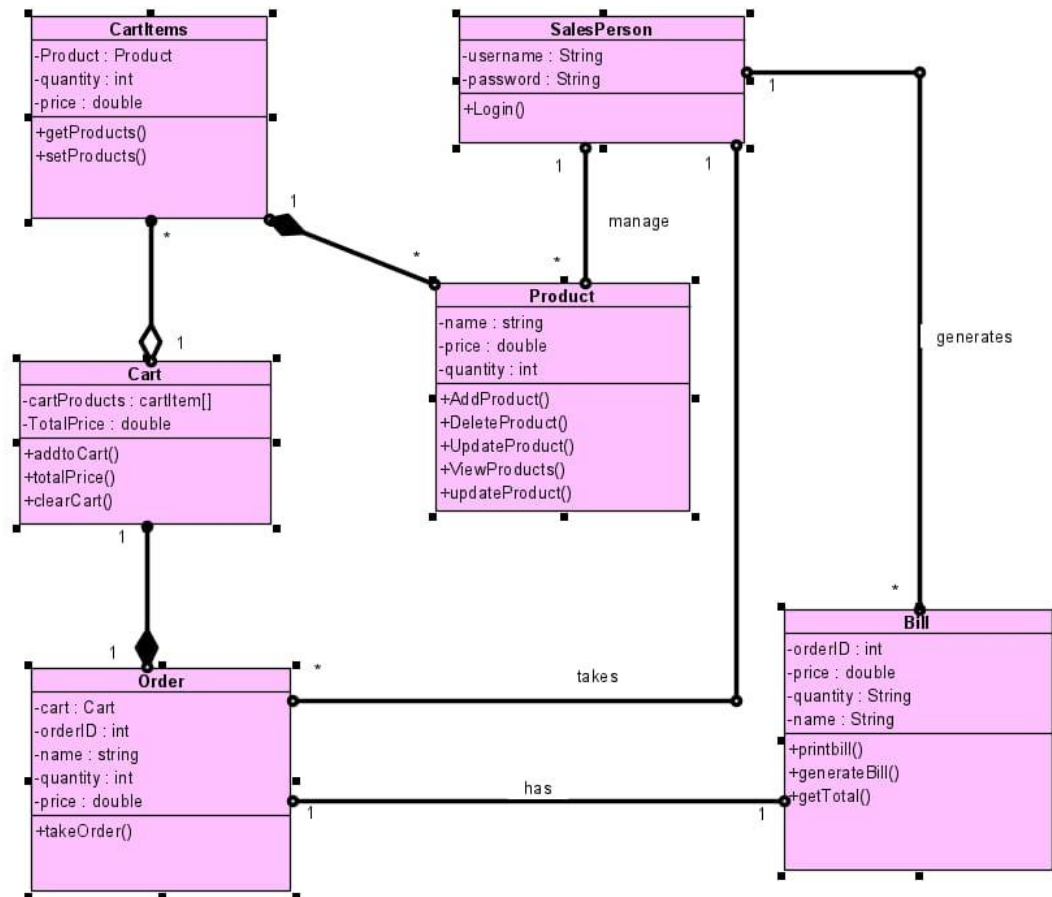
DATA FLOW DIAGRAM:



USE CASE DIAGRAM:



CLASS DIAGRAM:



MOCKUP SCREENS:

Sign up

First Name

Last Name

Email Address

TestFlight will use this address to authenticate and notify you of new builds.

Confirm Email

Please confirm your email address.

Password

Confirm

I am a developer

☐

If you would like to upload your own builds and invite your own testers

Sign up →



Tops



Type

- | | |
|---|--|
| <input type="checkbox"/> T-shirts | <input type="checkbox"/> Sequin tops |
| <input type="checkbox"/> Tank tops | <input type="checkbox"/> Off shoulder tops |
| <input type="checkbox"/> Crop tops | <input type="checkbox"/> Evening tops |
| <input type="checkbox"/> Shirts & blouses | <input type="checkbox"/> Camis |
| <input type="checkbox"/> Bodysuits | |

Sort by

Latest



Price



29

799

