ACADEMIC YEAR 2022-2023
LEBANESE UNIVERSITY
Faculty of Science
Department of Math & CS
Course: I3306-Database II
Semester:1

Shop Management System

Course Instructor

Dr. Mohammad Dbouk

Prepared By

Lynn Aouad – 97751

Rana Jaafar – 97456

Lynn El-mesmar - 97068

Acknowledgments

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project, people who gave their unending support right from the stage the project idea was conceived.

"To be successful, the first thing to do is to fall in love with your work"

The project was done with full passion in order to be able to grab the needed attention.

A special gratitude we give to **Dr. Mohammad Dbouk**, for the guidance, inspiration and contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report, we are very thankful.

Thank you for your dedication and hard work.

- **Rana Jaafar**
- **Lynn Aouad**
- **Lynn El-mesmar**

Introduction

The internet has revolutionized the way we shop. Because of the numerous advantages and benefits, more and more people these days prefer buying things online over the conventional method of going into stores.

Customers can purchase items from the comfort of their own homes or workplace. Shopping is made easier and convenient for the customer through the internet. It is also easy to cancel the transactions.

Why shop Online:

- Saves time and efforts.
- The convenience of shopping at home.
- Wide variety/range of products are available.
- Good discounts / lower prices.
- Get detailed information about the product.
- We can compare various models/brands.
- Generally, in physical stores, the sales representatives try to influence the buyers to buy the product. While in online shopping, you're free to do as you will.
- Online consumers can track the order status and delivery status tracking of shipping is also available.
- Customers do not have to stand in queues in cash counters to pay for the products that have been purchased by them. They can shop from their home or workplace and do not have to spend time travelling.

Chapter I: Application Architecture

In this project we have developed two Web applications:

- User Interface
- Administrator Interface

We have developed this project using the below technology

- HTML: Page layout has been designed in HTML.
- CSS: used for all the designing part with other frameworks.
- JavaScript: All validation task and animations has been developed by JavaScript.
- PHP: All the business logic has been implemented with PHP.
- Microsoft SQL Server: has been used as database for the project.
- XAMP Server: For executing the PHP scripts and opening the websites.

User Interface:

We have developed a user interface for the shop. The user will need first to access the website, all the products will be displayed in categories, then he will add to cart all the products that he wants to buy, he can remove from cart also. When the user wants to complete the purchase, he will check out, all the calculations are made including if there is a discount or not, then the total amount is displayed on the screen, if the user wants to continue he will need to insert all his billing details, then place the order.
The order and costumer details will be updated into the database and the administrator system.

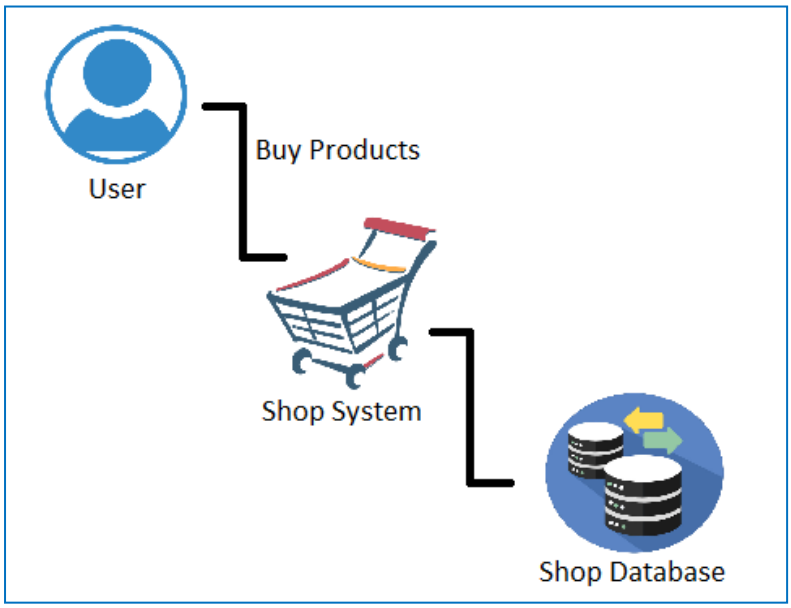
Admin Interface:

We have developed a Shop Management System. The basic concept to develop this system was to manage shop products, orders, sales and manage all the functionalities related to the shop…
The admin first will need to login then he will be able to:

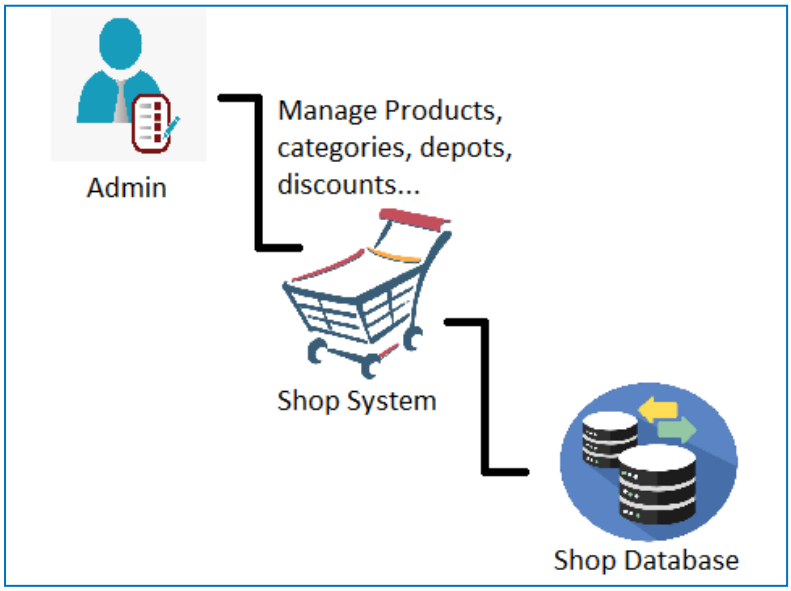
- Manage products sales
- Manage products stocks
- Manage products categories
- Manage depots
- Manage products discounts
- Manage suppliers and their supplies
- View all orders information
- View customers

All those changes will be updated directly into the database.

Architecture Part 1



Architecture Part 2



Chapter II: Database Modeling

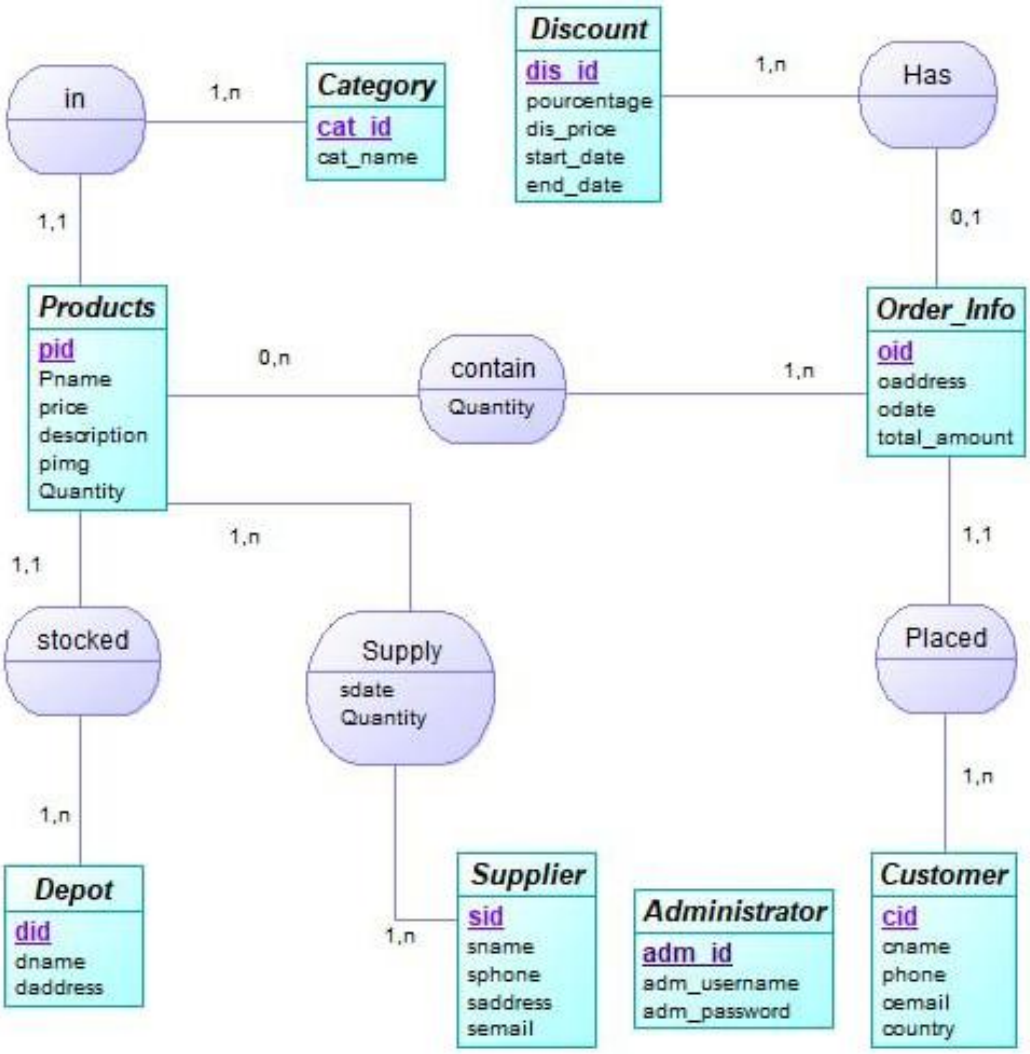
For our project we used power Designer to design the ER and physical diagram that helped us with the creation of database and the generation of the tables and indexes.

ER Diagram:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

In the ER diagram we have 8 entities, each entity has its own relations:

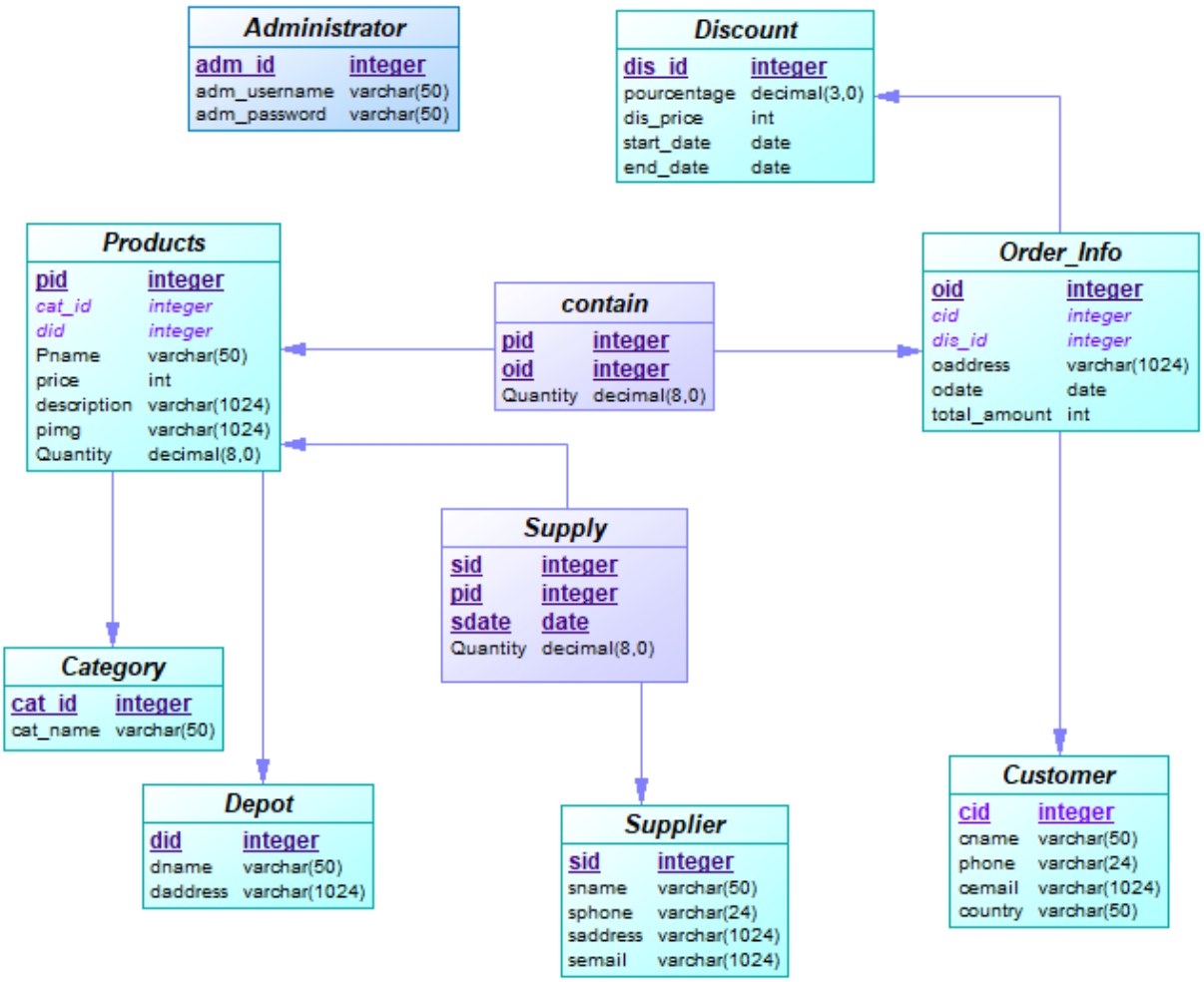
- **Products**: It contains the pid as primary key and has one to one relation with Category entity and Depot entity, and has one to many relation with Supplier entity, and has zero to many relation with Order_Info entity. Furthermore, a product can be in only one specific Category and stocked in one specific Depot, in addition, a product can be supplied by at least one or many suppliers, and can be contained in zero or many orders .
- **Category**: It contains the cat_id as primary key and has one to many relation with Products entity. Furthermore, a category can contain at least one or many products.
- **Depot**: It contains the did as primary key and has one to many relation with Products entity. Furthermore, a depot can stock at least one or many products.
- **Supplier**: It contains the sid as primary key and has one to many relation with Products entity. Furthermore, a supplier can supply at least one or many products.
- **Order_Info**: It contains the oid as primary key and has one to one relation with Customer entity and has one to many relation with Products entity, and has zero to one relation with Discount entity. Furthermore, an order can be ordered only by one specific Customer, in addition, an order can contain at least one or many products, and can have a discount or not .
- **Discount**: It contains the dis_id as primary key and has one to many relation with Order_Info entity. Furthermore, a discount can be on at least one or many orders.
- **Customer**: It contains the cid as primary key and has one to many relation with Order_Info entity. Furthermore, a customer can order at least one or many orders.
- **Administrator**: It contains the adm_id as primary key, and all the login information about an admin.


Discount
dis_id
pourcentage
dis_price
start_date
end_date
1,n
Has
0,1
in
1,n
Category
cat_id
cat_name
1,1
Products
pid
Pname
price
description
pimg
Quantity
0,n
contain
Quantity
1,n
Order_Info
oid
oaddress
odate
total_amount
1,n
1,1
1,1
stocked
Supply
sdate
Quantity
Placed
1,n
1,n
Depot
did
dname
daddress
1,n
Supplier
sid
sname
sphone
saddress
semail
Administrator
adm_id
adm_username
adm_password
Customer
cid
cname
phone
cemail
country

PDM (Physical-Data-Model):

The previous ER diagram was generated to produce a physical diagram that is shown below:

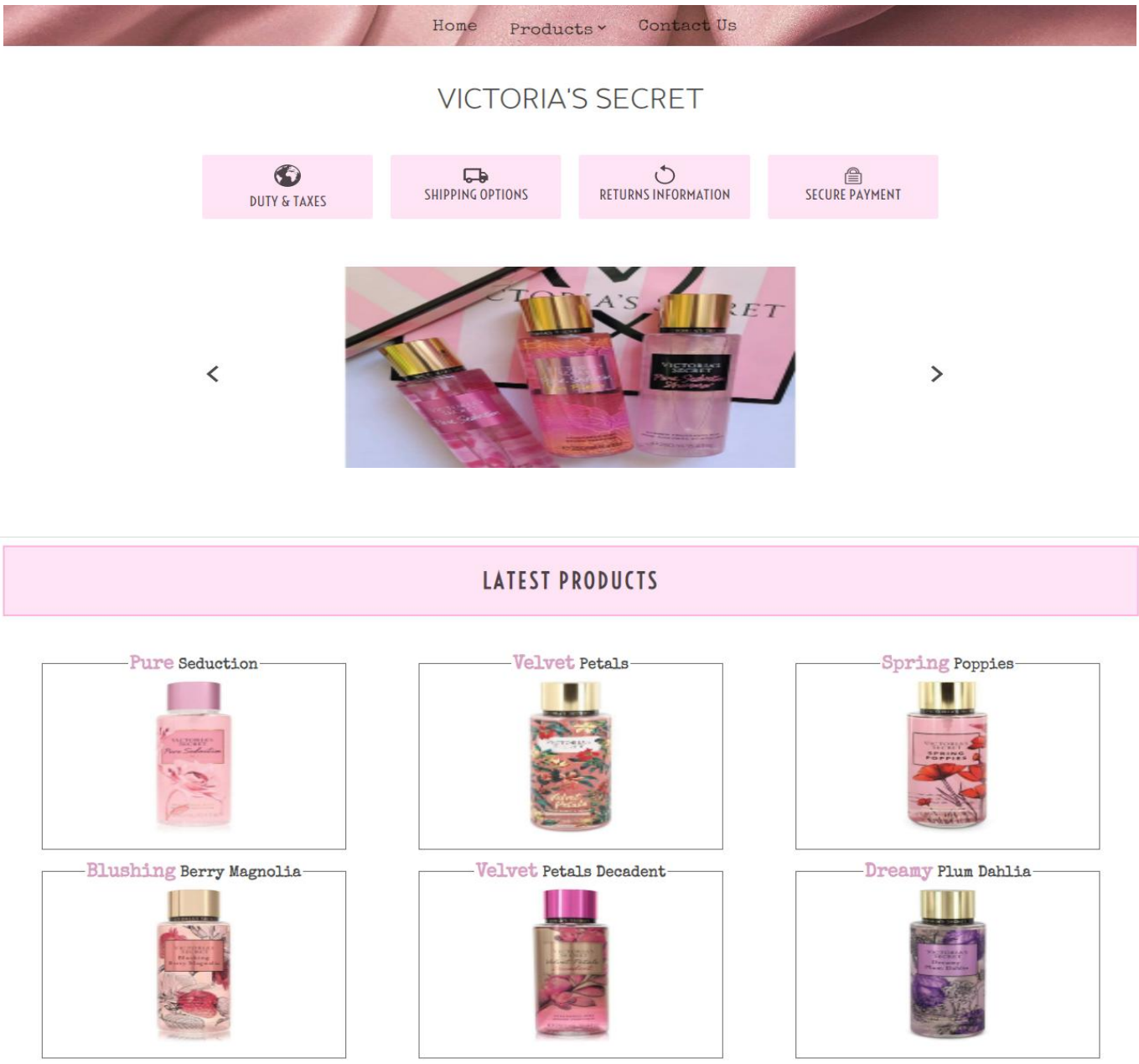
- Category (**cat_id**, cat_name)
- Depot (**did**, dname, daddress)
- Products (**pid**, **#cat_id**, **#did**, pname, price, description, pimg, quantity)
- Customer (**cid**, cname, phone, cemail, country)
- Supplier (**sid**, sname, sphone, saddress, semail)
- Supply (**#sid**, **#pid**, **sdate**, quantity)
- Discount (**dis_id**, pourcentage, dis_price, start_date, end_date)
- Order_Info (**oid**, **#cid**, **#dis_id**, oaddress, odate, total_amount)
- Contain (**#pid**, **#oid**, quantity)
- Administrator(**adm_id**, adm_username, adm_password)

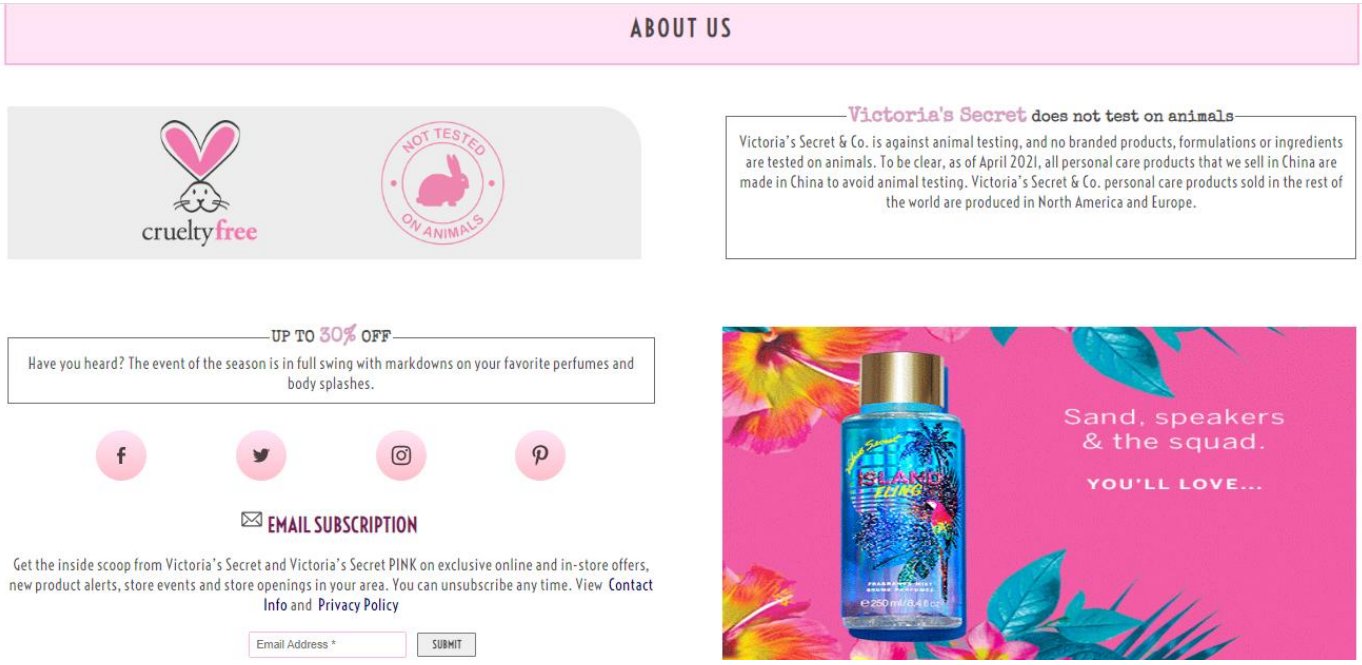


Chapter III: Services and Interface Modeling

User Interface:

1) Main Page



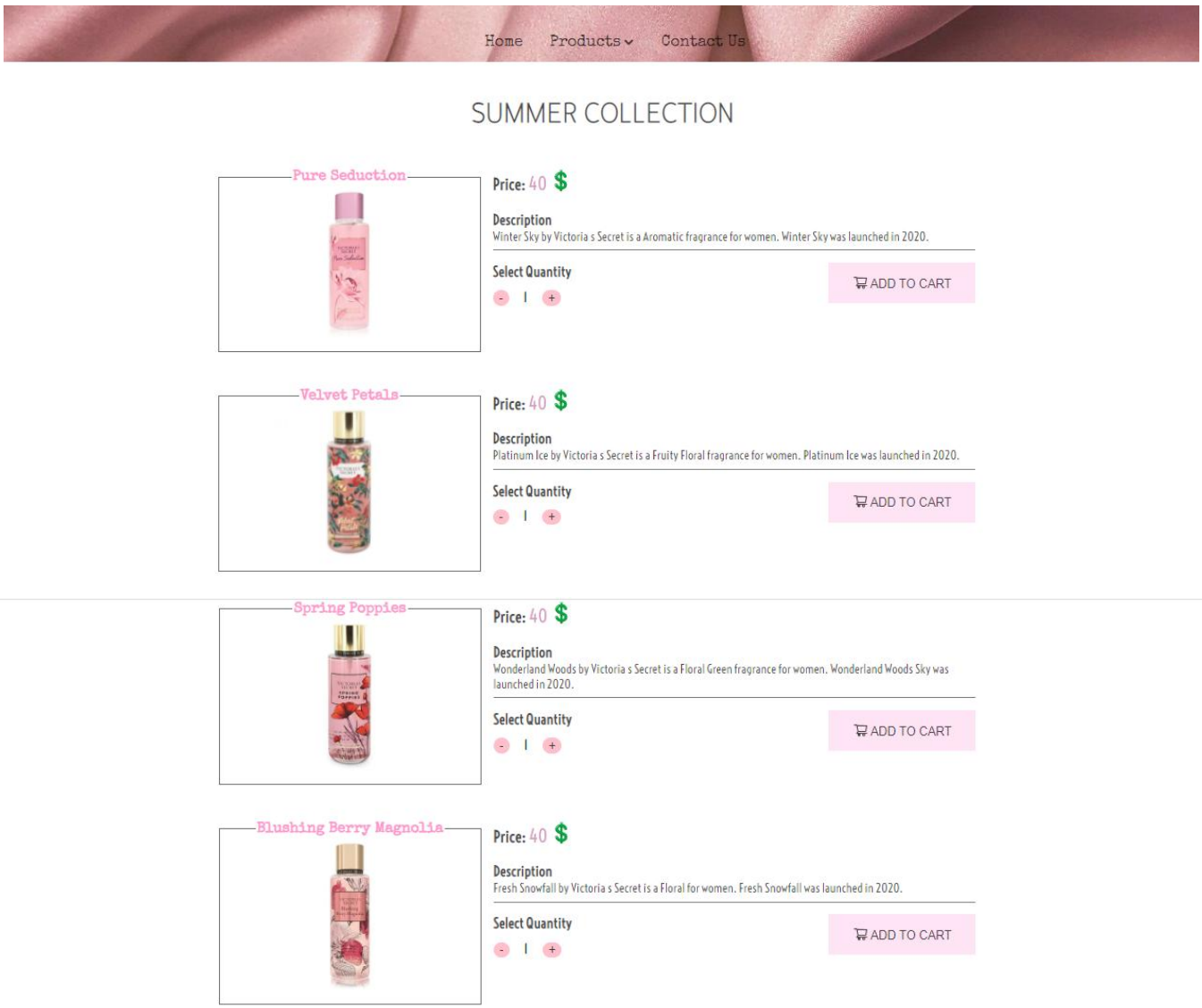


Cart is Empty

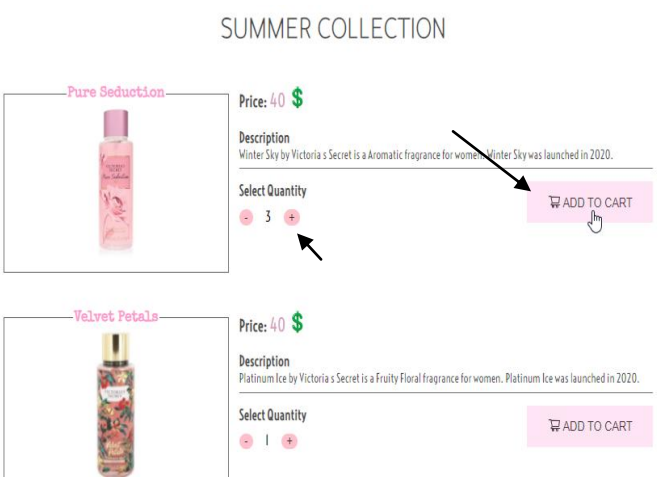
2) User choose a product category

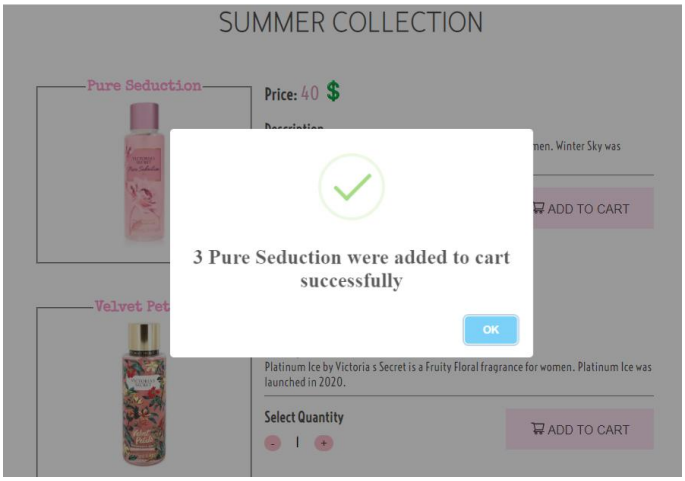
VICTORIA'S SECRET

DUTY & TAXES | SHIPPING OPTIONS | RETURNS INFORMATION | SECURE PAYMENT

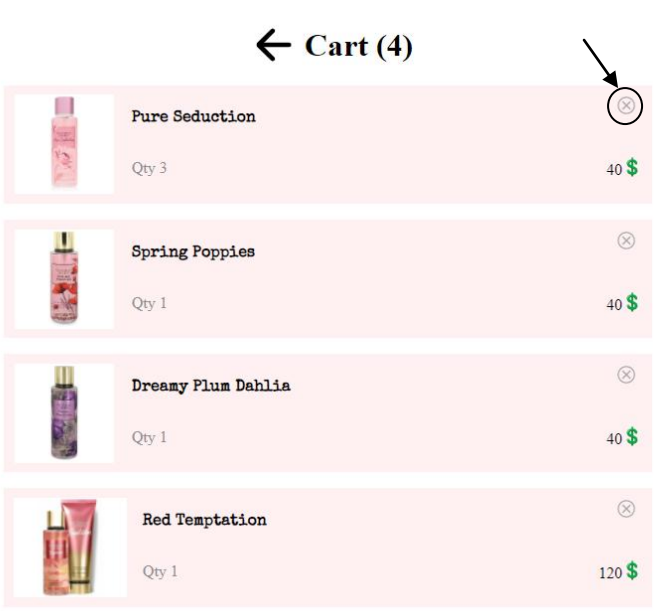


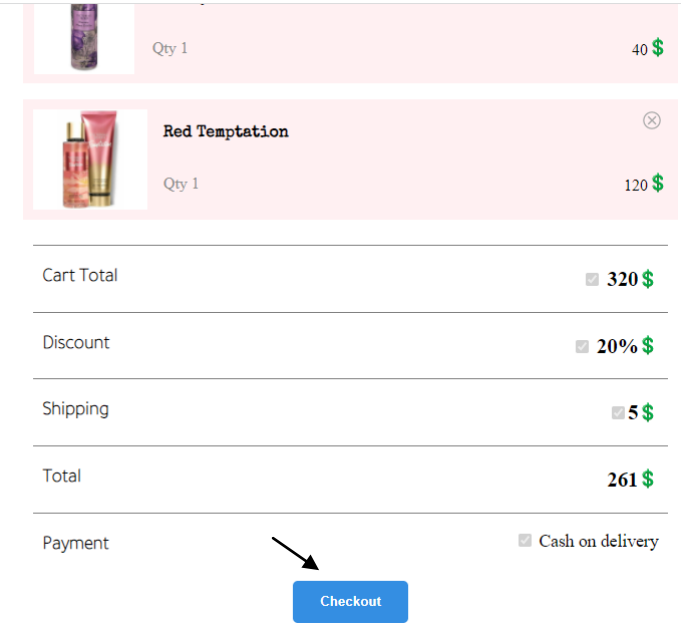
3) Add products into Cart



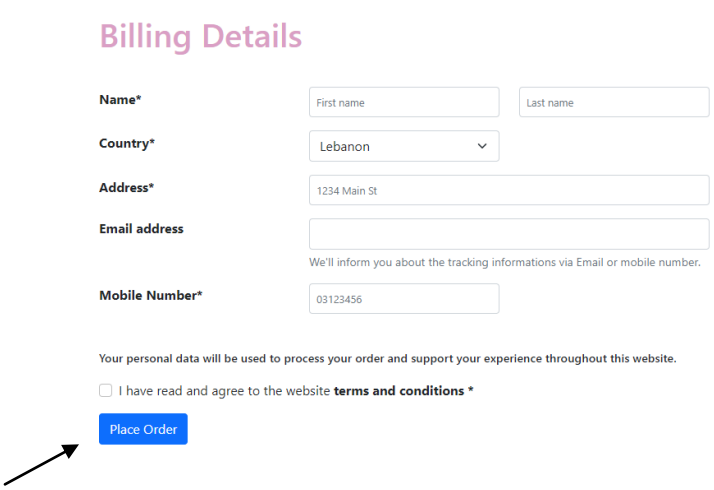


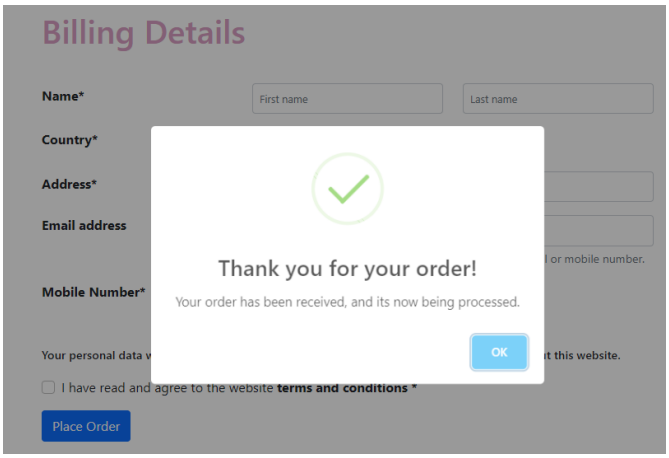
4) View Cart – User can remove product from cart





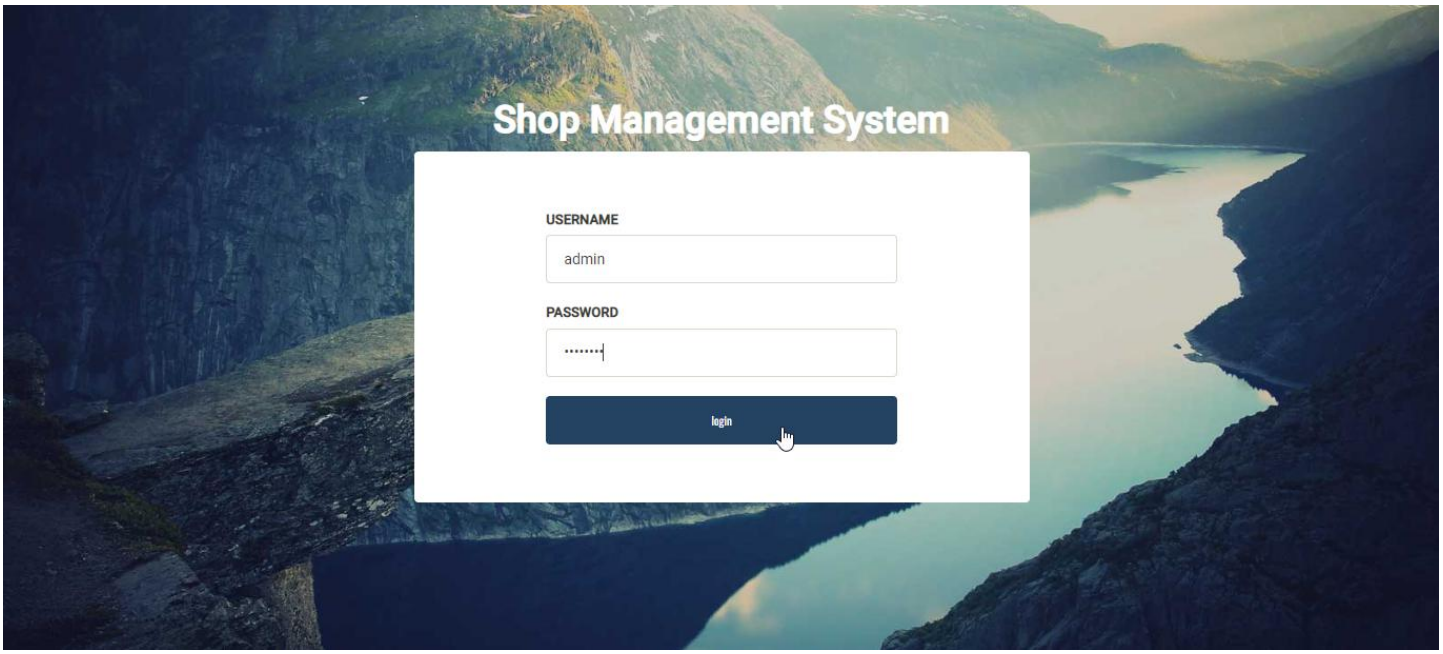
5) Checkout & Place Order



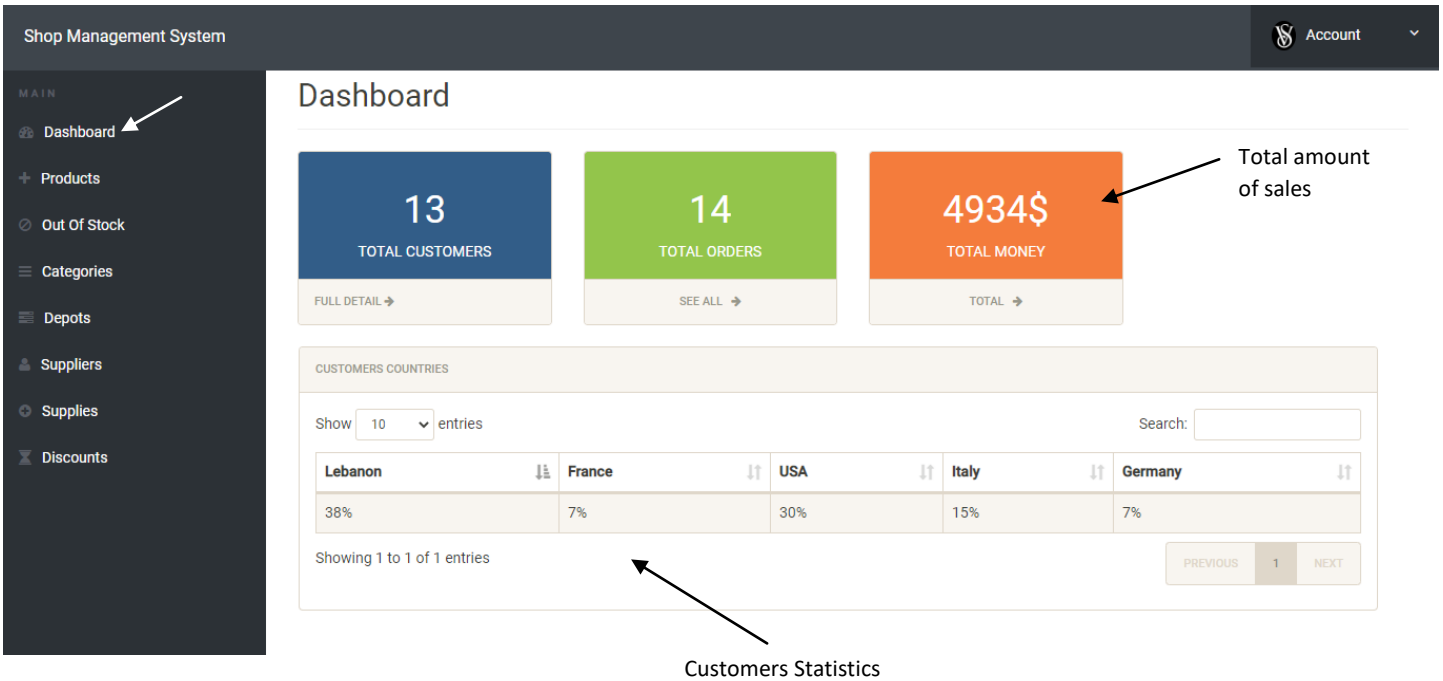


Administrator Interface:

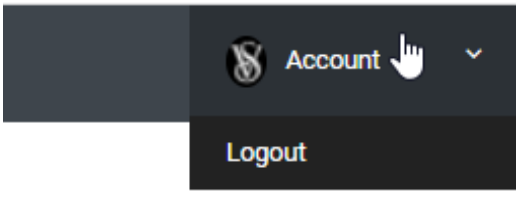
1) Login



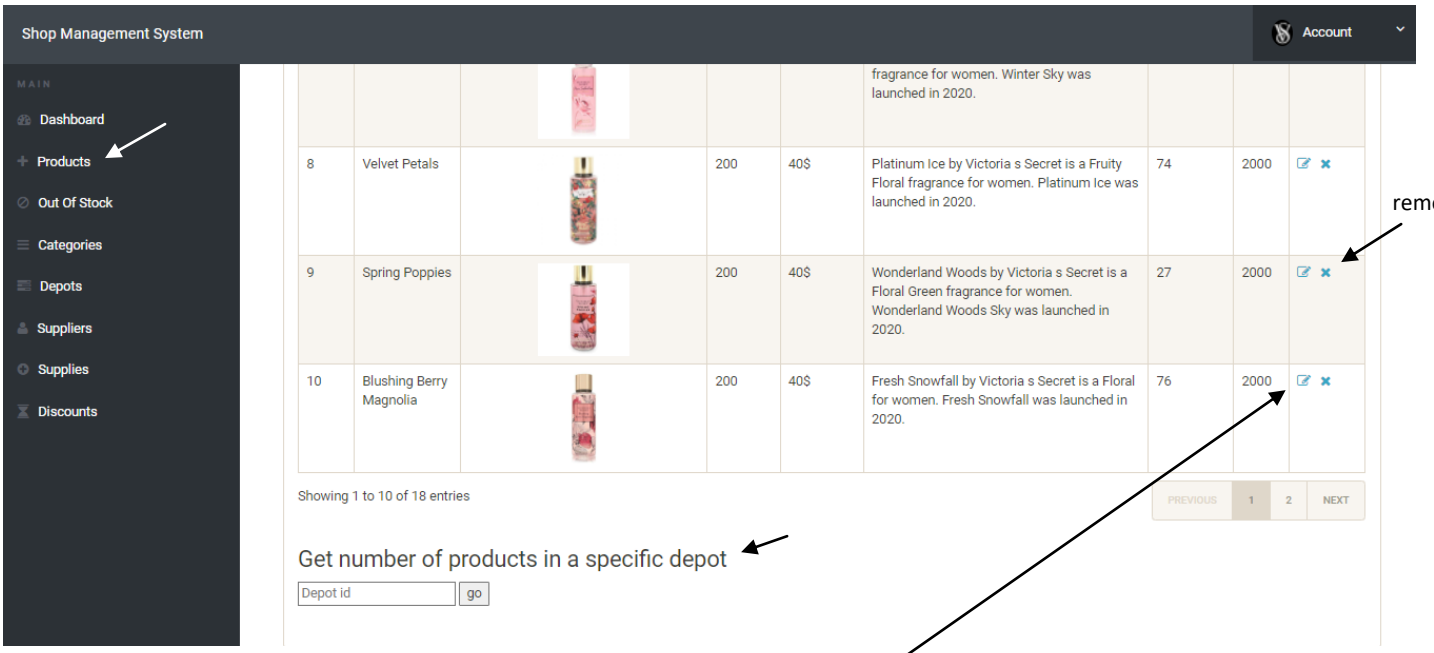
2) Dashboard – All sections admin can add, remove update…



Logout

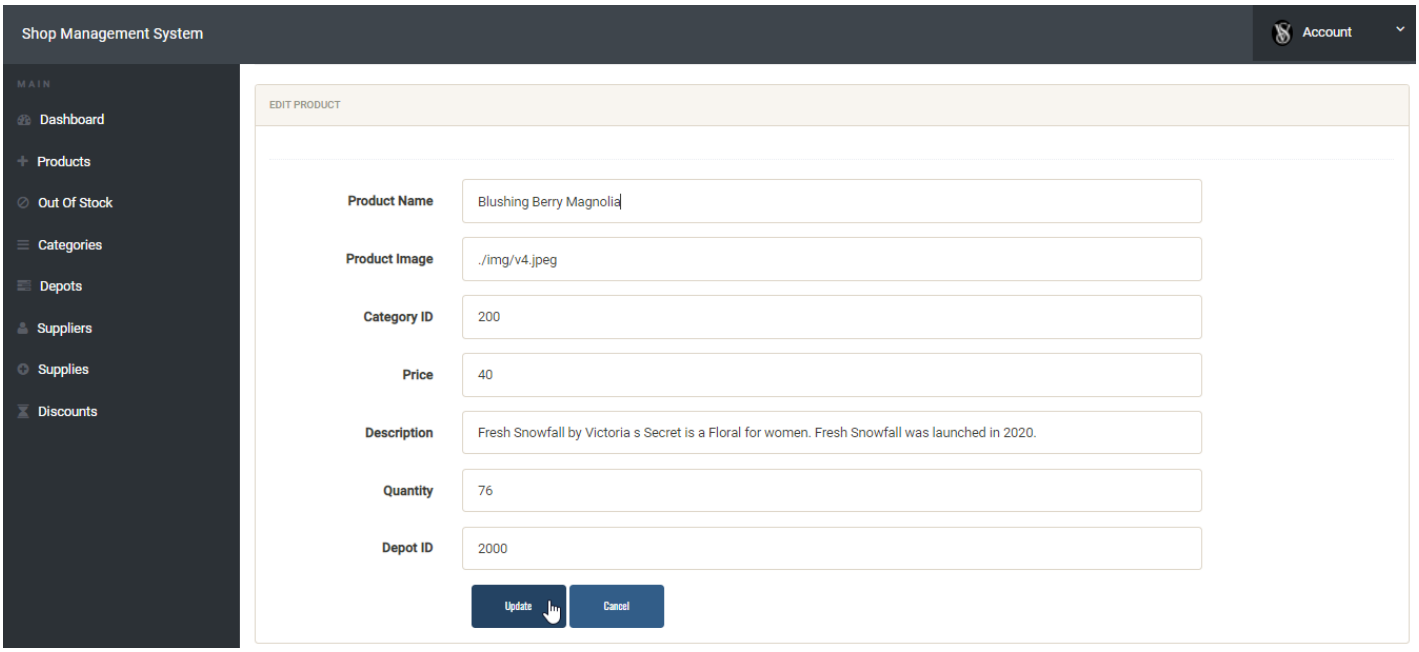


3) Products Section

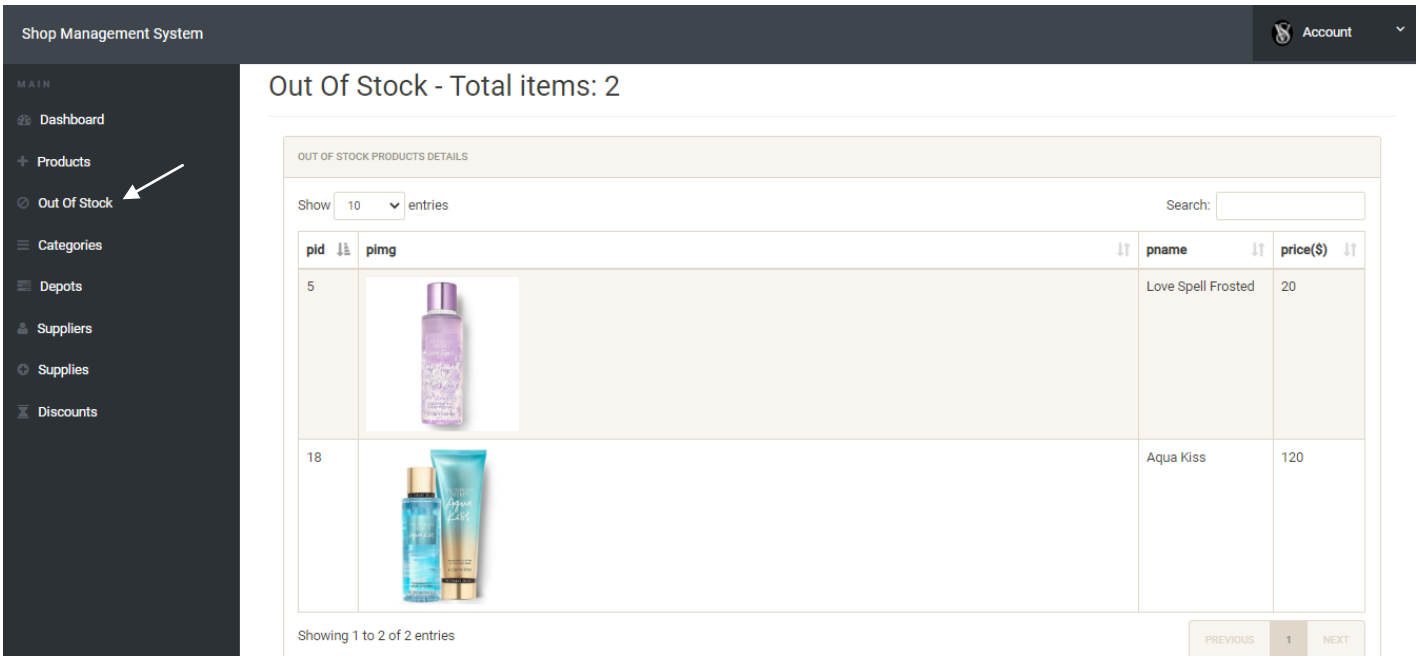


remove

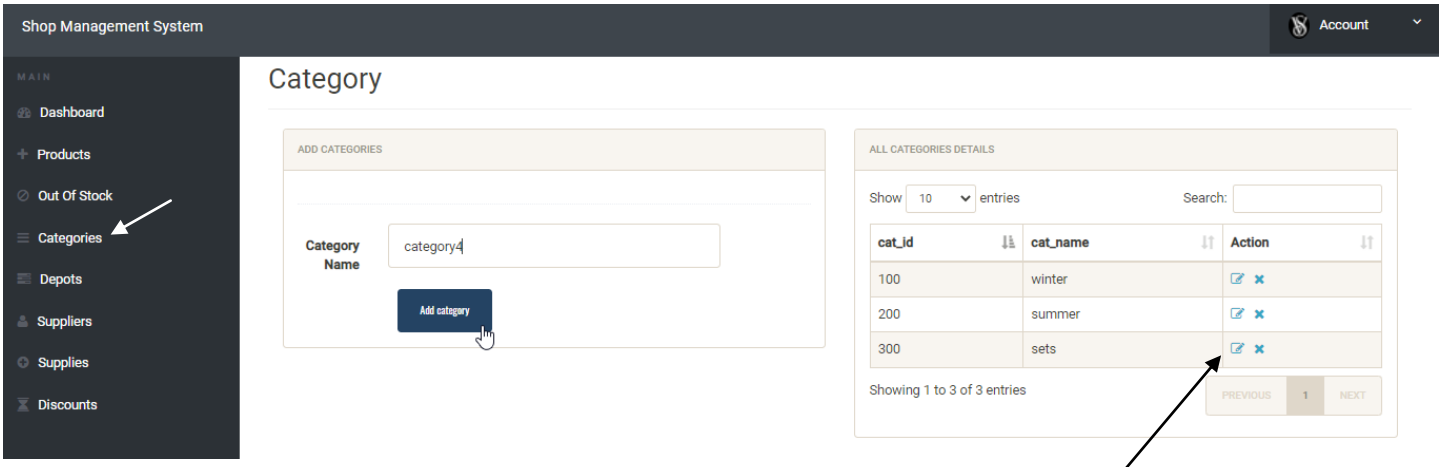
Update Product

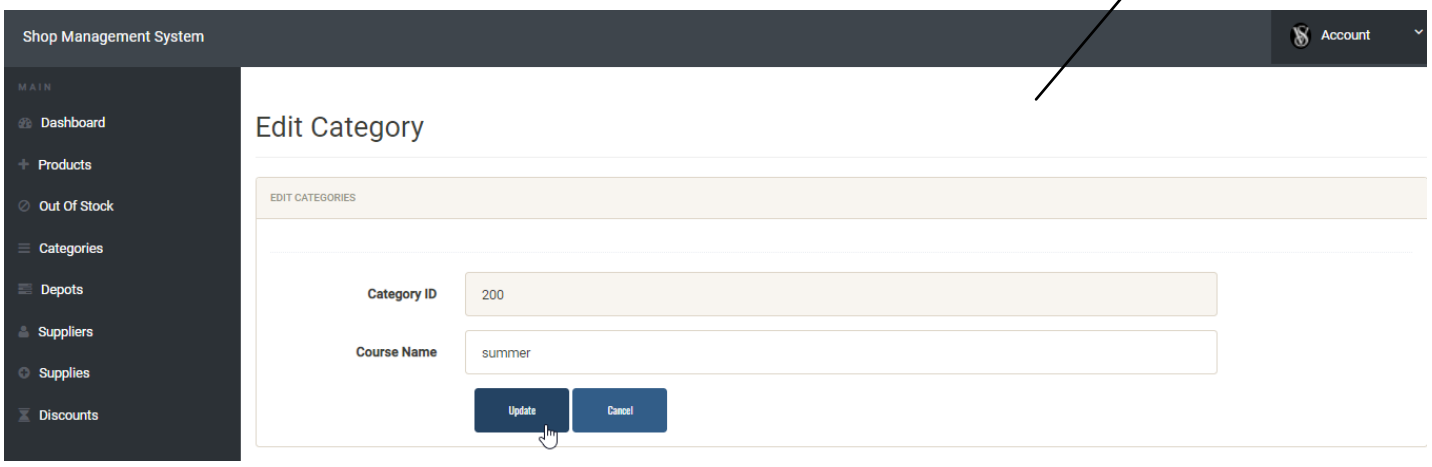


4) Out-Of-Stock

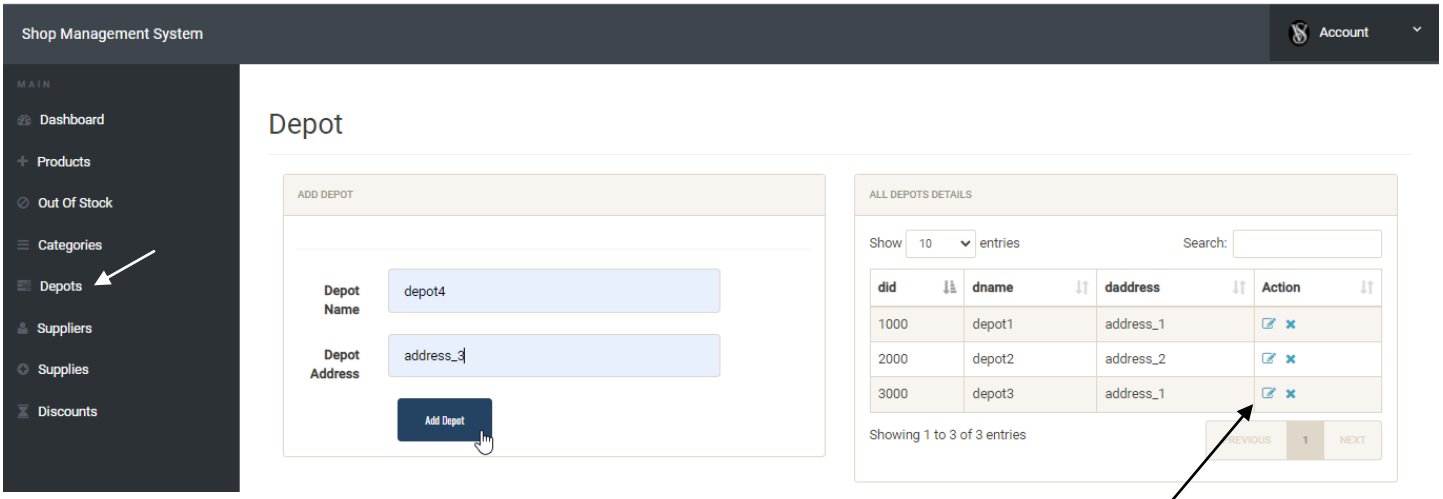


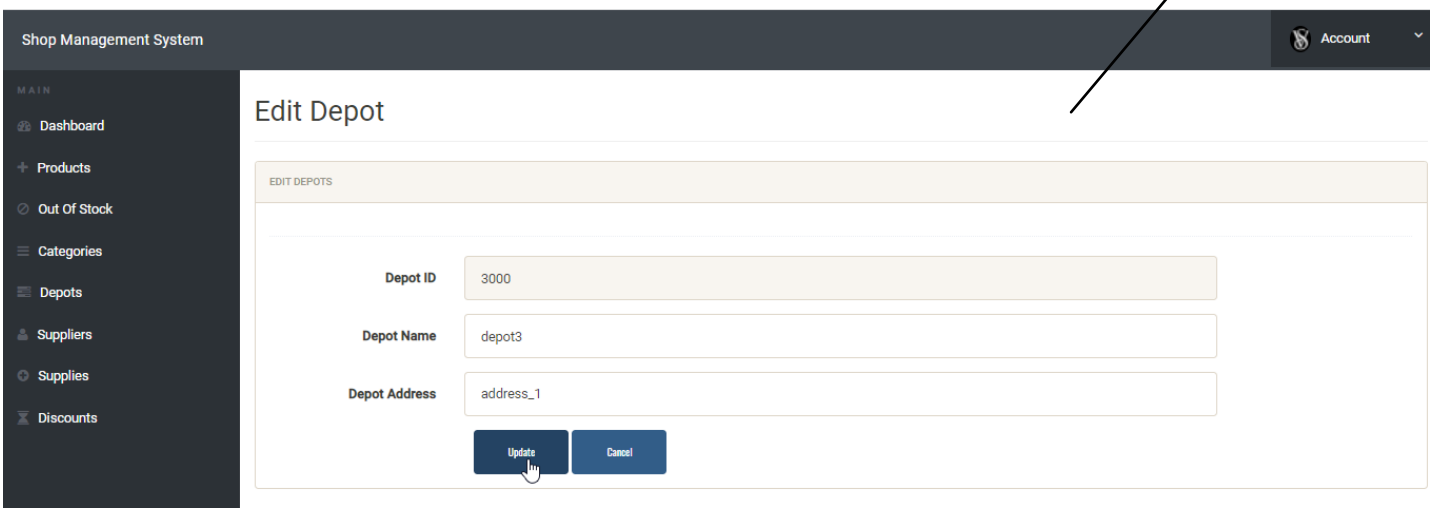
5) Category Section



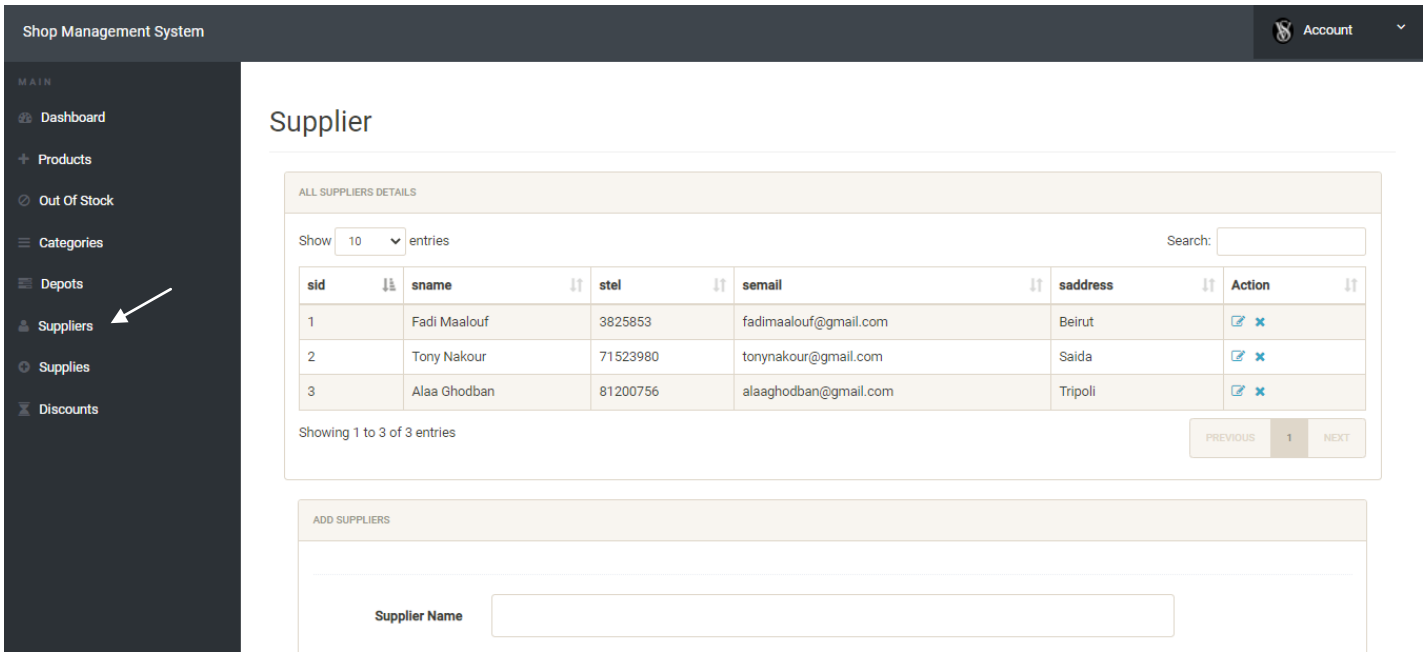


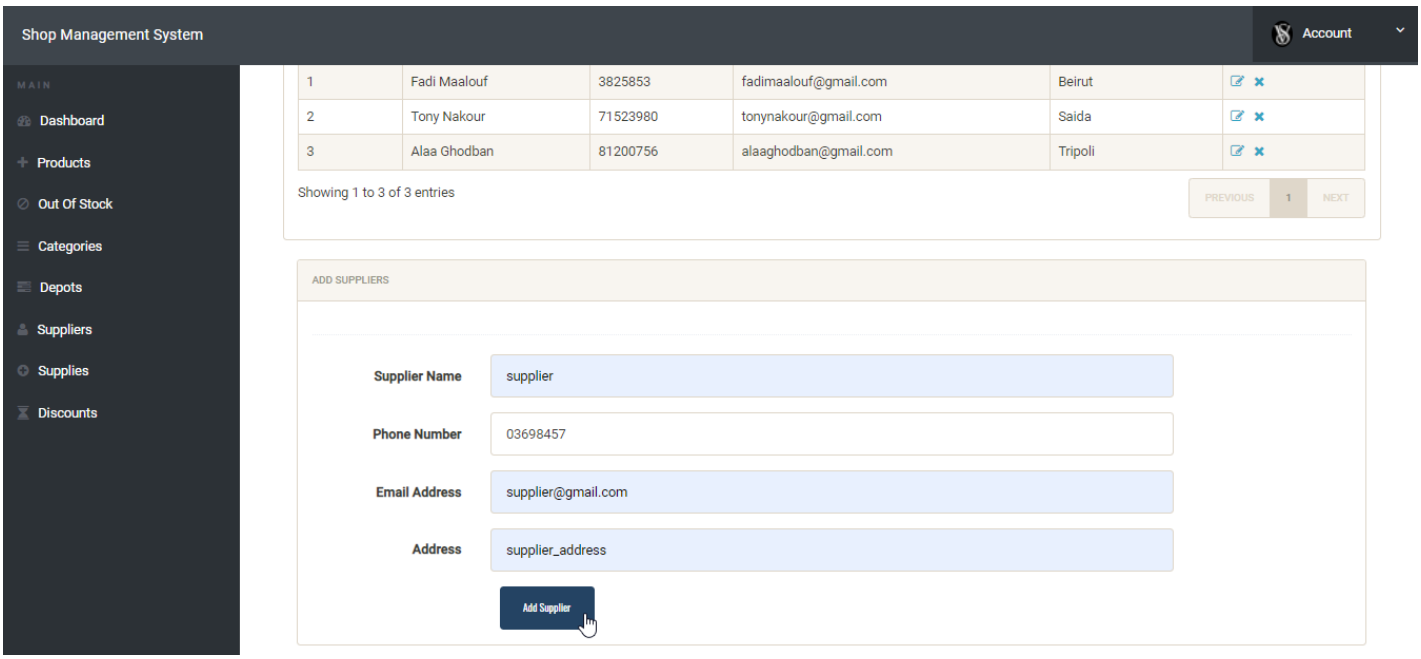
6) Depots Section

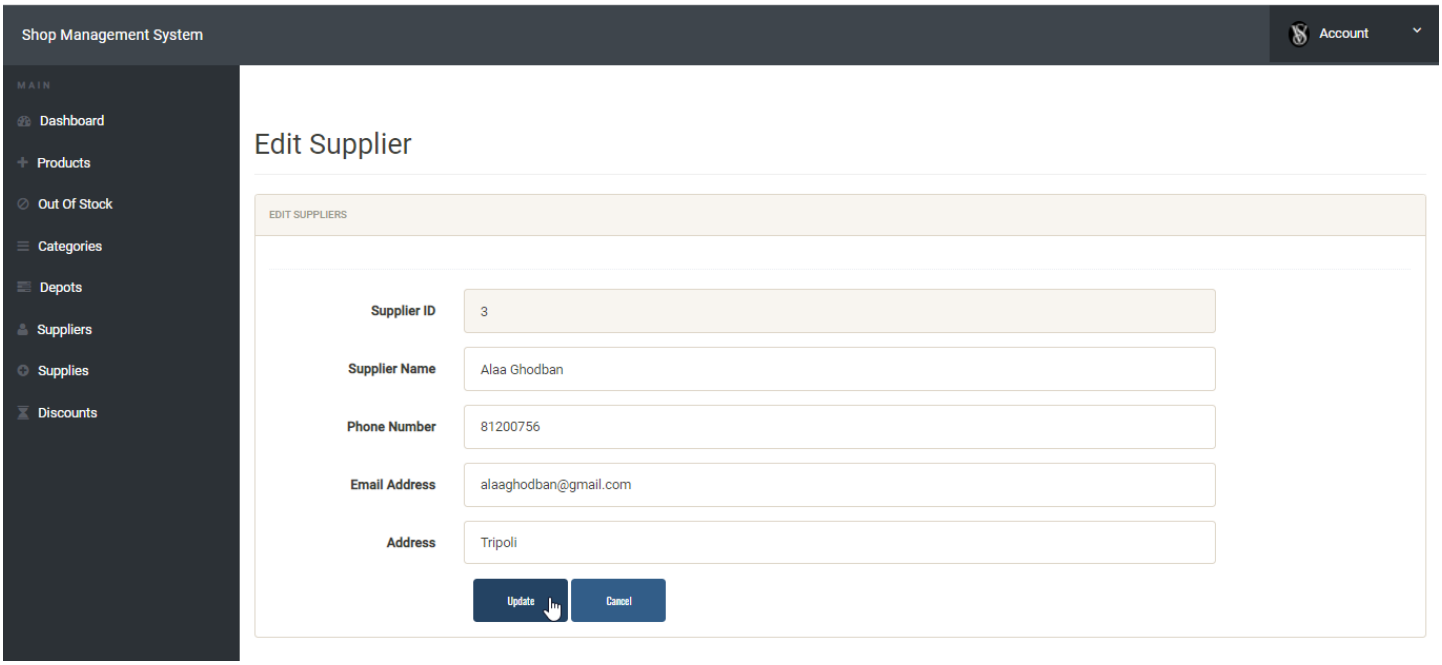




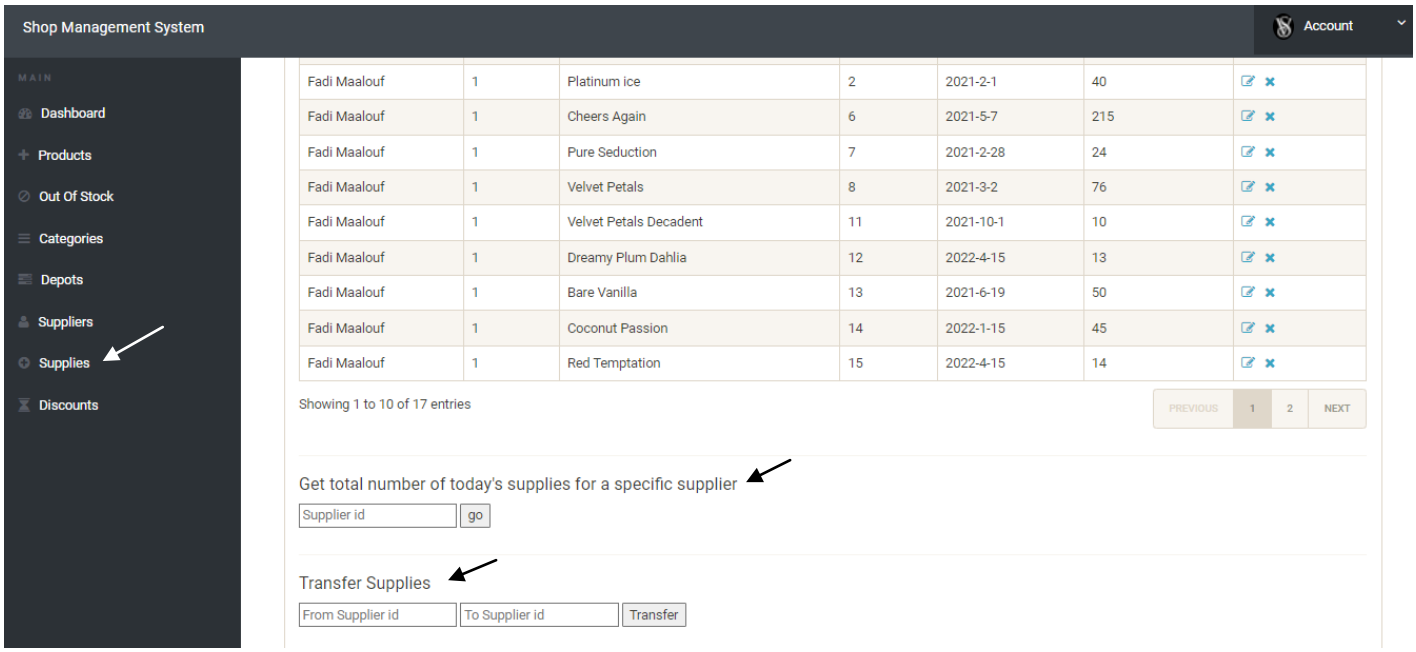
7) Suppliers Section

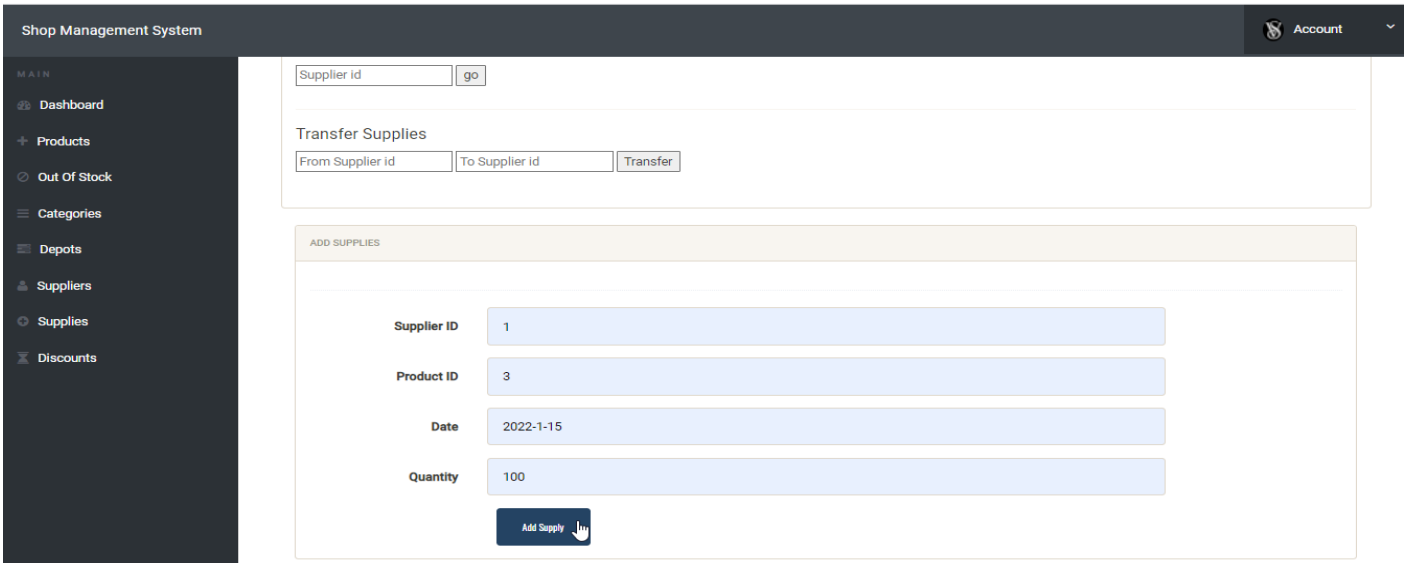


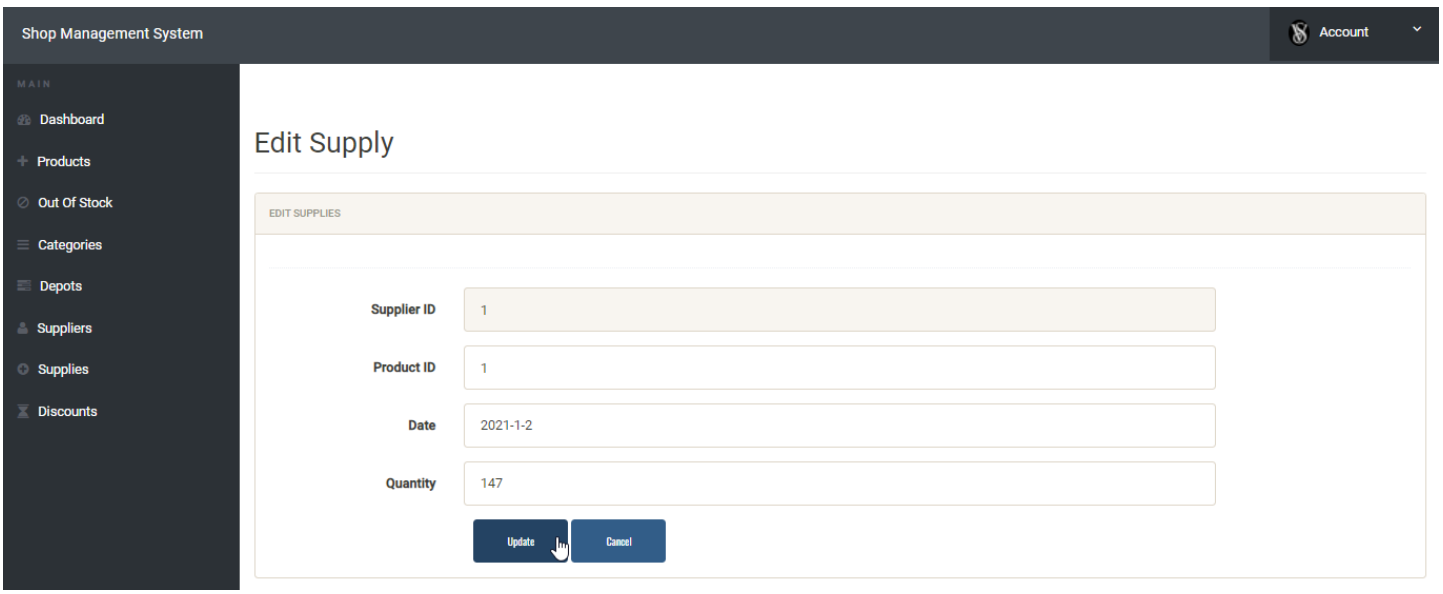

Shop Management System
Account
MAIN
Dashboard
Products
Out Of Stock
Categories
Depots
Suppliers
Supplies
Discounts
1 Fadi Maalouf 3825853 fadimaalouf@gmail.com Beirut
2 Tony Nakour 71523980 tonynakour@gmail.com Saida
3 Alaa Ghodban 81200756 alaaghodban@gmail.com Tripoli
Showing 1 to 3 of 3 entries
PREVIOUS 1 NEXT
ADD SUPPLIERS
Supplier Name supplier
Phone Number 03698457
Email Address supplier@gmail.com
Address supplier_address
Add Supplier


Shop Management System
Account
MAIN
Dashboard
Products
Out Of Stock
Categories
Depots
Suppliers
Supplies
Discounts
Edit Supplier
EDIT SUPPLIERS
Supplier ID 3
Supplier Name Alaa Ghodban
Phone Number 81200756
Email Address alaaghodban@gmail.com
Address Tripoli
Update
Cancel

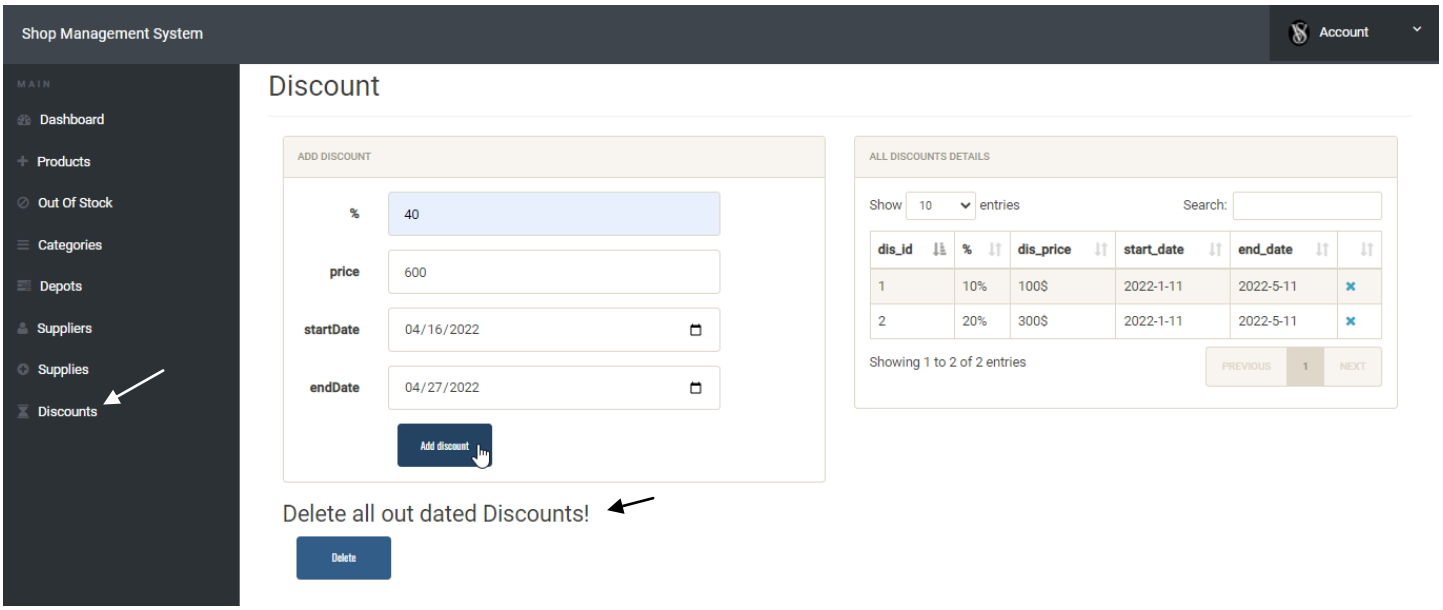
8) Supplies Section



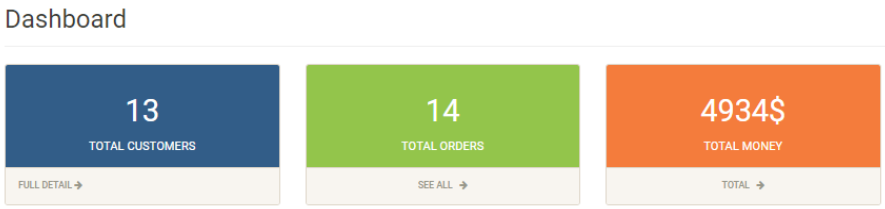


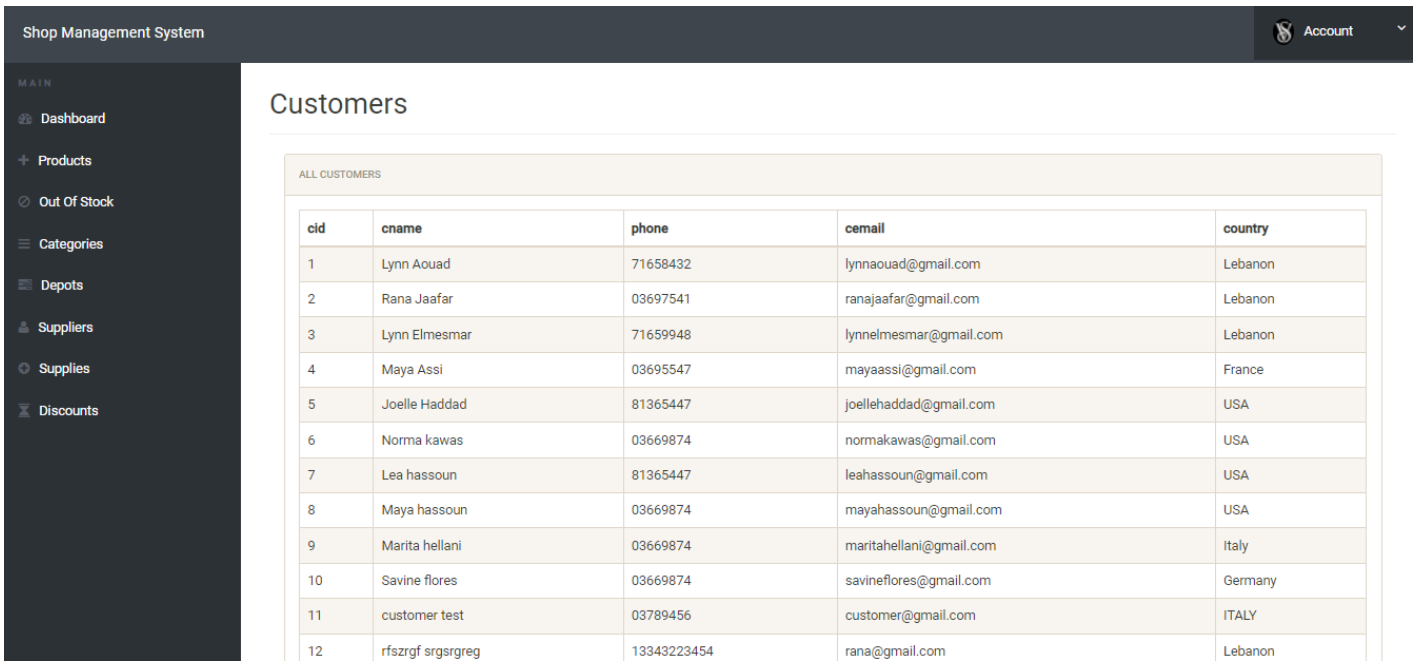


9) Discounts Section



10) Customers Details





11) Order Info

Chapter IV: Implementation and Testing

Database Connection:

User connection

```php
<?php
    $servername="HP-PC\SQLEXPRESS";
    $connectionInfo = array("Database"=>"shop", "UID"=>"USER", "PWD"=>"SHOP");

    $conn = sqlsrv_connect($servername,$connectionInfo);

    if(!$conn)
        die("Connection failed".sqlsrv_errors());
?>
```

Admin connection

```php
<?php
$servername = "HP-PC\SQLEXPRESS";
$connectionInfo = array("Database" => "shop", "UID" => "ADMIN", "PWD" => "admin123");

$conn = sqlsrv_connect($servername, $connectionInfo);

if (!$conn)
   die("Connection failed" . sqlsrv_errors());

?>
```

Some commands used in coding:

When user place order we call the **PLACE_ORDER procedure**, then we insert each product in Contain table.

```php
//CALLING A PROCEDURE TO PLACE THE ORDER (add the client info and the order info)
$dis_idd=$_SESSION['discount'];
$price_final = $_SESSION['price_final'];

$parameters = array( array($name,SQLSRV_PARAM_IN) ,
                    array($phone,SQLSRV_PARAM_IN),
                    array($email,SQLSRV_PARAM_IN) ,
                    array($country,SQLSRV_PARAM_IN),
                    array($dis_idd,SQLSRV_PARAM_IN),
                    array($address,SQLSRV_PARAM_IN));

$call_proc = "{CALL PLACE_ORDER(?,?,?,?,?,?)}";
$stmtt =  sqlsrv_query($conn, $call_proc, $parameters);

if($stmtt === false) {
    if( ($errors = sqlsrv_errors()) != null){
        foreach($errors as $error){
            echo "sqlstate:".$error['SQLSTATE']."<br>";
            echo "code:".$error['code']."<br>";
            echo "message:".$error['message']."<br>";
            echo "stmt error";}}}
```

```php
for($k=0; $k<$lenght; $k++){

    $query= "insert into Contain VALUES(".$data[$k]['id'].",".$oid.",".$data[$k]['qty'].")";

    $stmtt2 =  sqlsrv_query($conn, $query);

    if($stmtt2  === false) {
        if( ($errors = sqlsrv_errors()) != null){
            foreach($errors as $error){
                echo "sqlstate:".$error['SQLSTATE']."<br>";
                echo "code:".$error['code']."<br>";
                echo "message:".$error['message']."<br>";
                echo "stmt error";}}}
```

```
CREATE PROCEDURE PLACE_ORDER
      @CNAME    varchar(50),
      @PHONE      varchar(24),
      @CEMAIL    VARCHAR(1024),
      @COUNTRY     varchar(50),
      @DIS_ID      INT,
      @OADDRESS    varchar(1024)
AS BEGIN
    declare @cid int
    begin transaction
    exec dbo.ADD_CUSTOMER @CNAME,@PHONE,@CEMAIL,@COUNTRY,@cid out
    exec dbo.ADD_ORDER @CID,@DIS_ID, @OADDRESS
    commit
END
go
```

```
create procedure ADD_CUSTOMER
    @CNAME     varchar(50),
    @PHONE     varchar(24),
    @CEMAIL    varchar(1024),
    @COUNTRY   varchar(50),
    @CID       int   out
 as BEGIN

     if exists(select * from Customer where cname=@cname and PHONE=@PHONE and cemail=@cemail and country=@country)
        set @CID=(select cid from Customer where cname=@cname and PHONE=@PHONE and cemail=@cemail and country=@country)
     else
     BEGIN
            Insert into Customer(CNAME,PHONE,CEMAIL,COUNTRY)
                          values(@CNAME,@PHONE,@CEMAIL,@COUNTRY);

            set @CID=(select max(CID) from Customer);
      end
end
go

------------------------------------------------------------------------

create procedure ADD_ORDER
   @CID          int  ,
   @DIS_ID       int,
   @OADDRESS     varchar(1024)

as BEGIN
    declare @YY VARCHAR(5),@MM VARCHAR(5),@DD VARCHAR(5)
    declare @date_0 varchar(30), @OID int

    SET @YY=(select YEAR(GETDATE()))
    SET @MM=(select MONTH(GETDATE()))
    SET @DD=(select DAY(GETDATE()))
    set @date_0= @YY+'-'+@MM+'-'+@DD

    insert into Order_Info(CID,OADDRESS,DIS_ID,ODATE,TOTAL_AMOUNT)
                       values(@CID,@OADDRESS,@DIS_ID,@DATE_0,0);
END
go
```

When user add products into Cart we call the **FETCH_DIS procedure**, to know if there is a discount.

```php
    $parameters = array( array($carttotal,SQLSRV_PARAM_IN) ,
                         array(&$price_final,SQLSRV_PARAM_OUT),
                         array(&$pourcentage,SQLSRV_PARAM_OUT),
                         array(&$dis_id,SQLSRV_PARAM_OUT));

    $call_proc = "{CALL FETCH_DIS(?,?,?,?)}";
    $stmtt =  sqlsrv_query($conn, $call_proc, $parameters);

    // to put the result in the out parameter
    if(sqlsrv_next_result($stmtt)===false){
        if( ($errors = sqlsrv_errors()) != null){
            foreach($errors as $error){
                echo "sqlstate:".$error['SQLSTATE']."<br>";
                echo "code:".$error['code']."<br>";
                echo "message:".$error['message']."<br>";
                echo "result error";}}}


   $_SESSION["discount"]=$dis_id;
   $_SESSION["price_final"]=$price_final;
```

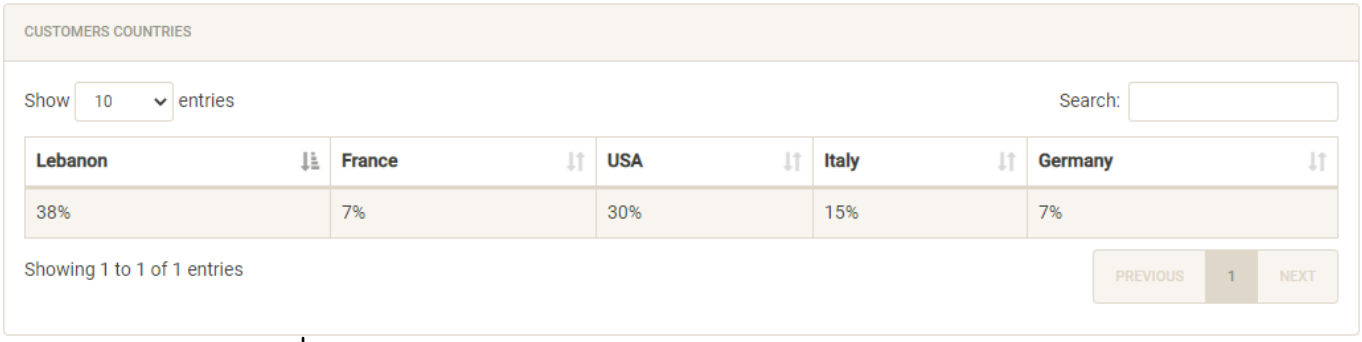
```sql
create procedure FETCH_DIS
  @price int,
  @price_final int out,
  @pourcentage decimal(3,0) OUT,
  @ID int OUT
AS
BEGIN
  declare crs_1 cursor for select dis_id,pourcentage,dis_price from dbo.DIS

  declare @dis_id int, @pr decimal(3,0) ,@dis_price int, @dis_id1 int, @pr1 int, @delivery int
  set @delivery = 5

  open crs_1
  fetch next from crs_1 into @dis_id, @pr , @dis_price

  while @@fetch_status=0
  BEGIN
    if(@dis_id1>=0)
        BEGIN
              set @pr1=(select dis_price from Discount  where dis_id=@dis_id1)  -- dis_price for the discount before!

              if(@price >= @dis_price)
              BEGIN
                  if(@pr1<@dis_price)
                    set @dis_id1=@dis_id
              end
        end
    else
        BEGIN
           if(@price >= @dis_price)
                set @dis_id1 = @dis_id
        end

  fetch next from crs_1 into @dis_id,@pr,@dis_price
  end

  if(@dis_id1>=0)
      BEGIN
        set @pr= (select pourcentage from discount where dis_id=@dis_id1)
        set @price_final= (@price-(@pr/100)*@price)+@delivery
        set @ID=@DIS_ID1
        set @pourcentage = @pr
      end

  CLOSE crs_1
  deallocate crs_1
end
go
```

When admin transform all supplies from supplier1 to supplier 2, we call the **TRANSFORME procedure**.

```php
<?php

if(isset($_POST['transfer'])){
    $s1 = trim($_POST['s1']);
    $s2 = trim($_POST['s2']);

    $sql = "SELECT count(*) as count from supplier s1, supplier s2 where s1.sid=".$s1." AND s2.sid=".$s2;
    $result =  sqlsrv_query($conn, $sql);
    $row=sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC);

    if($row['count'] == 1){


    $parameters = array( array($s1,SQLSRV_PARAM_IN) ,
    array($s2,SQLSRV_PARAM_IN));

    $call_proc = "{CALL TRANSFORME(?,?)}";
    $stmtt =  sqlsrv_query($conn, $call_proc, $parameters);

    if(sqlsrv_next_result($stmtt)===false){
        if( ($errors = sqlsrv_errors()) != null){
            foreach($errors as $error){
                echo "sqlstate:".$error['SQLSTATE']."<br>";
                echo "code:".$error['code']."<br>";
                echo "message:".$error['message']."<br>";
                echo "result error";}}}
    }
    else{
        echo "<script> alert('Please enter a valid sid!')</script>";
    }}
                ?>
```

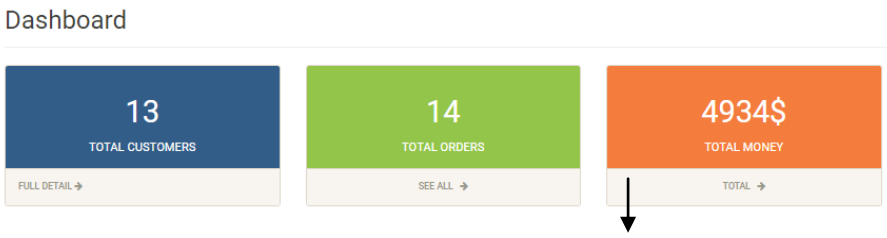
```sql
create procedure TRANSFORME
       @sid int,
       @sid1 int
as
BEGIN
  if(@sid!=@sid1)
  begin
     if ((EXISTS(select * from supplier where sid=@sid1)) and (exists(select * from supplier where sid=@sid)))
     begin
       begin transaction
         declare cursortr CURSOR
          for     select * from supply where sid=@sid
          declare @pid int,@date_s date,@qt decimal(8,0)
          open cursortr
          fetch next from cursortr into @sid,@pid,@date_s,@qt
          while @@fetch_status=0
          BEGIN
           if exists(select * from supply where pid=@pid and sid=@sid1 and sdate=@date_s)
           begin
                 update supply set QUANTITY=QUANTITY+@QT   where pid=@pid and sid=@sid1 and sdate=@date_s
                 delete supply where pid=@pid and sid=@sid and sdate=@date_s and QUANTITY=@QT
           end
           else
           BEGIN
              insert into supply VALUES(@sid1,@pid,@date_s,@qt)
              delete supply where pid=@pid and sid=@sid and sdate=@date_s AND QUANTITY=@QT
           end
           fetch next from cursortr into @sid,@pid,@date_s,@qt
          end
          CLOSE cursortr
          deallocate cursortr
       commit
     end

  end
end
go
```

In Dashboard we used multiple functions:




```php
<?php
$sql1 = "SELECT [dbo].[country_pourcentage]('Lebanon') as pr1";
$result1 =  sqlsrv_query($conn, $sql1);
$row1 = sqlsrv_fetch_array($result1, SQLSRV_FETCH_ASSOC);

$sql2 = "SELECT [dbo].[country_pourcentage]('France') as pr2";
$result2 =  sqlsrv_query($conn, $sql2);
$row2 = sqlsrv_fetch_array($result2, SQLSRV_FETCH_ASSOC);

$sql3 = "SELECT [dbo].[country_pourcentage]('USA') as pr3";
$result3 =  sqlsrv_query($conn, $sql3);
$row3 = sqlsrv_fetch_array($result3, SQLSRV_FETCH_ASSOC);

$sql4 = "SELECT [dbo].[country_pourcentage]('Italy') as pr4";
$result4 =  sqlsrv_query($conn, $sql4);
$row4 = sqlsrv_fetch_array($result4, SQLSRV_FETCH_ASSOC);

$sql5 = "SELECT [dbo].[country_pourcentage]('Germany') as pr5";
$result5 =  sqlsrv_query($conn, $sql5);
$row5 = sqlsrv_fetch_array($result5, SQLSRV_FETCH_ASSOC);
```

```sql
-- get customers country percentage
create function country_pourcentage(@coutry varchar(50))
returns INT
BEGIN
 declare @count integer,@pr integer,@RES decimal(5,2)
 set @count=(select count(*)
             from CUSTOMER)
 set @pr=(select count(*)
          from CUSTOMER
          where COUNTRY=@coutry
          )
 set @RES=(@pr*100)/(@count*1.0)
 return @RES
end
go
```




```sql
-- get total amount of all orders
create function total_amount()
returns  int
BEGIN
 declare @a int
 set @a=(select sum(total_amount) from order_info)
 return @a
end
go
```

```php
$sql = "SELECT [dbo].[total_amount]() as nb";
$result =  sqlsrv_query($conn, $sql);
$row = sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC);
```

Creating Tables:

```
create table CATEGORY (
   CAT_ID               int         IDENTITY(100,100)        not null,
   CAT_NAME             varchar(50)                          not null,
   constraint PK_CATEGORY primary key nonclustered (CAT_ID)
)
```

```
create table CUSTOMER (
   CID                  int         IDENTITY(1,1)     not null,
   CNAME                varchar(50)                   not null,
   PHONE                varchar(24)                   not null,
   CEMAIL               varchar(1024)                 not null,
   COUNTRY              varchar(50)                   not null,
   constraint PK_CUSTOMER primary key nonclustered (CID)
)
```

```
create table DISCOUNT (
   DIS_ID               int         IDENTITY(0,1)    not null,
   POURCENTAGE          decimal(3,0)                 not null,
   DIS_PRICE            int                          not null,
   START_DATE           date                         null,
   END_DATE             date                         null,
   constraint PK_DISCOUNT primary key nonclustered (DIS_ID)
)
```

```
create table ORDER_INFO (
   OID                  int     IDENTITY(4835,1)    not null,
   CID                  int                      not null,
   OADDRESS             varchar(1024)            not null,
   DIS_ID               int                      null,
   ODATE                date                     not null,
   TOTAL_AMOUNT         int                      not null,
   constraint PK_ORDER_INFO primary key nonclustered (OID),
   constraint FK_ORDER_IN_PLACED_CUSTOMER foreign key (CID) references CUSTOMER (CID),
   constraint FK_ORDER_IN_HAS_DISCOUNT foreign key (DIS_ID) references DISCOUNT (DIS_ID)
)
```

```
create table DEPOT (
   DID                  int         IDENTITY(1000,1000)      not null,
   DNAME                varchar(50)                          not null,
   DADDRESS             varchar(1024)                        not null,
   constraint PK_DEPOT primary key nonclustered (DID)
)
```

```
create table PRODUCTS (
   PID                  int         IDENTITY(1,1)       not null,
   PNAME                varchar(50)                     not null,
   PIMG                 varchar(1024)                   not null,
   CAT_ID               int                             not null,
   PRICE                int                             not null
      constraint CKC_PRICE_PRODUCT check (PRICE >= 0),
   DESCRIPTION          varchar(1024)                   not null,
   QUANTITY             decimal(8,0)                    not null
      constraint CKC_QUANTIT_PRODUCT check (QUANTITY >= 0),
   DID                  int                             not null,
   constraint PK_PRODUCT primary key nonclustered (PID),
   constraint FK_PRODUCT_STOCKED_DEPOT foreign key (DID) references DEPOT (DID),
   constraint FK_PRODUCT_IN_CATEGORY foreign key (CAT_ID) references CATEGORY (CAT_ID)
)
```

```
create table CONTAIN (
   PID                  int                 not null,
   OID                  int                 not null,
   QUANTITY             decimal(8,0)        not null
      constraint CKC_QUANTITY_CONTAIN check (QUANTITY >= 1),
   constraint PK_CONTAIN primary key (PID, OID),
   constraint FK_CONTAIN_CONTAIN_ORDER_IN foreign key (OID) references ORDER_INFO (OID),
   constraint FK_CONTAIN_CONTAIN2_PRODUCTS foreign key (PID) references PRODUCTS (PID)
)
```

```
create table SUPPLIER (
   SID                  int       IDENTITY(1,1)     not null,
   SNAME                varchar(50)                 not null,
   SPHONE               varchar(24)                 not null,
   SEMAIL               varchar(1024)               not null,
   SADDRESS             varchar(1024)               null,
   constraint PK_SUPPLIER primary key nonclustered (SID)
)
```

```
create table SUPPLY(
   SID                  int                 not null,
   PID                  int                 not null,
   SDATE                date                not null,
   QUANTITY             decimal(8,0)        not null
      constraint CKC_QUANTITY_SUPPLY check (QUANTITY >= 1),
   constraint PK_SUPPLY primary key (SID, PID, SDATE),
   constraint FK_SUPPLY_SUPPLY_PRODUCT foreign key (PID) references PRODUCTS (PID),
   constraint FK_SUPPLY_SUPPLY2_SUPPLIER foreign key (SID) references SUPPLIER (SID)
)
```

```
create table ADMINISTRATOR (
   ADM_ID                  int         IDENTITY(1,1)        not null,
   ADM_USERNAME            varchar(50)                          not null,
   ADM_PASSWORD            varchar(50)                          not null,
   constraint PK_ADMINISTRATOR primary key nonclustered (ADM_ID)
)
```

Creating Logins:

```
USE [master]
GO
CREATE LOGIN [ADMIN] WITH PASSWORD=N'admin123', DEFAULT_DATABASE=[Shop]
GO
use Shop
GO
CREATE user [admin]  for Login [ADMIN]

grant select on out_of_stock to admin
grant select,update,delete,insert on products to admin
grant select,update,delete,insert on depot to admin
grant select,update,delete,insert on category to admin
grant select,update,delete,insert on discount to admin
grant select,update,delete,insert on supplier to admin
grant select,update,delete,insert on supply to admin
grant select on order_info to admin
grant select on customer to admin
grant select on contain to admin
grant select on administrator to admin
grant  exec on DELETE_DIS to  admin
grant  exec on TRANSFORME to  admin

grant  exec on country_pourcentage to  admin
grant  exec on NB_OFSUPPLY to  admin
grant  exec on NB_OFPRODUCT to  admin
grant  exec on NB_OFORDER to  admin
grant  exec on TODAY_ORDERS to  admin
grant  exec on NB_OFORDER2 to  admin
grant  exec on total_amount to  admin
```

```
USE [master]
GO
CREATE LOGIN [USER] WITH PASSWORD=N'SHOP', DEFAULT_DATABASE=[Shop]
GO
use Shop
GO
create user [shop]  for Login [USER]


grant SELECT ON on_stock TO shop
grant SELECT,UPDATE on order_info to shop
grant SELECT,UPDATE on products to shop
grant SELECT,UPDATE,INSERT on Contain to shop
grant exec on ADD_ORDER to  shop
grant exec on ADD_CUSTOMER to  shop
grant exec on PLACE_ORDER to  shop
grant exec on FETCH_DIS to shop
grant exec on GET_QT  to shop
```

Creating Indexes:

```
create index CONTAIN_FK on CONTAIN (
OID ASC
)
go
```

```
create index STOCKED_FK on PRODUCTS (
DID ASC
)
go
```

```
create index PLACED_FK on ORDER_INFO (
CID ASC
)
go
```

```
create index IN_FK on PRODUCTS (
CAT_ID ASC
)
go
```

```
create index HAS_FK on ORDER_INFO (
DIS_ID ASC
)
go
```

```
create index SUPPLY2_FK on SUPPLY (
PID ASC
)
go
```

Creating Functions:

```
-- get quantity in stock of a specific product
create function GET_QT(@PID int)
returns  int
BEGIN
  declare @Q int
  if((select count(*) from products where pid=@pid)=1)
  BEGIN
    set @Q=(select quantity FROM products WHERE pid=@PID)
  END

 return @Q
end
go
```

```
-- get number of supplies of a specific supplier
create function NB_OFSUPPLY(@sid int)
returns  int
BEGIN
 declare @nb int
 set @nb=(select count(*) from supply
          where sid=@sid AND sdate=(cast((select YEAR(GETDATE())) as varchar(50))+'-'+
          cast(( select MONTH(GETDATE())) as varchar(50))+'-'+
          cast((select DAY(GETDATE())) as varchar(50))))
 return @nb
end
go
```

```
-- get number of today's orders
create function TODAY_ORDERS()
returns  int
BEGIN
 declare @nb int
 set @nb=(select count(*) from ORDER_INFO
          where odate=(cast((select YEAR(GETDATE())) as varchar(50))
          +'-'+cast(( select MONTH(GETDATE())) as varchar(50))
          +'-'+cast((select DAY(GETDATE())) as varchar(50))))
 return @nb
end
go
```

```
-- get number of orders in a specific date
create function NB_OFORDER2(@date_0 date)
returns integer
BEGIN
  DECLARE @nb integer
  SET @NB=(select count(*) from ORDER_INFO where Odate=@date_0)
  RETURN @NB
end
go
```

```
-- get number of products in a specific depot
create function NB_OFPRODUCT(@did int)
returns  int
BEGIN
 declare @nb int
 set @nb=(select count(*) from products where did=@did AND QUANTITY!=0)
 return @nb
end
go
```

```
-- get number of orders in a specific customer
create function NB_OFORDER(@cid int)
returns int
BEGIN
 declare @nb int
 set @nb=(select count(*) from order_INFO where cid=@cid)
 return @nb
end
go
```

Creating Triggers:

Triggers on Category Table

```
-- we can't insert 2 categories having the same name

        ---  insert ---

create trigger CATEGORY_I on Category for insert
as
begin
  declare @c int
  set @c=(select count(*) from category)

  if ((select count(*) from category t1,category t2 where t1.cat_name = t2.cat_name)!=@c)
  begin
      rollback
  end

end
go
          ---  update ---

create trigger CATEGORY_U on category for update
as
begin
  declare @c int
  set @c=(select count(*) from category)
  if ((select count(*) from category t1,category t2 where t1.cat_name = t2.cat_name)!=@c)
      rollback

end
go
```

Triggers on Supply Table

```
         ---  insert ---

create trigger SUPPLY_I on Supply for insert
as
begin
 declare crs_SUPPLY_I cursor FOR select * from inserted
 declare @pid int, @sid int, @DATE_S date, @qt decimal(8,0)

 open crs_SUPPLY_I
 fetch next from crs_SUPPLY_I into @SID,@PID,@DATE_S,@qt
 while @@fetch_status=0
 BEGIN
    update Products set quantity = quantity + @qt WHERE pid = @PID

   fetch next from crs_SUPPLY_I into @SID,@PID,@DATE_S,@qt
 end
 CLOSE crs_SUPPLY_I
 deallocate crs_SUPPLY_I
end
go
```

```
go
         --- delete ---

create trigger SUPPLY_D on Supply for delete
as
begin
   declare crs_SUPPLY_D cursor FOR select * from deleted

 declare @PID int, @SID int, @DATE_S date, @qt decimal(8,0)
 open crs_SUPPLY_D
 fetch next from crs_SUPPLY_D into @SID,@PID,@DATE_S,@qt
 while @@fetch_status=0
 BEGIN
   update Products set quantity = quantity - @qt WHERE pid = @PID
   fetch next from crs_SUPPLY_D into @SID,@PID,@DATE_S,@qt
 end
 CLOSE crs_SUPPLY_D
 deallocate crs_SUPPLY_D
end
go

         ---  update ---

create trigger SUPPLY_U on SUPPLY for update
as
begin
  declare crs_SUPPLY_U1 cursor FOR select * from deleted
  declare crs_SUPPLY_U cursor FOR select * from inserted

  declare @pid int, @sid int, @DATE_S date, @qt decimal(8,0)
  declare @pid1 int, @sid1 int, @DATE_S1 date , @qt1 decimal(8,0)

   open crs_SUPPLY_U
   fetch next from crs_SUPPLY_U into @SID,@PID,@DATE_S,@qt
   open crs_SUPPLY_U1
   fetch next from crs_SUPPLY_U1 into @SID1,@PID1,@DATE_S1,@qt1
   while @@fetch_status=0
    BEGIN

        update Products set quantity = quantity - @qt1 where pid = @pid1

        update Products set quantity = quantity + @qt where pid = @pid

        fetch next from crs_SUPPLY_U into @SID,@PID,@DATE_S,@qt
        fetch next from crs_SUPPLY_U1 into @SID1,@PID1,@DATE_S1,@qt1
    end
  CLOSE crs_SUPPLY_U
  deallocate crs_SUPPLY_U

 CLOSE crs_SUPPLY_U1
  deallocate crs_SUPPLY_U1
end
go
```

Trigger on Contain Table

```
        ---  insert ---

create trigger CONTAIN_I on Contain for insert
as
begin
  if exists(select 1 from products t1,inserted t2 where t1.pid=t2.pid and t1.quantity<t2.quantity)
    rollback

  declare crs_CONTAIN_I cursor FOR select * from inserted
  declare @OID int, @PID int, @qt decimal(8,0), @q decimal(8,0), @i CHAR(1), @pr int

  open crs_CONTAIN_I
  fetch next from crs_CONTAIN_I into  @PID,@OID, @qt
  while @@fetch_status=0
   BEGIN
        set @pr = (select price from Products where pid=@PID)

        update Products set quantity = quantity-@qt where pid=@PID
        update Order_Info set TOTAL_AMOUNT = TOTAL_AMOUNT + @qt*@pr  where oid=@OID


      fetch next from crs_CONTAIN_I into @PID,@OID,@qt
   END
  CLOSE crs_CONTAIN_I
  deallocate crs_CONTAIN_I
 END
go
```

Creating Views:

```
CREATE VIEW out_of_stock
AS select * from Products WHERE quantity=0
go
-----------------------------------------------------

CREATE VIEW on_stock
AS select * from Products WHERE quantity>0
go
-----------------------------------------------------

CREATE VIEW DIS
as  select *
    from discount
    where (cast((select YEAR(GETDATE())) as varchar(50))+'-'+cast(( select MONTH(GETDATE())) as varchar(50))
    +'-'+cast((select DAY(GETDATE())) as varchar(50)) between start_date and end_date)
                or
          (END_DATE  IS NULL and START_DATE is NULL)
go
-----------------------------------------------------

CREATE VIEW ORDERTODAY
as select oid,total_amount
   from order_info
   where cast((select YEAR(GETDATE())) as varchar(50))+'-'+cast(( select MONTH(GETDATE())) as varchar(50))
   +'-'+cast((select DAY(GETDATE())) as varchar(50))=odate
 go
```

Creating Rest of the Procedures:

```
-- delete all out dated discounts

create procedure DELETE_DIS
AS
begin
  declare @YY VARCHAR(5),@MM VARCHAR(5),@DD VARCHAR(5), @date DATE, @DIS_ID INT

  SET @YY=(select YEAR(GETDATE()))
  SET @MM=( select MONTH(GETDATE()))
  SET @DD=(select DAY(GETDATE()))
  set @date= @YY+'-'+@MM+'-'+@DD

  declare crs CURSOR FOR select DIS_ID
                          FROM DISCOUNT
                          WHERE END_DATE < @date

  open crs
  fetch next from crs into @DIS_ID
  while @@fetch_status=0
  BEGIN
     UPDATE ORDER_INFO SET DIS_ID=NULL WHERE DIS_ID=@DIS_ID
     delete from DISCOUNT where DIS_ID=@DIS_ID
     fetch next from crs into @DIS_ID
  END
  CLOSE crs
  deallocate crs
end
go
```

Conclusion

This is the modern age, this system will allow customers to place orders without even visiting the shop, being able to buy anything any time, It became more enjoyable and easier than the real-world shopping.