Team – 5

Shoe Stop Application

1.INTRODUCTION:

1.1 Project Overview and Statement of Proposal

The purpose of our project is to design a relational database that will provide backend engineering and support for an e-commerce web application that sells a diverse selection of shoes as products. The application will provide a means for users (customers) to create a profile, browse and shop for shoes from the e-commerce inventory, browse and shop for shoes according to their customized profile, and then complete a purchase order of one or more pairs of shoes. A unique aspect of our e-commerce design is the ability for customers to specify a certain preference for shoes (sorted by shoe type, size, color, etc.) in order to enhance user experience on the web application.

1.2 Statement of Proposal:

We propose to design a relational database schema for an e-commerce web application. The main purpose of this web application is to provide customers with an opportunity to browse and purchase a broad collection of shoes, with a focus on personalization of each customer's preference.

Project Scope and Objectives:

The scope of the e-commerce web application is encompassed by the customer's preference of the product (shoes), as well as the entire e-commerce stock inventory. The objective is to maximize personalization of the product in order to satisfy the customer's choice of shoes and enhance user experience while shopping.

Our e-commerce web application design includes certain business functions that allow the customer to easily navigate the application, review their purchase and order history, participate in discounts and specials, as well as manage their profile.

Business Functions:

1. Login
2. Register user
3. Update Profile
    a. Create profile
    b. Update profile
    c. Delete profile
    d. Reset password
4. Payment details
5. Customer's order history
6. View cart
7. Frequently Asked Questions
8. Deals of the Day
9. VIP customer
10. Now trending

2. REQUIREMENT ANALYSIS:

Data on the shoes (color, brand name, size, category), customer (name, address, etc.), payment (payment information for each customer, card information, etc.), will be stored in the database. The application provides the customers a way to shop for shoes, select shoes, and place into the shopping cart to be purchased later. A business function to enhance user experience is called 'Now trending', and is a list comprised of popular shoes recently purchased by other customers who use the e-commerce application. Once a pair of shoes is purchased, the application keeps a log of the customer's orders by an order history for each customer.

The application allows for sales of shoes along with a specific discount, called Deals of the Day. These sales will be different each day and is optional for each customer. Other specials will only be given to customers who purchase a minimum of five pairs of shoes within a span of one month. These customers will become VIP members and will receive a 20% discount off their next purchase.

The admin is a special user of the application who has master control to add, remove, and update the database. Customers will have interaction with the web application by customization/personalization with their shoe purchases, as well as providing reviews/feedback on shoes.

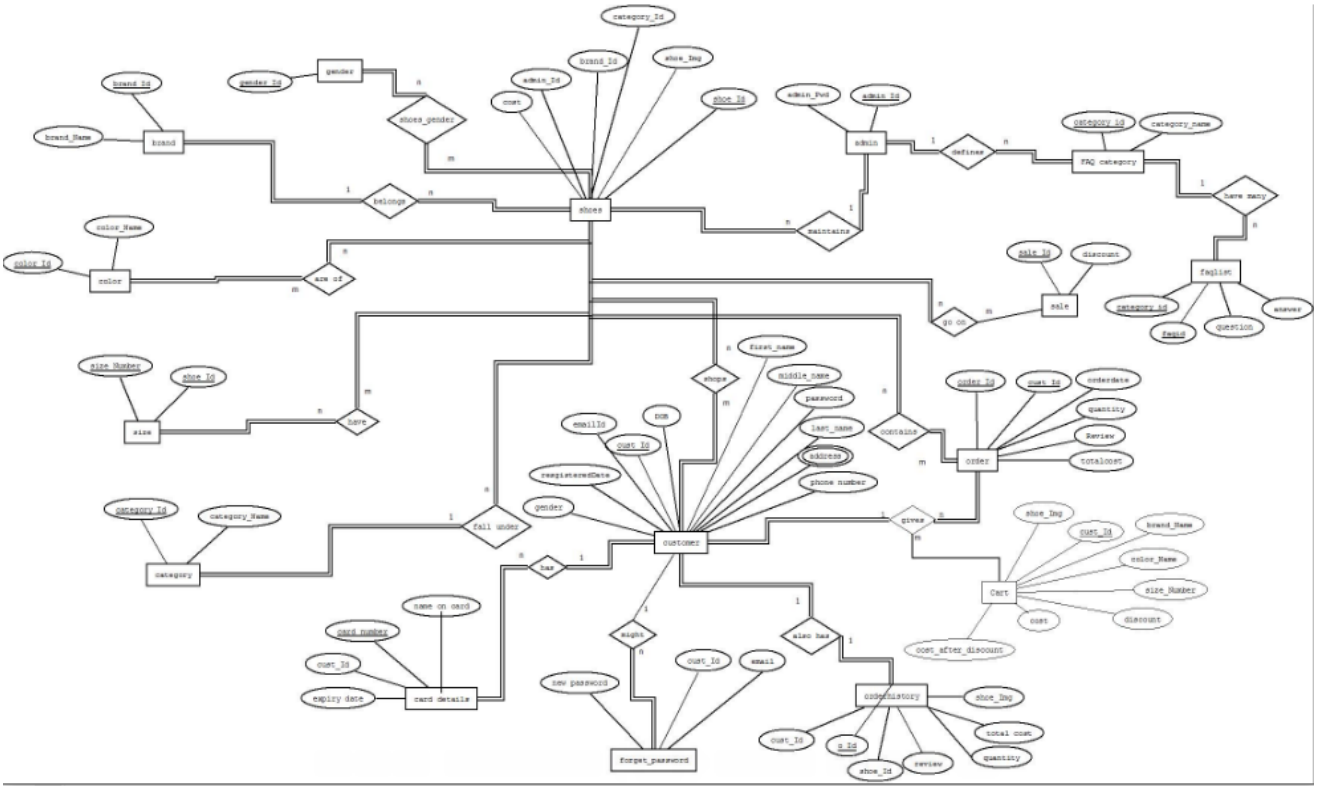
During the development of this project, we used a public server to test and merge our code. This server can be accessed by visting the following URL: http://104.236.29.246

However to run this website locally and for yourself you must have basic understanding of how to setup a PHP webserver, along with a MySQL server. Ater that simple copy the files to your webserver. The files are:

- about.html
- c.png
- cart.png
- contact.html
- css
- database_connection.php
- delete-profile.php
- edit-profile.html.php
- edit-profile.php
- faq.html.php
- fonts
- forgotpwd.html
- goback.php
- images
- index.php
- insertcart.php
- insertorders.php
- js
- kids.php
- login-nav.html.php
- login.html
- login.php
- logout.php
- main-header.html.php
- mens.php
- nav.html.php
- OrderHistory.html.php
- Orders.php
- Payment.html
- payment.html.php
- payment.php
- register.html
- register.php
- resetpwd.html.php
- resetpwd.php
- ShoppingCart.html.php
- ShoppingCart.php
- success.html
- trending.html
- trending.php
- updaterating.php
- womens.php

3. ER MODEL DESIGN

3.1 ER Diagram:



3.2 Description of the Entity Relationship Diagram:

1. The e-commerce web application has a collection of shoes. Each **shoe** should have a unique identifier. Each shoe must fall under only one subclass: Men, Women, or Children. Each shoe will have a brandID, sizeNumber, and colorID.

2. **Customers** use the web application. The customers have a autoincremented unique ID, a firstName, middleName, lastName, and dateOfBirth, email/userID, password, phoneNumber, billingAddress, shippingAddress, securityQuestion, and securityAnswer.

3. Gender defines the gender category of the shoe i.e men, women, and kids.

4. Each **sale** has a unique ID and a discount. More than one shoe can go on sale.

5. An **order** for shoes is made by customers. Each order has a unique ID, shoeID, customerID, orderDate, and orderTime.

6. Once a customer makes an order, the order details get updated into the **order history** of that customer. Each order history entry has an order ID, orderDate, orderTime, shoeID and customerID.

7. A ternary relationship connects **Customer, Order and Cart**.

8. A customer might forget their password. The web application has an entity for **forgetPassword**. The attributes include a customerID, userName, email, and newPassword.

9. The web application allows the customer to add one or more cards for their **payment**. Each card includes a customerID, cardNumber, expirationDate, and cardName.

10. Each shoe falls under a specific **category**. The categories have a categoryName and categoryID.

11. Each shoe has a **brand**. Each brand has a brandName and brandID.

12. Each shoe has a **size**. Each size has a sizeNumber.

13. Each shoe has a **color**. Each color has a colorName and colorID.

14. The web application has one **Admin**. The Admin has a masterID and masterPassword, and shoeID. The Admin controls the shoes by adding, removing, updating data.

15. **FAQ category** is defined by **admin** and it determines the category in which the questions fall.

16. Each **FAQ category** has an **FAQ list** which is the list of questions with respective answers.

Each customer can purchase one or more shoes, and each customer can add one or more card information for payment. Each customer has at most one order history. Each order history can have multiple orders. Each customer can place one or more orders by shopping for one or more shoes.

4. ER Model Mapping to Relational Database:

4.1 SQL statements for entities and relationships creation:

```
DROP DATABASE IF EXISTS shoestore;

create database shoestore;

use shoestore;

#tables for admin

create table admin

(

        admin_Id char (5) NOT NULL,

        admin_Pwd varchar (10) NOT NULL,

        primary key(admin_Id)

);


#tables for shoes

#USE THESE FOR ALL TABLES; ALL PRIMARY KEYS CHAR(5),

create table brand

(

        brand_Id char (5),

        brand_Name varchar (30) NOT NULL,

        primary key (brand_Id)

);



create table categories
```

```
(
        category_Id char (5),
        category_Name varchar (30) NOT NULL,
        primary key (category_Id)
);
create table shoes
#table for shoes and maintains as it is a one to many relation along with brand table and categories
(
        admin_Id char (5),
        brand_Id char (5) NOT NULL,
        shoe_Id char (5) NOT NULL,
        shoe_Img BLOB NOT NULL,
        category_Id char (5) NOT NULL,
        cost decimal (6,2) NOT NULL,
        primary key(shoe_Id),
        foreign key (admin_Id) references admin(admin_Id),
        foreign key (brand_Id) references brand(brand_Id),
        foreign key (category_Id) references categories(category_Id)
);

create table gender
(
gender_id char (10) NOT NULL,
primary key (gender_id)
);
create table color
(
        color_Id char (5),
        color_Name varchar (15) NOT NULL,
```

```
        primary key(color_Id)
);
create table size
(
        size_Number float (5,1) NOT NULL,
        shoe_Id char (5) NOT NULL,
        primary key (size_Number,shoe_Id),
        foreign key (shoe_Id) references shoes(shoe_Id)
);
create table sale
(
        discount decimal (9,9) NOT NULL,
        sale_Id char (5) NOT NULL,
        primary key(sale_Id)
);
create table customer
(
        id int(11) NOT NULL AUTO_INCREMENT,
        dob date NOT NULL,
        first_name varchar (30) NOT NULL,
        middle_name varchar (30),
        last_name varchar (30) NOT NULL,
        gender varchar (6) NOT NULL,
        password varchar (8) NOT NULL,
        emailId varchar (40) NOT NULL,
        registeredDate date NOT NULL,
        primary key (id)
);
create table phonenumber
```

```
(
        ph_no bigint (13) NOT NULL,
        cust_Id int (11) NOT NULL,
        primary key (ph_no, cust_Id),
        foreign key  (cust_Id) references customer(id)
        ON DELETE CASCADE
);
create table address
(
        address varchar (200) NOT NULL,
        cust_Id int NOT NULL,
        type varchar (20) NOT NULL,
        foreign key (cust_Id) references customer (id)
        ON DELETE CASCADE
);
create table orders
(
        order_Id char (5) NOT NULL,
        cust_Id int NOT NULL,
        orderdate date NOT NULL,
        quantity int NOT NULL,
        Review decimal (4,2) NOT NULL,
        totalcost decimal (6,2) NOT NULL,
        primary key (order_Id, cust_Id),
        foreign key(cust_Id) references customer (id)
);
create table alsohasorderhistory
(
        order_Id int(10) NOT NULL AUTO_INCREMENT,
```

```
	cust_Id int NOT NULL,
	orderdate date NOT NULL,
	shoe_Id char(5) NOT NULL,
	quantity int NOT NULL,
	Review decimal(4,2) NOT NULL,
	totalcost decimal(6,2) NOT NULL,
	primary key(order_Id),
	foreign key(cust_Id) references customer (id)
);
create table mightforgotpassword
(
	cust_Id int NOT NULL,
	emailId varchar(40) NOT NULL,
	newpwd varchar(8),
	primary key (cust_Id),
	foreign key(cust_Id) references customer (id)
);
create table hascarddetails
(
	cust_Id int NOT NULL,
	card_no bigint(16) NOT NULL,
	nameoncard varchar(60) NOT NULL,
	exp_date date NOT NULL,
	primary key(cust_Id, card_no),
	foreign key(cust_Id) references customer (id)
);

#Tables for relations areof,havea,alsohas,go_on,contains,shop
create table shoe_gender
```

```
(
gender_id char (10) NOT NULL,
shoe_id char (10) NOT NULL,
primary key (gender_id,shoe_id),
foreign key (gender_id) references gender (gender_id),
foreign key (shoe_id) references shoes(shoe_id)
);
create table shoesareofcolor
(
color_Id char (5) NOT NULL,
shoe_Id char (5) NOT NULL,
primary key (color_Id,shoe_Id),
foreign key (color_Id) references color(color_Id),
foreign key (shoe_Id) references shoes(shoe_Id)
ON UPDATE CASCADE
);
create table shoeshavesize
(
size_Number float (5,1) NOT NULL,
shoe_Id char (5) NOT NULL,
primary key (size_Number,shoe_Id),
foreign key(size_Number) references size(size_Number),
foreign key(shoe_Id) references shoes(shoe_Id)
ON UPDATE CASCADE
);
create table shops
(
shoe_Id char (5) NOT NULL,
cust_Id int NOT NULL,
```

```
primary key (shoe_Id,cust_Id),
foreign key (shoe_Id) references shoes (shoe_Id),
foreign key (cust_Id) references customer (id)
ON UPDATE CASCADE
);
create table shoesgoondeal
(
shoe_Id char(5) NOT NULL,
sale_Id char(5) NOT NULL,
primary key (shoe_Id,sale_Id),
foreign  key(shoe_Id) references shoes(shoe_Id),
foreign  key(sale_Id) references sale(sale_Id)
ON UPDATE CASCADE
);
create table contains
(
shoe_Id char(5),
order_Id char(5),
primary key (shoe_Id,order_Id),
foreign  key(shoe_Id) references shoes(shoe_Id),
foreign  key(order_Id) references orders(order_Id)
ON UPDATE CASCADE
);
CREATE TABLE IF NOT EXISTS `admindefinesfaqcategory`
(
 `category_id` varchar(4) NOT NULL,
 `category_name` varchar(56) NOT NULL,
`admin_Id` char(5),
PRIMARY KEY (`category_id`,`admin_id`),
```

```
foreign key (admin_id) references admin(admin_id)
);
CREATE TABLE IF NOT EXISTS `faqlist`
(
`category_id` varchar (4) NOT NULL,
`faqid` varchar (4) NOT NULL,
`question` varchar (56) NOT NULL,
`Answer` varchar (456) NOT NULL,
PRIMARY KEY (`category_id`,`faqid`),
foreign key (category_id) references admindefinesfaqcategory(category_id)
);
#cart
create table cart
(
   shoe_Img BLOB NOT NULL,
   shoe_Id char (5) NOT NULL,
   cust_Id int NOT NULL,
   brand_Name varchar (30) NOT NULL,
   color_Name varchar (15) NOT NULL,
   size_Number float (5,1) NOT NULL,
   discount decimal (9,9) NOT NULL,
   cost decimal (6,2) NOT NULL,
   cost_after_discount decimal(9,2) NOT NULL,
   #primary key (shoe_Id,cust_Id),
   foreign key(shoe_Id) references shoes(shoe_Id),
   foreign key(cust_Id) references customer (id)
);
```

4.2 SQL statements for business functions:

//LOGIN

```
SELECT * FROM customer
WHERE emailid = "Johndoe@gmail.com" AND password = "John@123";
```

id	dob	first_name	middle_name	last_name	gender	password	emailId	registeredDate
1	1990-02-22	john	NULL	doe	Male	John@123	Johndoe@gmail.com	2016-12-09

//REGISTER

```
INSERT INTO customer (dob, first_name, last_name, gender, password, emailId,registeredDate)
 VALUES ('1994-02-22', 'john', 'patson', 'Male', 'Jonp@123', 'Johnpatson@gmail.com',curdate()
);
```

id	dob	first_name	middle_name	last_name	gender	password	emailId	registeredDate
1	1990-02-22	john	NULL	doe	Male	John@123	Johndoe@gmail.com	2016-12-09
2	1989-07-10	mary	NULL	rose	Female	Mary@123	Mary@gmail.com	2016-12-09
3	1992-07-23	Usman	NULL	khan	Male	Usma@123	Usman@gmail.com	2016-12-09
4	1994-08-17	Anudeesh	NULL	U	Male	Desh@123	Deesh@gmail.com	2016-12-09
5	1991-06-18	Narendra	NULL	Modi	Male	Modi@123	Modi@gmail.com	2016-12-09
6	1994-02-22	john	NULL	patson	Male	Jonp@123	Johnpatson@gmail.com	2016-12-09

/** to get all info for selected customer id **/

SELECT * FROM customer WHERE id = ‘6’;

```
SELECT * FROM customer WHERE id = 6;
```

id	dob	first_name	middle_name	last_name	gender	password	emailId	registeredDate
6	1994-02-22	john	NULL	patson	Male	Jonp@123	Johnpatson@gmail.com	2016-12-09

//UPDATE PROFILE

/** to update customer first name and last name **/

```
UPDATE customer
SET first_name = 'Anna', last_name = 'Smith'
WHERE id = 1
```

1	1990-02-22	Anna	NULL	Smith	Male	John@123	Johndoe@gmail.com	2016-12-10

/** to update customer email id **/

```
UPDATE customer

SET emailid = 'newemail@gmail.com'

WHERE id = 1
```

1	1990-02-22	Anna	NULL	Smith	Male	John@123	newemail@gmail.com	2016-12-10

/** to update phone number for selected customer id **/

```
UPDATE phonenumber

SET ph_no = '1234358809'

WHERE id=1
```

ph_no	cust_Id
1234358809	1

/** update address for selected customer id **/

```
UPDATE address

SET address = 'UT, 1101, NC -28208'

WHERE cust_id = 'ct104'
```

address	cust_Id	type
UT, 1101, NC -28208	1	shipping

//RESET PASSWORD

/** to reset password for selected customer id **/

```
SELECT emailid
FROM customer
WHERE id = 6
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

emailid
Johnpatson@gmail.com

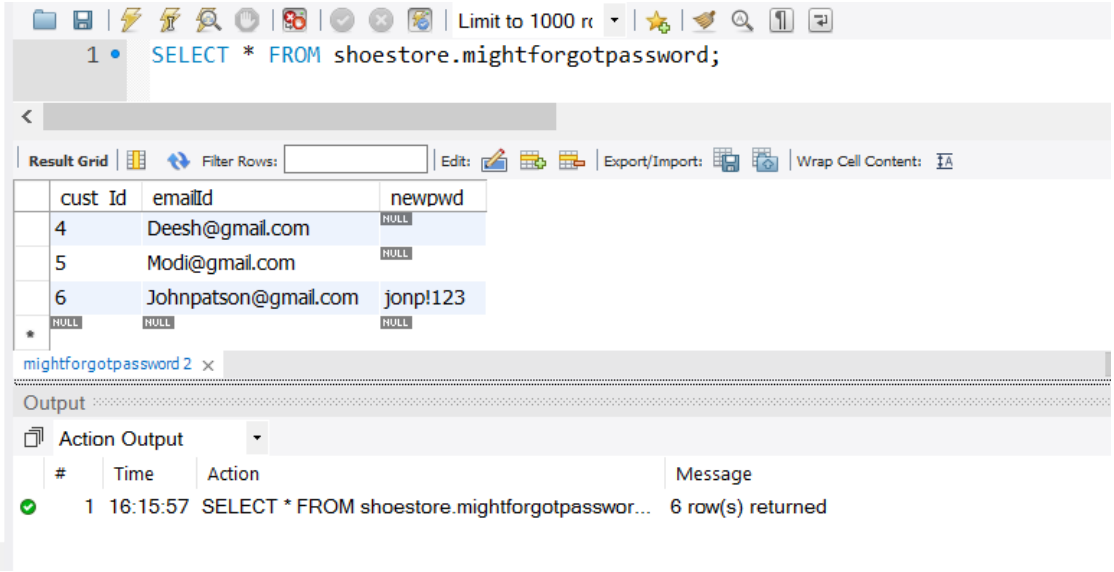
customer 1

```
Insert into mightforgotpassword
(cust_id,emailId,newpwd) values(6,'Johnpatson@gmail.com','jonp!123');
```

Output

Action Output

#	Time	Action	Message
1	16:14:59	Insert into mightforgotpassword (cust_id,emailId,n...	1 row(s) affected



//PAYMENT DETAILS

//add new card

Insert into hascarddetails

values('6','2564775758392040','puneeth','2021-05-22');

cust_Id	card_no	nameoncard	exp_date
1	1144223355664478	johndoe	2019-10-10
2	1177000011112222	maryrose	2020-08-25
3	4444888899992222	usmankhan	2017-11-28
4	4477999933335555	anudessh u	2019-04-19
5	4141525289897632	narendra modi	2021-09-09
6	2564775758392040	puneeth	2021-05-22
NULL	NULL	NULL	NULL

//delete existing card

```
1  delete from hascarddetails
2  where cust_Id='6' AND card_no='2564775758392040';
3
4
```

Output

Action Output

#	Time	Action	Message
1	15:58:49	delete from hascarddetails where cust_Id='6' AN...	1 row(s) affected

```
1  SELECT * FROM shoestore.hascarddetails;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

cust_Id	card_no	nameoncard	exp_date
1	1144223355664478	johndoe	2019-10-10
2	1177000011112222	maryrose	2020-08-25
3	4444888899992222	usmankhan	2017-11-28
4	4477999933335555	anudessh u	2019-04-19
5	4141525289897632	narendra modi	2021-09-09
NULL	NULL	NULL	NULL

//change details

```
1  update hascarddetails
2  set card_no='2564775758392040'
3  where cust_Id='5' AND card_no='4141525289897632';
4
```

Output

Action Output

#	Time	Action	Message
1	16:03:24	update hascarddetails set card_no='2564775758...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0

```
1 • SELECT * FROM shoestore.hascarddetails;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

cust_Id	card_no	nameoncard	exp_date
1	1144223355664478	johndoe	2019-10-10
2	1177000011112222	maryrose	2020-08-25
3	4444888899992222	usmankhan	2017-11-28
4	4477999933335555	anudessh u	2019-04-19
5	2564775758392040	narendra modi	2021-09-09
NULL	NULL	NULL	NULL

//ORDER HISTORY

SELECT s.shoe_Img image, s.brand_Id brand, a.order_Id orderid,
s.shoe_Id shoeid, a.orderdate orderdate, a.quantity quantity,
a.totalcost totalcost, a.Review rating
FROM alsohasorderhistory a ,
shoes s where a.shoe_ID = s.shoe_Id and cust_Id = 4;

```
1 • SELECT s.shoe_Img image, s.brand_Id brand, a.order_Id orderid,
2    s.shoe_Id shoeid, a.orderdate orderdate, a.quantity quantity,
3    a.totalcost totalcost, a.Review rating
4   FROM alsohasorderhistory a ,
5   shoes s where a.shoe_ID = s.shoe_Id and  cust_Id = 4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

image	brand	orderid	shoeid	orderdate	quantity	totalcost	rating
BLOB	Ad102	4	109	2016-10-19	1	140.00	4.50

Result 8

Output

Action Output

#	Time	Action	Message
1	12:01:51	SELECT s.shoe_Img image, s.brand_Id brand, a.order_Id orderid,...	1 row(s) returned

//CART

/*insert into statement*/

insert into cart(`shoe_Img`,`shoe_Id`,`cust_Id`,`brand_Name`,`color_Name`,
`size_Number`,`discount`,`cost`,`cost_after_discount`)
values ('s3.jpg','103','5','Fila','c103','8.0','0.5','60','30');

```
insert into cart(`shoe_Img`,`shoe_Id`,`cust_Id`,`brand_Name`,`color_Name`,
`size_Number`,`discount`,`cost`,`cost_after_discount`)
values ('s3.jpg','103','5','Fila','c103','8.0','0.5','60','30');
```

Output

Action Output

#	Time	Action	Message
1	12:46:41	insert into cart(`shoe_Img`,`shoe_Id`,`cust_Id`,`brand_Name`,`color...	1 row(s) affected

//select from cart

select shoes.shoe_Img as image, shoes.shoe_Id as shoeid,
customer.id as custid, brand.brand_Name brand, color.color_Name as ccolor,
size.size_Number as snumber,
ROUND(sale.discount,2) as cdiscount, ROUND(shoes.cost,2) as scost,
ROUND(shoes.cost*(1-sale.discount),2) as cad
from cart, customer, shoes, brand, color, size, shoesareofcolor, shoesgoondeal, sale where
cart.cust_Id = customer.id and
cart.shoe_Id = shoes.shoe_Id and shoes.shoe_Id = shoesareofcolor.shoe_Id and
shoesareofcolor.color_Id = color.color_Id and shoes.shoe_Id = shoesgoondeal.shoe_Id and
shoesgoondeal.sale_Id = sale.sale_Id and shoes.shoe_Id=size.shoe_Id and
shoes.brand_Id=brand.brand_Id and cart.cust_Id=5

```
select shoes.shoe_Img as image, shoes.shoe_Id as shoeid,
customer.id as custid, brand.brand_Name brand, color.color_Name as ccolor,
size.size_Number as snumber,
ROUND(sale.discount,2) as  cdiscount, ROUND(shoes.cost,2) as scost,
ROUND(shoes.cost*(1-sale.discount),2) as cad
from cart, customer, shoes, brand, color, size, shoesareofcolor, shoesgoondeal,  sale where
cart.cust_Id = customer.id and
cart.shoe_Id = shoes.shoe_Id and shoes.shoe_Id =    shoesareofcolor.shoe_Id and
shoesareofcolor.color_Id = color.color_Id and shoes.shoe_Id = shoesgoondeal.shoe_Id and
shoesgoondeal.sale_Id = sale.sale_Id and  shoes.shoe_Id=size.shoe_Id and
shoes.brand_Id=brand.brand_Id  and cart.cust_Id=5
```

Result Grid

image	shoeid	custid	brand	ccolor	snumber	cdiscount	scost	cad
BLOB	103	5	Fila	white	8.0	0.20	60.00	48.00

Result 13 — Read Only

Output

Action Output

#	Time	Action	Message
1	13:03:26	select shoes.shoe_Img as image, shoes.shoe_Id as shoeid, custo...	1 row(s) returned

/* delete from cart */

delete from cart where shoe_Id = '105';

```
1 • delete from cart where shoe_Id = '105';
```

Output

Action Output

#	Time	Action	Message
1	13:07:43	delete from cart where shoe_Id = '105'	1 row(s) affected

//FREQUENTLY ASKED QUESTIONS

```
1 • SELECT * FROM faqlist,admindefinesfaqcategory
2 •  WHERE admindefinesfaqcategory.category_id = faqlist.category_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_id	faqid	question	Answer	category_id	category_nam
c001	q001	How can I track my order?	Once you have submitted your order...	c001	Order status
c001	q002	How can I change/cancel my order?	nfortunately, once an order has been...	c001	Order status
c002	q001	How can I return/exchange my order?	Please allow 1-2 business days for ha...	c002	Returns & exch
c003	q001	What are your size conversions?	Our footwear is labeled with US sizes	c003	Product informa
c003	q002	How can I locate styles available in m...	You can shop our full collection by siz...	c003	Product informa
c004	q001	is it safe to shop on our site?	yes.. it is safe to shop on our site	c004	our Website
c005	q001	are there any stores available	no.. not yet started any physical stores	c005	stores

Result 4

Output

Action Output

#	Time	Action	Message
1	16:19:27	SELECT * FROM faqlist,admindefinesfaqcategor...	7 row(s) returned

//FUNCTION DISCOUNT

```
SELECT shoes.shoe_id, Shoe_img,brand_Name, category_name,
 (cost-cost*discount) as new_cost, discount, cost as actual_cost, color_Name,
 SUBSTRING(gender.gender_id, 1, CHAR_LENGTH(gender.gender_id) - 3) AS gender FROM
sale, shoes, brand, categories, shoesgoondeal, color, shoesareofcolor,
gender, shoe_gender where
sale.sale_id = shoesgoondeal.sale_id and
shoesgoondeal.shoe_id = shoes.shoe_id and
shoes.brand_id = brand.brand_id and
categories.category_id = shoes.category_id and
color.color_id = shoesareofcolor.color_id and
shoesareofcolor.shoe_id = shoes.shoe_id and
gender.gender_id = shoe_gender.gender_id and
shoes.shoe_id = shoe_gender.shoe_id and
discount > 0
```

Output

Action Output

#	Time	Action	Message
1	16:31:03	SELECT shoes.shoe_id, Shoe_img,brand_Nam...	12 row(s) returned

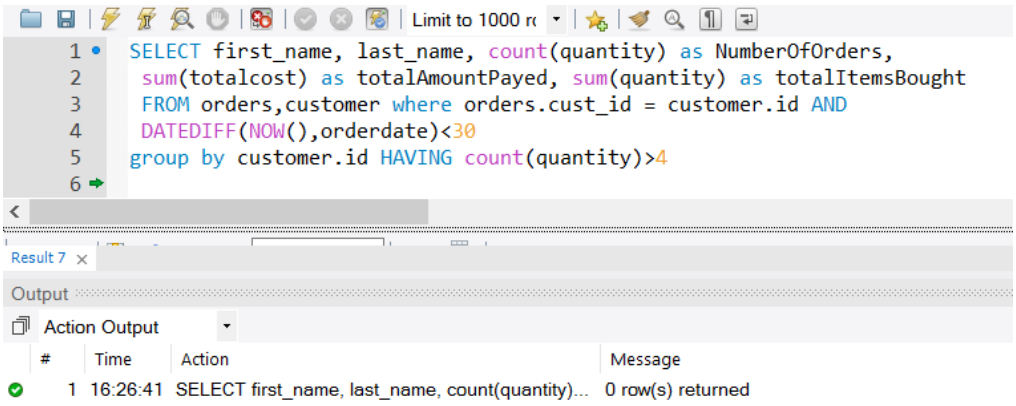
```
SELECT shoes.shoe_id, Shoe_img,brand_Name, category_name,
 (cost-cost*discount) as new_cost, discount, cost as actual_cost, color_Name,
 SUBSTRING(gender.gender_id, 1, CHAR_LENGTH(gender.gender_id) - 3) AS gender FROM
sale, shoes, brand, categories, shoesgoondeal, color, shoesareofcolor,
gender, shoe_gender where
sale.sale_id = shoesgoondeal.sale_id and
shoesgoondeal.shoe_id = shoes.shoe_id and
shoes.brand_id = brand.brand_id and
categories.category_id = shoes.category_id and
color.color_id = shoesareofcolor.color_id and
shoesareofcolor.shoe_id = shoes.shoe_id and
gender.gender_id = shoe_gender.gender_id and
shoes.shoe_id = shoe_gender.shoe_id and
discount > 0
```

Result Grid

shoe_id	Shoe_img	brand_Name	category_name	new_cost	discount	actual_cost	color_Name	gend
112	BLOB	Croc	Casuals	165.00000000000	0.250000000	220.00	red	KIDS
113	BLOB	Fila	Sports	32.50000000000	0.500000000	65.00	white	KIDS
102	BLOB	Croc	Casuals	35.00000000000	0.300000000	50.00	black	MEN
103	BLOB	Fila	Sports	48.00000000000	0.200000000	60.00	white	MEN
104	BLOB	Reebok	Loafers	42.00000000000	0.400000000	70.00	grey	MEN

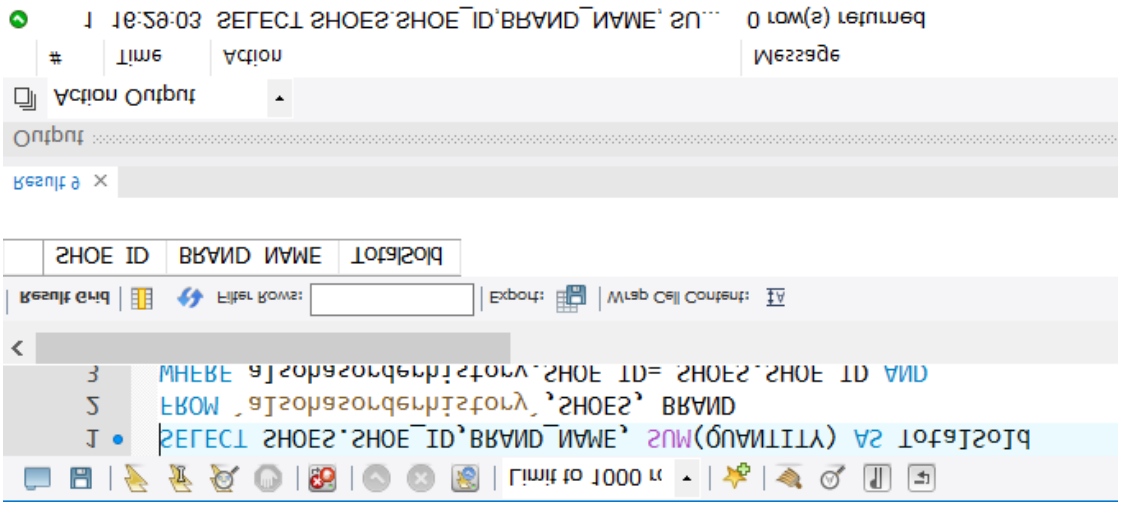
//VIP CUSTOMERS

SELECT first_name, last_name, count(quantity) as NumberOfOrders, sum(totalcost) as totalAmountPayed, sum(quantity) as totalItemsBought FROM orders,customer where orders.cust_id = customer.id AND DATEDIFF(NOW(),orderdate)<30 group by customer.id HAVING count(quantity)>4



//TREDNING ORDER

SELECT SHOES.SHOE_ID, BRAND_NAME, SUM(QUANTITY) AS TotalSold FROM `alsohasorderhistory`,SHOES, BRAND WHERE alsohasorderhistory.SHOE_ID= SHOES.SHOE_ID AND BRAND.BRAND_ID = SHOES.BRAND_ID GROUP BY SHOES.SHOE_ID HAVING SUM(QUANTITY) > 2 ORDER BY SUM(QUANTITY) DESC



5. Normalization:

The major point of normalizing tables is to get rid of any redundancies present in the table. We

believe that we have removed any redundant attribute in all our tables. Each table has a unique primary key and only interacts with other tables using a relationship that it is connected to. Primary Key and Foreign Key have been named the same in general. It has been designed towards 3NF. Notably, there are separate tables for relationships. Foreign keys have been used for the 1-to-many tables.

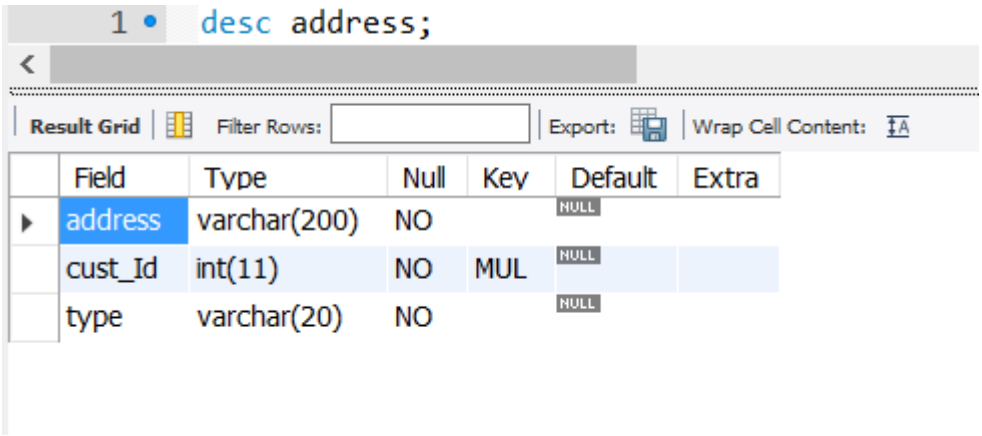
All the tables in our project are in 3rd Normal Form (3NF) since all the tables are in 2NF and every non-key column are mutually independent. All the columns depend directly on the key. None of the columns depend on another column which in turn depends on the key i.e. there is no transitive dependency.

6. Physical Data Base Design:

1. In regards to the physical DB design, appropriate sized integers have been chosen. VARCHAR has often been chosen. The nulls have been reduced wherever possible. Maximum number of primary keys are of CHAR type.

2. All the entities have been translated from the E-R Diagrams to their appropriate tables to serve as our Physical Database Design.

Address Table:



```
1  desc customer
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
dob	date	NO		NULL	
first_name	varchar(30)	NO		NULL	
middle_name	varchar(30)	YES		NULL	
last_name	varchar(30)	NO		NULL	
gender	varchar(6)	NO		NULL	
password	varchar(8)	NO		NULL	
emailId	varchar(40)	NO		NULL	
registeredDate	date	NO		NULL	

Admindefinesfaqcategory Table:

```
1  desc admindefinesfaqcategory;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
category_id	varchar(4)	NO	PRI	NULL	
category_name	varchar(56)	NO		NULL	
admin_Id	char(5)	NO	PRI		

Alsohasorderhistory Table:

```
1  desc Alsohasorderhistory;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
order_Id	int(10)	NO	PRI	NULL	auto_increment
cust_Id	int(11)	NO	MUL	NULL	
orderdate	date	NO		NULL	
shoe_Id	char(5)	NO		NULL	
quantity	int(11)	NO		NULL	
Review	decimal(4,2)	NO		NULL	
totalcost	decimal(6,2)	NO		NULL	

Brand Table:

```
1  desc Brand;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
brand_Id	char(5)	NO	PRI		
brand_Name	varchar(30)	NO		NULL	

Admin Table:

```
1  desc admin;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
admin_Id	char(5)	NO	PRI	NULL	
admin_Pwd	varchar(10)	NO		NULL	

Categories Table:

```
1  desc Categories;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
category_Id	char(5)	NO	PRI		
category_Name	varchar(30)	NO		NULL	

Color Table:

```
1  desc color;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
color_Id	char(5)	NO	PRI		
color_Name	varchar(15)	NO		NULL	

Contains Table:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
shoe_Id	char(5)	NO	PRI		
order_Id	char(5)	NO	PRI		

Faqlist Table:

```
1  desc Faqlist ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
category_id	varchar(4)	NO	PRI	NULL	
faqid	varchar(4)	NO	PRI	NULL	
question	varchar(56)	NO		NULL	
Answer	varchar(456)	NO		NULL	

Gender Table:

```
1  desc gender ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
gender_id	char(10)	NO	PRI	NULL	

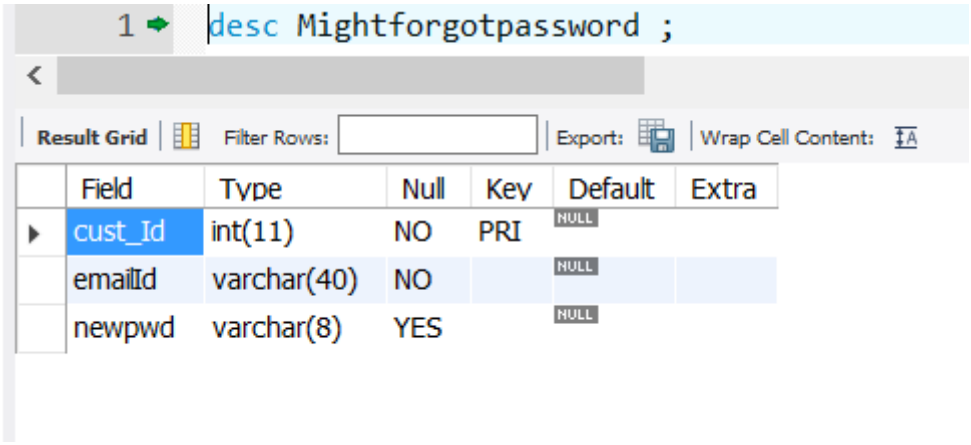
Hascarddetails Table:

```
1  desc Hascarddetails ;
```

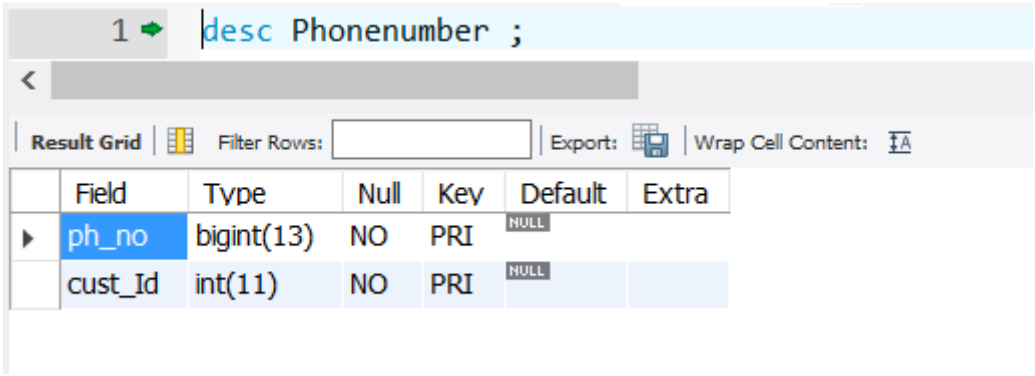
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
cust_Id	int(11)	NO	PRI	NULL	
card_no	bigint(16)	NO	PRI	NULL	
nameoncard	varchar(60)	NO		NULL	
exp_date	date	NO		NULL	

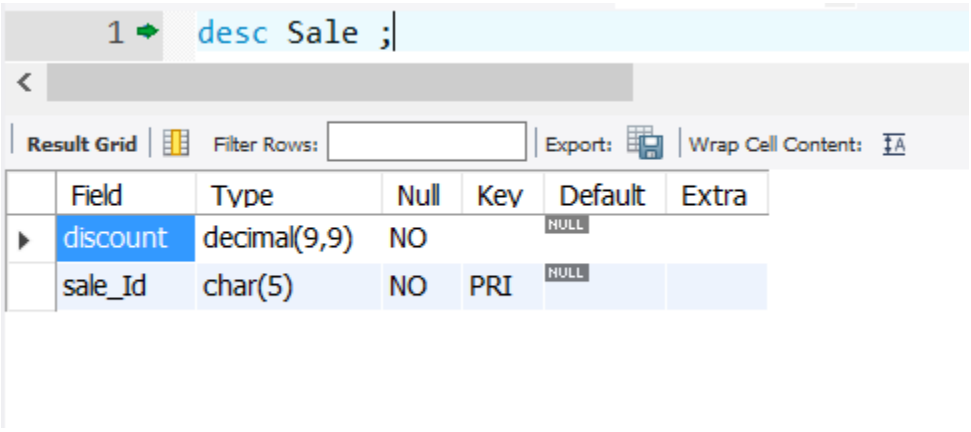
Mightforgotpassword Table:



Phonenumber Table:



Sale Table:



Shoe_gender Table:

```
1  desc Shoe_gender ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
gender_id	char(10)	NO	PRI	NULL	
shoe_id	char(10)	NO	PRI	NULL	

Orders table:

```
1  desc orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
order_Id	char(5)	NO	PRI	NULL	
cust_Id	int(11)	NO	PRI	NULL	
orderdate	date	NO		NULL	
quantity	int(11)	NO		NULL	
Review	decimal(4,2)	NO		NULL	
totalcost	decimal(6,2)	NO		NULL	

Shoes Table:

```
1  desc Shoes ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
admin_Id	char(5)	YES	MUL	NULL	
brand_Id	char(5)	NO	MUL	NULL	
shoe_Id	char(5)	NO	PRI	NULL	
shoe_Img	blob	NO		NULL	
category_Id	char(5)	NO	MUL	NULL	
cost	decimal(6,2)	NO		NULL	

Shoesareofcolor Table:

```
1  desc Shoesareofcolor;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
color_Id	char(5)	NO	PRI	NULL	
shoe_Id	char(5)	NO	PRI	NULL	

Shoesgoondeal Table:

```
1  desc Shoesgoondeal;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
shoe_Id	char(5)	NO	PRI	NULL	
sale_Id	char(5)	NO	PRI	NULL	

Shoeshavesize Table:

```
1  desc Shoeshavesize;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
size_Number	float(5,1)	NO	PRI	NULL	
shoe_Id	char(5)	NO	PRI	NULL	

Shops Table:

```
1  desc Shops;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
shoe_Id	char(5)	NO	PRI	NULL	
cust_Id	int(11)	NO	PRI	NULL	

Size Table:

```
1  desc Size;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
size_Number	float(5,1)	NO	PRI	NULL	
shoe_Id	char(5)	NO	PRI	NULL	

Cart Table:

```
1  desc cart;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

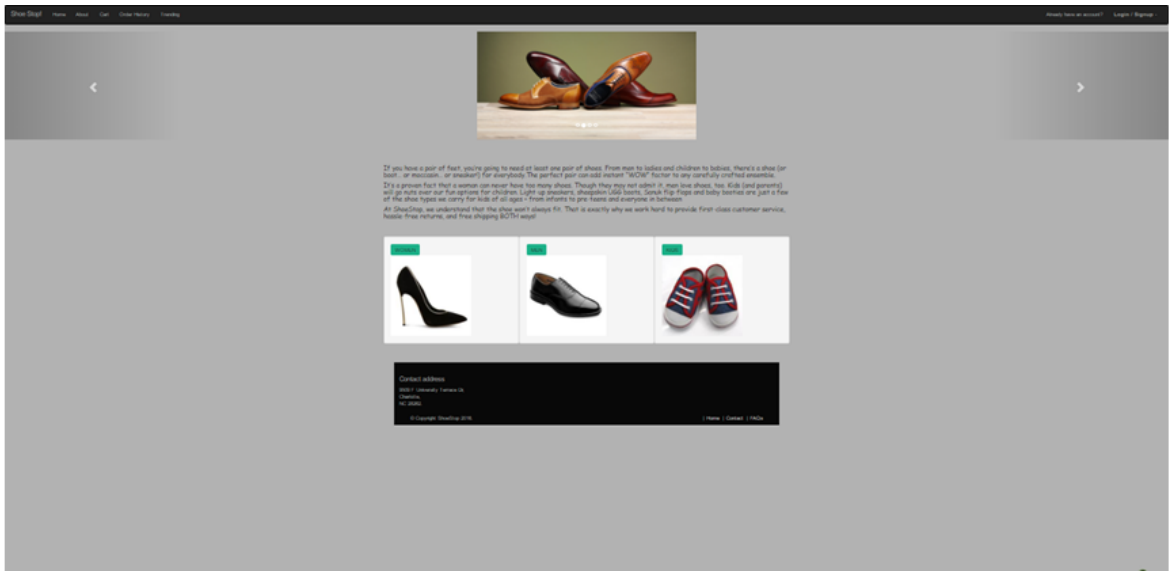
Field	Type	Null	Key	Default	Extra
shoe_Img	blob	NO		NULL	
shoe_Id	char(5)	NO	MUL	NULL	
cust_Id	int(11)	NO	MUL	NULL	
brand_Name	varchar(30)	NO		NULL	
color_Name	varchar(15)	NO		NULL	
size_Number	float(5,1)	NO		NULL	
discount	decimal(9,9)	NO		NULL	
cost	decimal(6,2)	NO		NULL	
cost_after_discount	decimal(9,2)	NO		NULL	

7. GUI DESIGN:

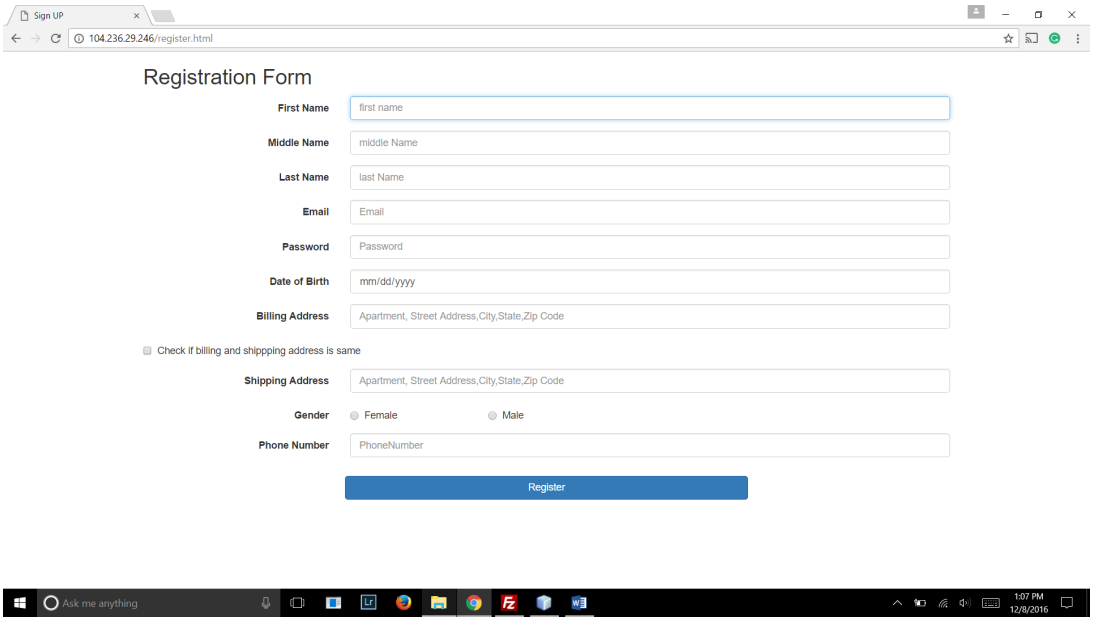
The GUI design for this shoe stop application allows user to register and then login. When the user logs in, he/she can be able to look into different kinds of shoes for men, women and kids and if user likes any shoe, he/she can add it to cart where he/she can checkout later. In cart page, if user wish to buy he/she can go to proceed to checkout or if not interested he/she can cancel the items in the cart.Following snapshots of the application helps to understand better about the website.

Snapshots of the GUI Design are shown below:

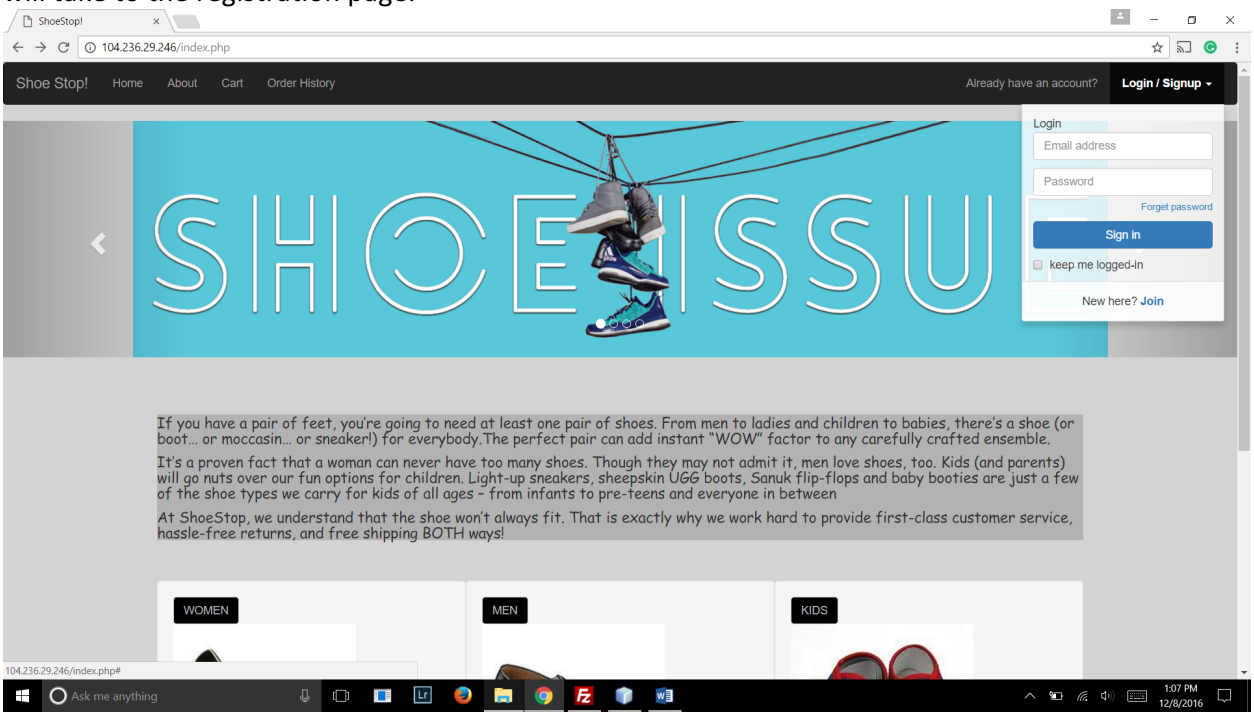
Home screen:



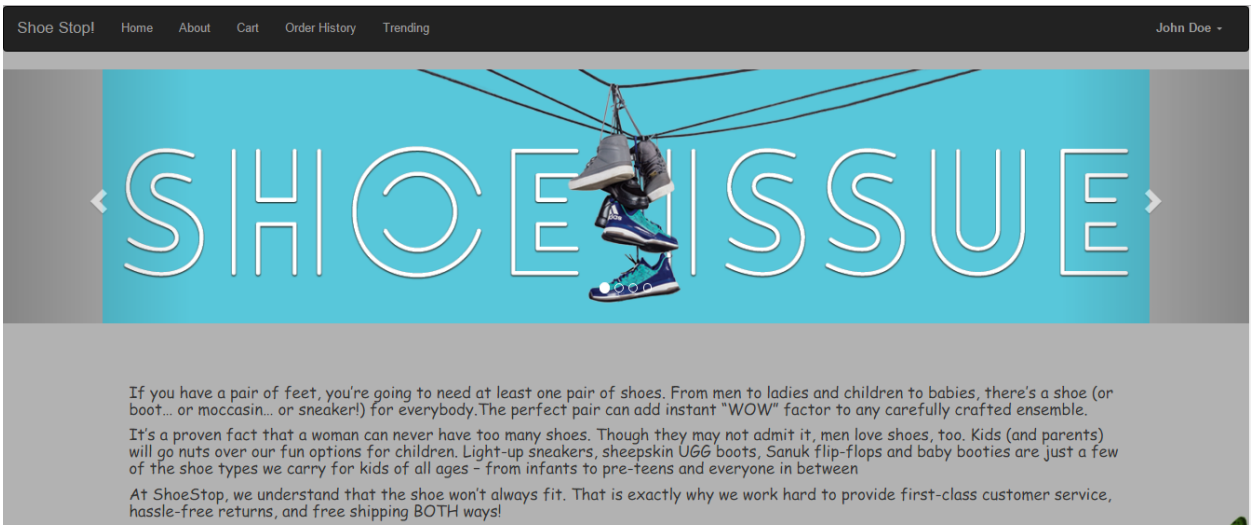
Registration screen: In this page, user able to register by entering first name, last name, email, password, Bate of birth, billing address, gender and phone number.



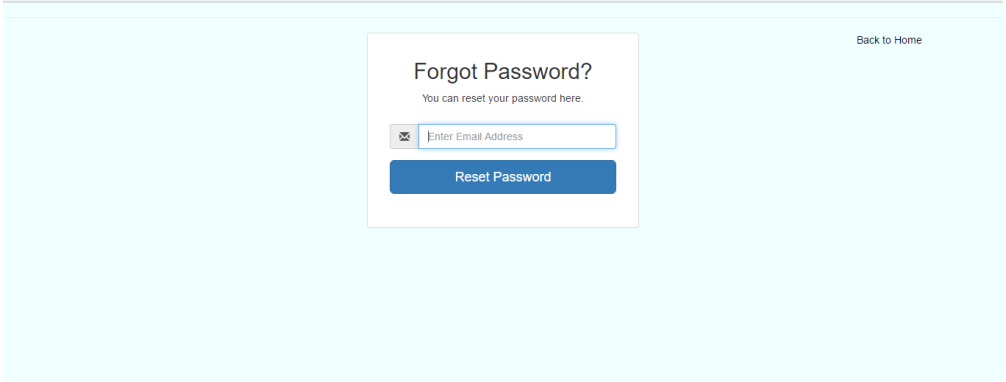
Login Screen: Below page is Login page for our shoes tore website, when user clicks on the "Join" it will take to the registration page.



The "login" button would take you to the welcome page below:

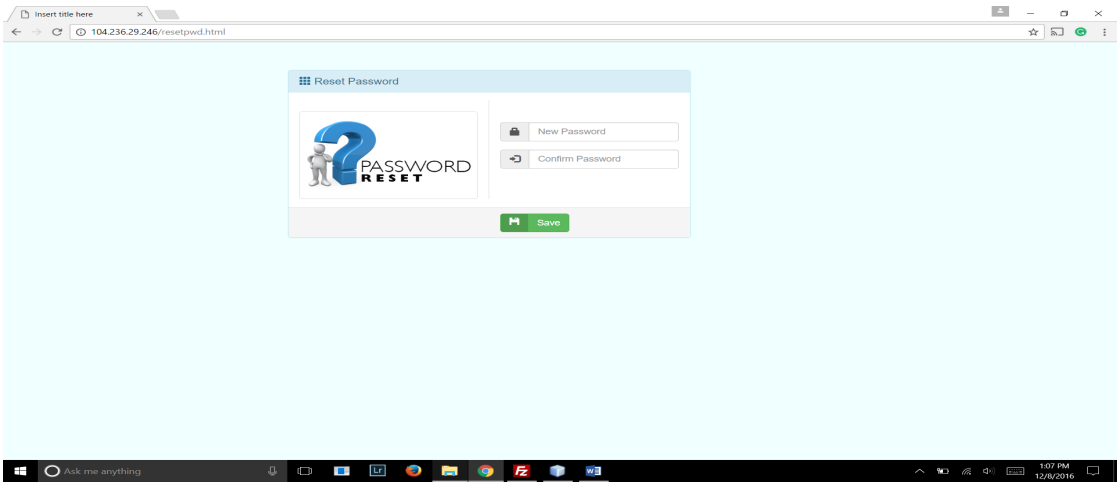


Forgot Password Screen:

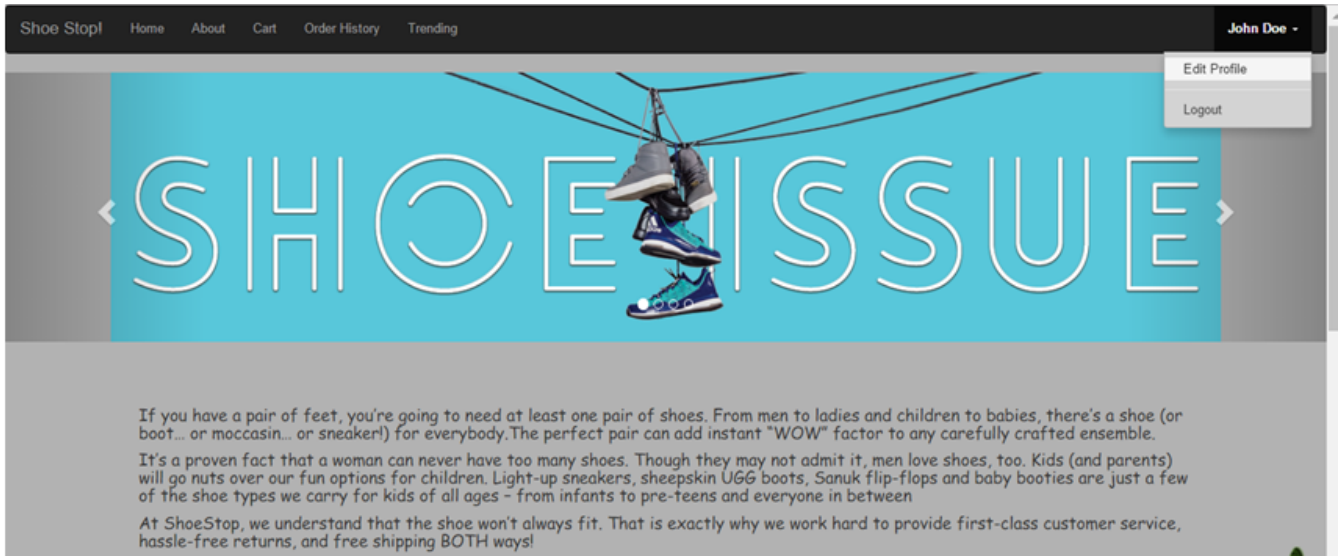


When user enters email id in above screen and clicks on "Reset Password" button,it will redirect to reset password screen.

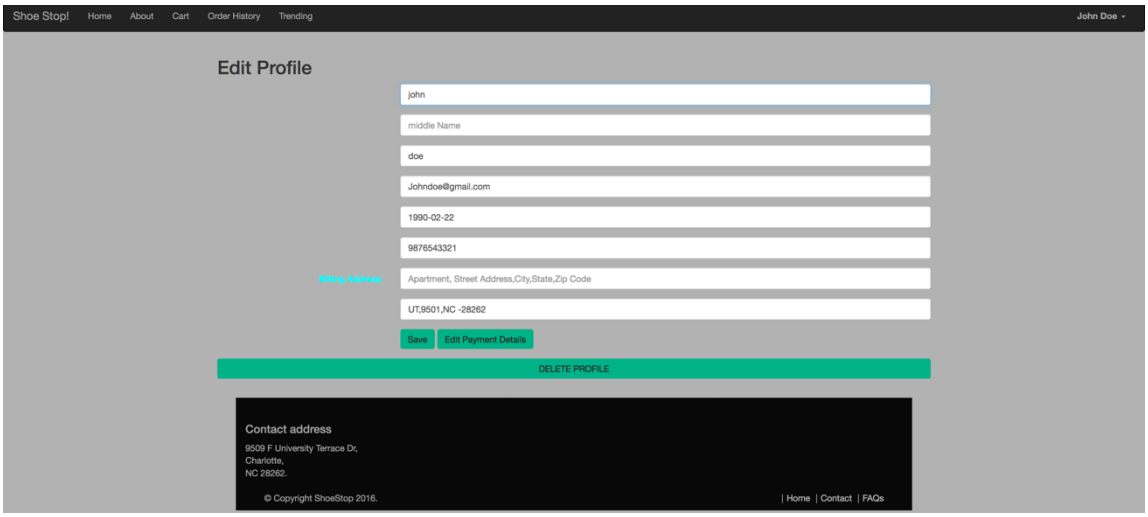
Reset Password screen: In this page user can reset his/her password.



Edit user Details screen: In this page user can edit his personal details.

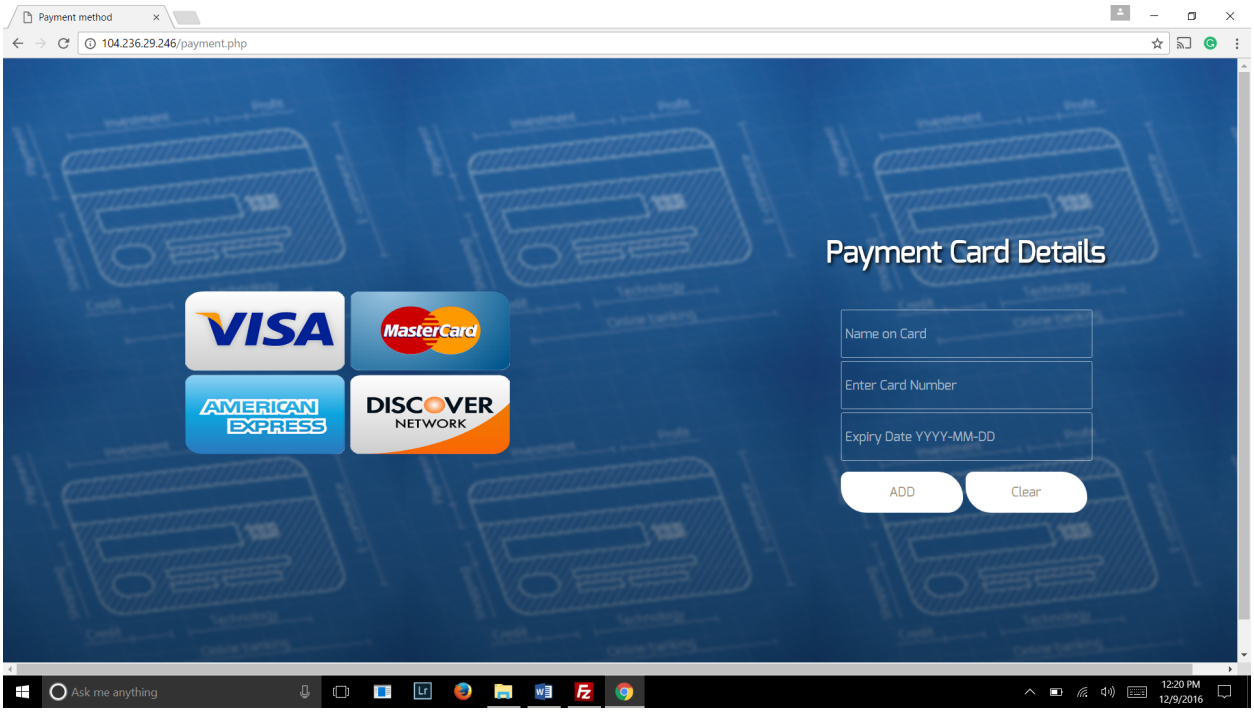


When user clicks on "Edit Profile" button, it will redirect to the below page.

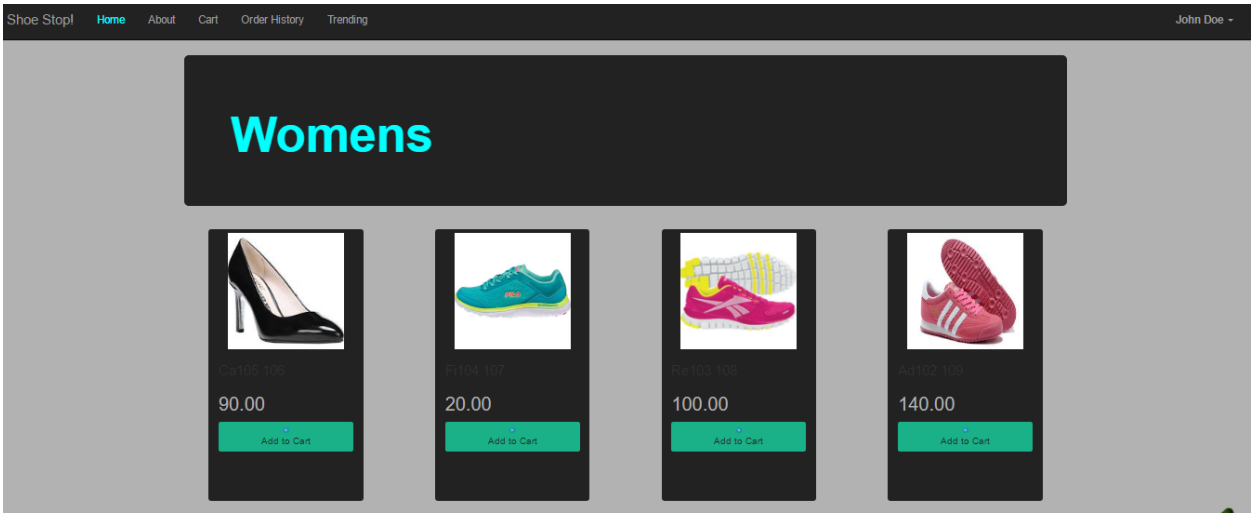


Enter Payment Details screen:

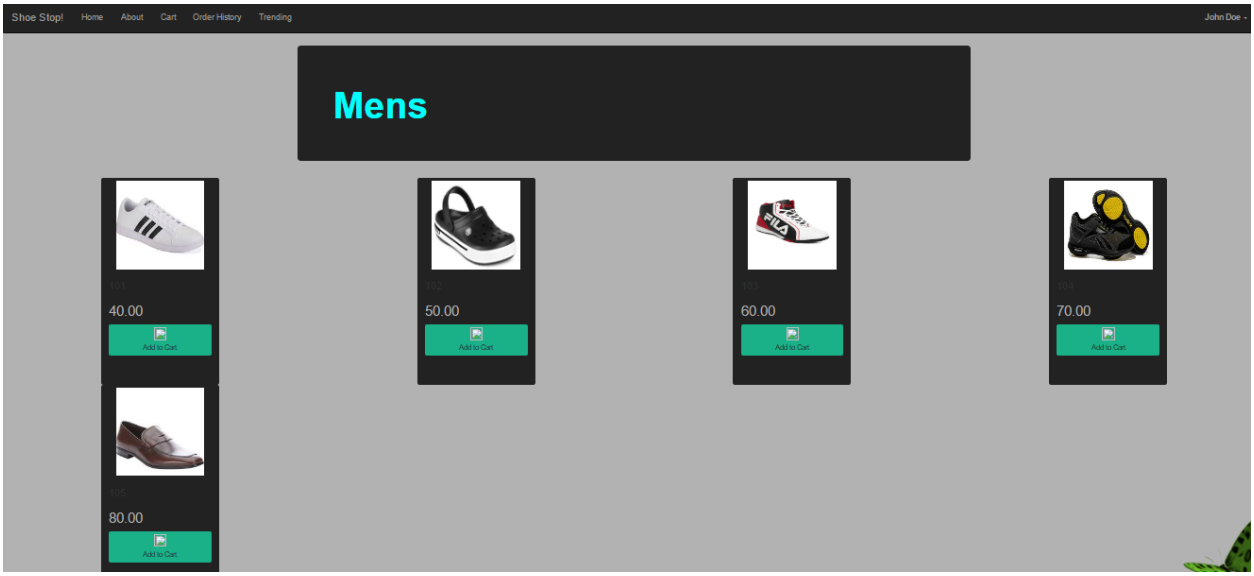
When user clicks on “EditPaymentDetails” button in Edit details page, it will redirect to the below page.



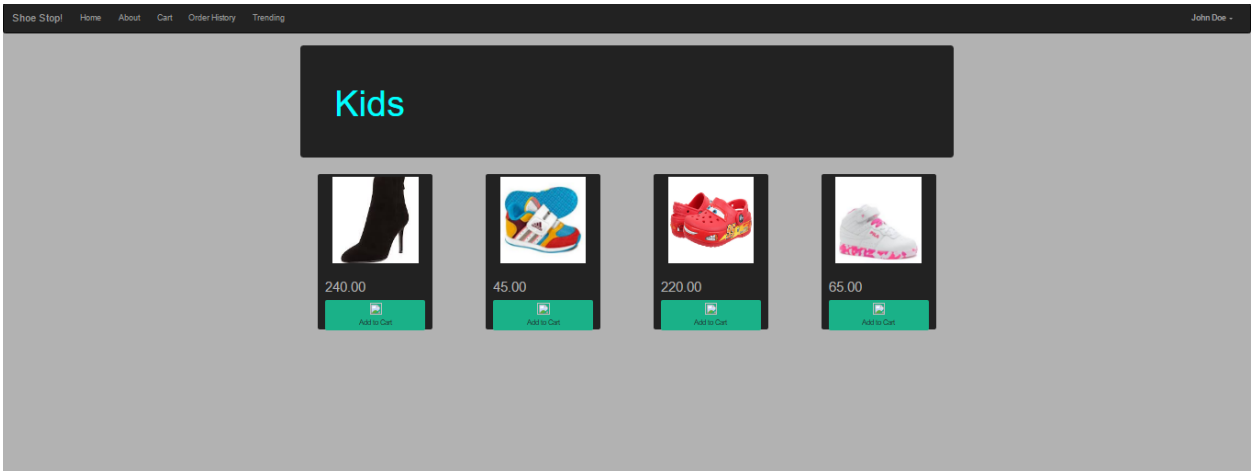
Womens Shoes screen: When user click on “Women” button, it will redirect to below page.



Mens Shoes screen: When user click on “Men” button, it will redirect to below page.

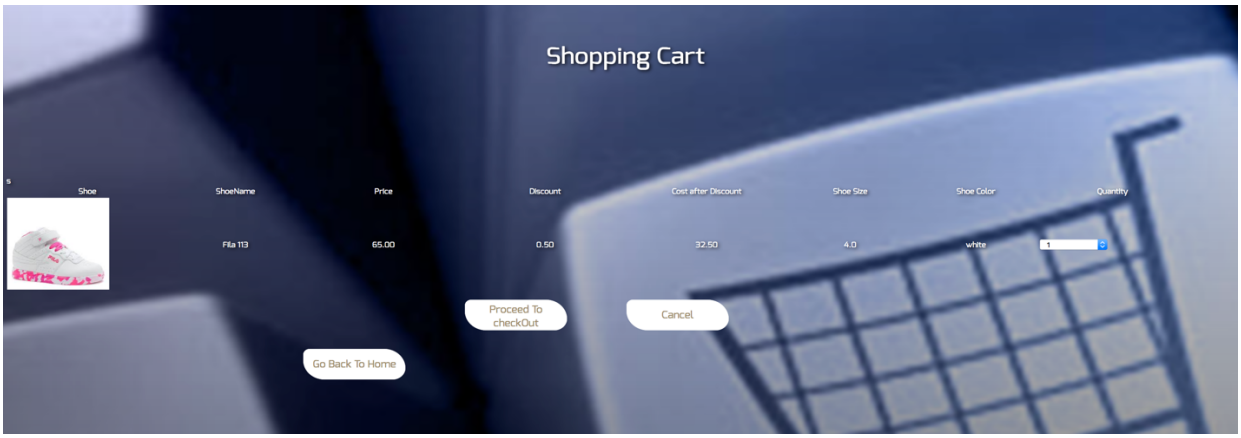


Kids Shoes screen: When user click on “Kids” button, it will redirect to below page.

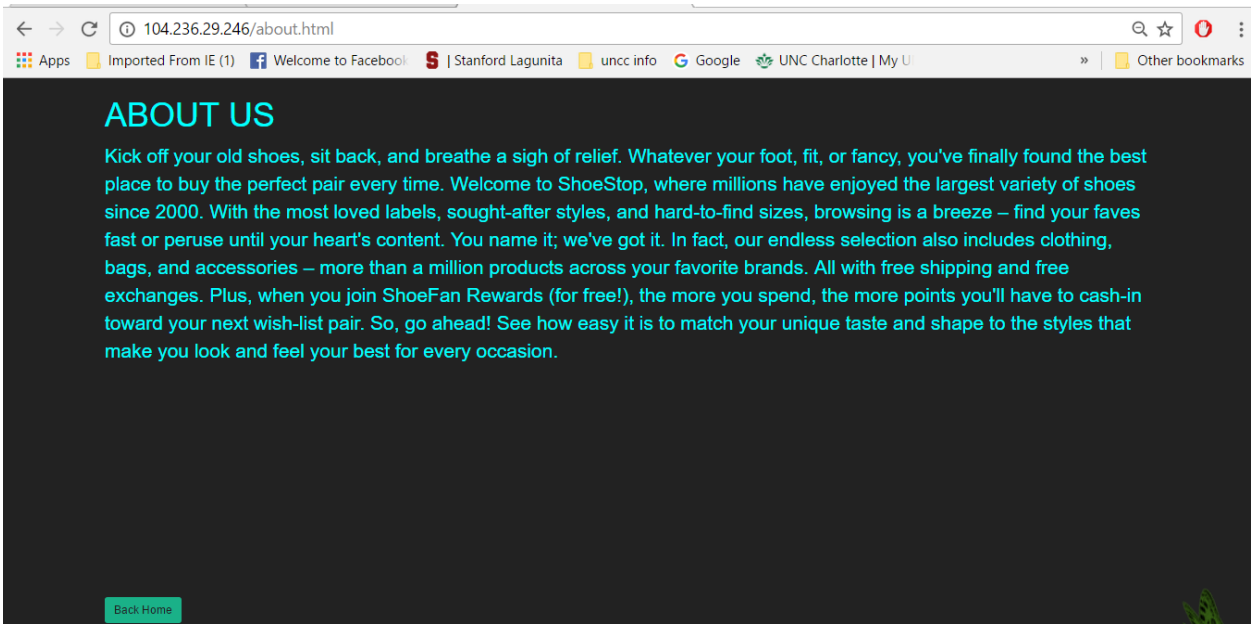


Order History Screen: When user click on “Order History” button, it will redirect to below page.

Order History Screen: When user click on “Order History” button, it will redirect to below page.

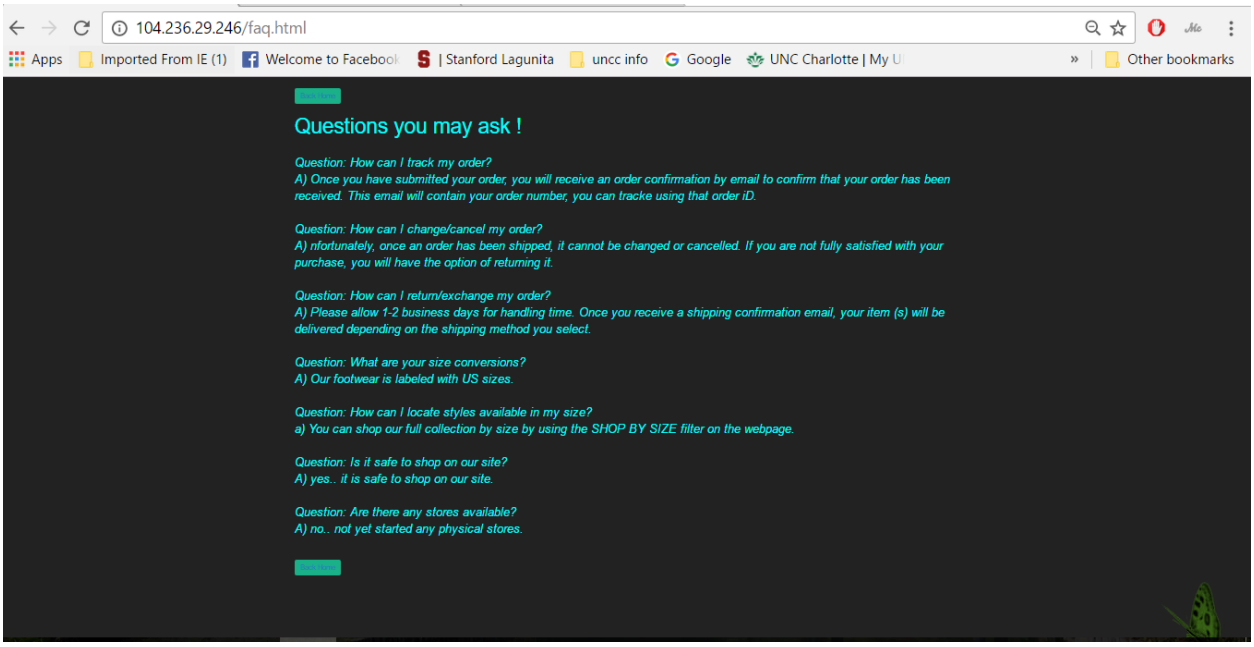


About Screen: When user clicks on About link, it will redirect to below page.

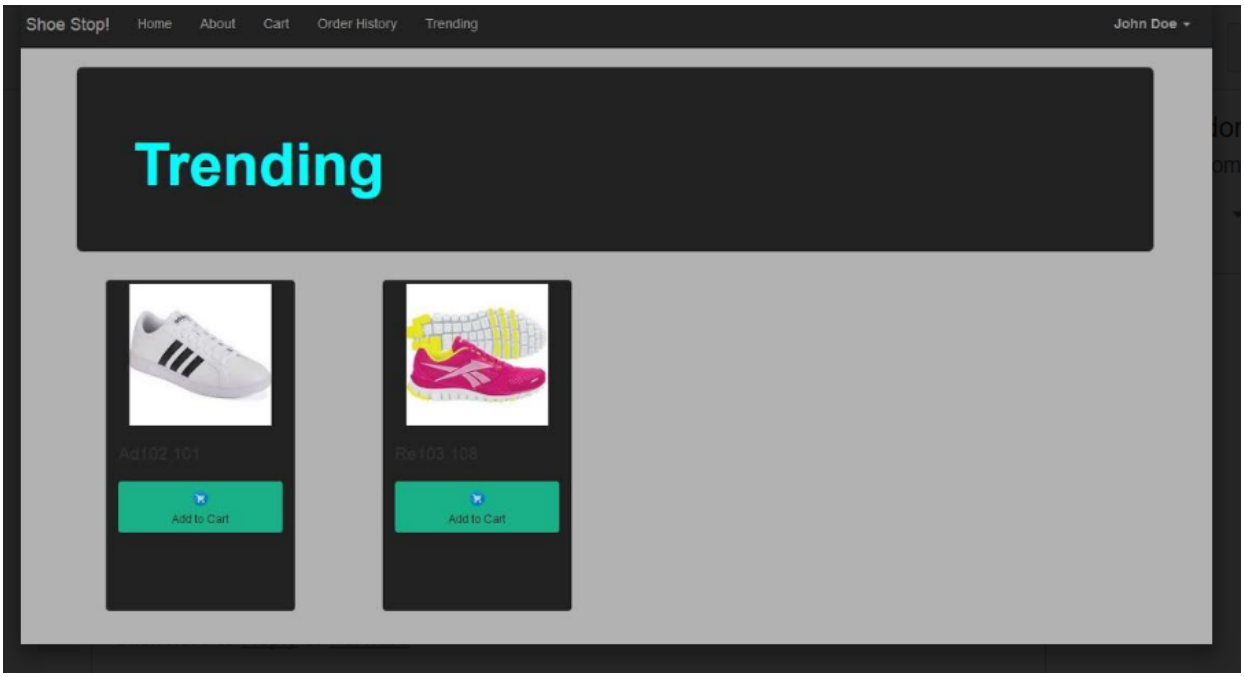


FrequentlyAskedQuestions Screen:

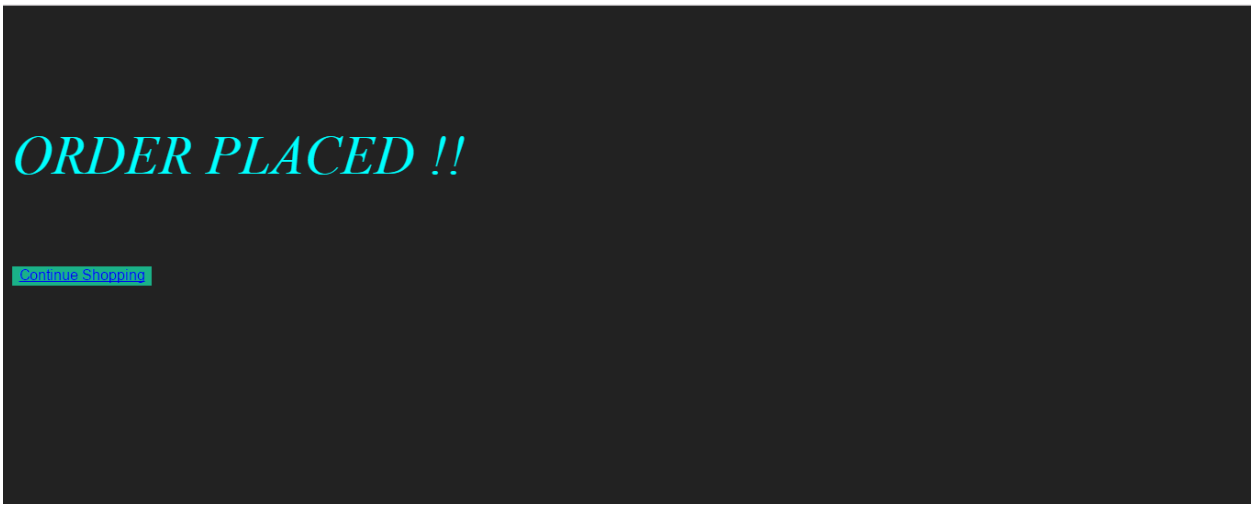
When user click on “FAQs” at the bottom of home page, it will redirect to below page.



Trending Screen: When user clicks on “Trending” tab in home screen, it will redirect to below screen.



Success Screen: In cart page, when user clicks on proceed to checkout button, then it will redirect to success page and again if user clicks on “continue Shopping” link, it will redirect to home screen.



8. Conclusion and Future work:

Our project successful implemented the concepts of database design. The future of shopping lies in the e-commerce. That's where our website steps into picture. With the personalization and its vast collection of shoes, we think it is the future of shoes shopping. Also, our project can be further developed by analyzing the type of shoes customers are more interested in. We can process the data to emerge with patterns which help us understand what the customers are actually liking and cater to those needs. We also need to implement security measure to prevent attacks along with filter & search options.

9. Appendices

9.1 Project Status

S.NO	DATE	PROJECT STATUS
1	9/4/2016	Project Topic discussion
2	9/9/2016	Project Plan Finalization
3	9/14/2016	Discussed on the requirements, walkthrough of the project to the team members
4	9/17/2016	Prepared sample ER Discussed new ideas and few business functions
5	9/23/2016	Project Report Part 1- Final
6	10/4/2016	Database Table creations
7	10/13/2016	SQL Query for Business function
8	10/27/2016	Project Report Part 2 final
9	11/15/2016	GUI Design
10	11/22/2016	Business Functionalities
11	12/06/2016	Business Functionalities
12	12/10/2016	Final report documentation

9.2 Project meetings

S.NO	DATE	AGENDA	LOCATION	TYPE	TIME
1	9/4/2016	Project Topic discussion	Atkins Library	Face to Face/Hangout	2:00 PM-5:00 PM
2	9/9/2016	Project Plan Finalization	Atkins Library	Face to Face	2:00 PM-5:00 PM
3	9/14/2016	Discussed on the requirements, walkthrough of the project to the team members	Atkins Library	Face to Face	4:00PM-6:00PM
4	9/17/2016	Prepared sample ER Discussed new ideas and few business functions	Atkins Library	Face to Face	10:00AM-1:30PM
5	9/23/2016	Project Report Part 1- Final	Atkins Library	Face to Face	2:30PM - 6:30 PM
6	10/4/2016	Database Table creations	Atkins Library	Face to Face	2PM – 6PM
7	10/13/2016	SQL Query for Business function	Atkins Library	Face to Face	1PM – 6PM
8	10/27/2016	Project Report Part 2 final	Atkins Library	Face to Face	2PM – 6PM
9	11/15/2016	GUI Design	Atkins Library	Face to Face	11AM- 2PM
10	11/22/2016	Business Functionalities	Atkins Library	Face to Face	1PM- 6PM
11	12/06/2016	Business Functionalities	Atkins Library	Face to Face	11AM-7PM

12	12/10/2016	Final report documentation	Atkins Library	Face to Face	11AM-9PM

9.3 Team Members:

Name	Niner ID	Email-id	Phone number
Puneeth Devabhaktuni	800953156	pdevabha@uncc.edu	330-212-0534
Rahul Reddy Arva	800955965	rarva@uncc.edu	980-636-2196
Vinusha Bobburu	800969202	vbobburu@uncc.edu	716-472-9454
Lily Naoudom	800648084	pnaoudom@uncc.edu	980-579-7601
Justin Toler	800578466	Jtoler5@uncc.edu	980-229-5194
Nithya Gummadi	800960708	sgummad2@uncc.edu	980-777-5671
Akshay Kadari	800954104	akadari@uncc.edu	919-961-1194
Himaja Choudary	800907552	hmuppall@uncc.edu	410-790-8261
Aditya Viswanadha	800959537	aviswan2@uncc.edu	848-252-9616