Database Management System Project

B. Tech ICT
(4th Semester)

Project Title : **Online Shopping System**

Under the Guidance of **Prof. Shefali Naik**

Group Members

Parth N Patel	(AU1841028)
Shreyansh Shah	(AU1841046)
Hardi Kadia	(AU1841059)

School of Engineering and Applied Science
Ahmedabad University

Description:

Online shopping is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser. Here we have developed an Online Shopping System with the help of Java for Frontend and Oracle Database Management System in the Backend.

Online Shopping System provides an user friendly Interactive system from which users can buy goods. This System allows the user to buy different kinds of goods from the System and it will maintain a history record for that user.

In this Shopping System the Admin of the System can add different Categories in the System and he/she can also add different products under that particular category.

Main Functionalities of the Online Shopping System are :

- Functionalities for the Admin
  - Admin can sign in to the System with his Admin ID
  - Admin can Add New Categories in the System
  - Admin can Add Product corresponding to that Category and if that Category is not Present then first that Category is created and then that Product is added.
  - Admin can see the data of all the Registered Customer of the System
  - Admin can see the Order placed by all the Customers
  - Admin can see the Order placed by a Specific Customer

- Functionalities for the User
  - User need to Register to the System by entering his personal details

- Users can Sign In to the System in order to place an order from the System.
- User can select category his favourite from the list of categories provided to him/her
- After Selecting the category, User can see all the available products of that Category Available to him
- Now User Can Select any product from the given list and he needs to give the quantity of the product he wants to buy.
- Now User will see all the Products added by him in the Cart menu and he will also see the total amount to pay for all the products he has selected.
- Here User can clear his whole Shopping Cart or he can Update the Quantities of the Product in the Shopping Cart.
- Now when the user will place an order for the selected products then he needs to fill details where the order is to be placed.
- After Adding the Details User can choose his Favourable Method for Payment of his order.
- After Placing the Order the Shopping Cart of the User will be emptied automatically.
- Users can also see his order history of the system i.e how many orders have been placed from this System.

Triggers are use to check the password satisfies required conditions(first character should be a special character and minimum length should be 8) and whether the phone no. is 10 digit or not. It is also used while adding to the shopping cart whether the entered no. of quantity is available or not. Also when the order is placed the no. of quantity of that product is subtracted from the database which is purchased by the customer.

Entity-Relationship Diagram:

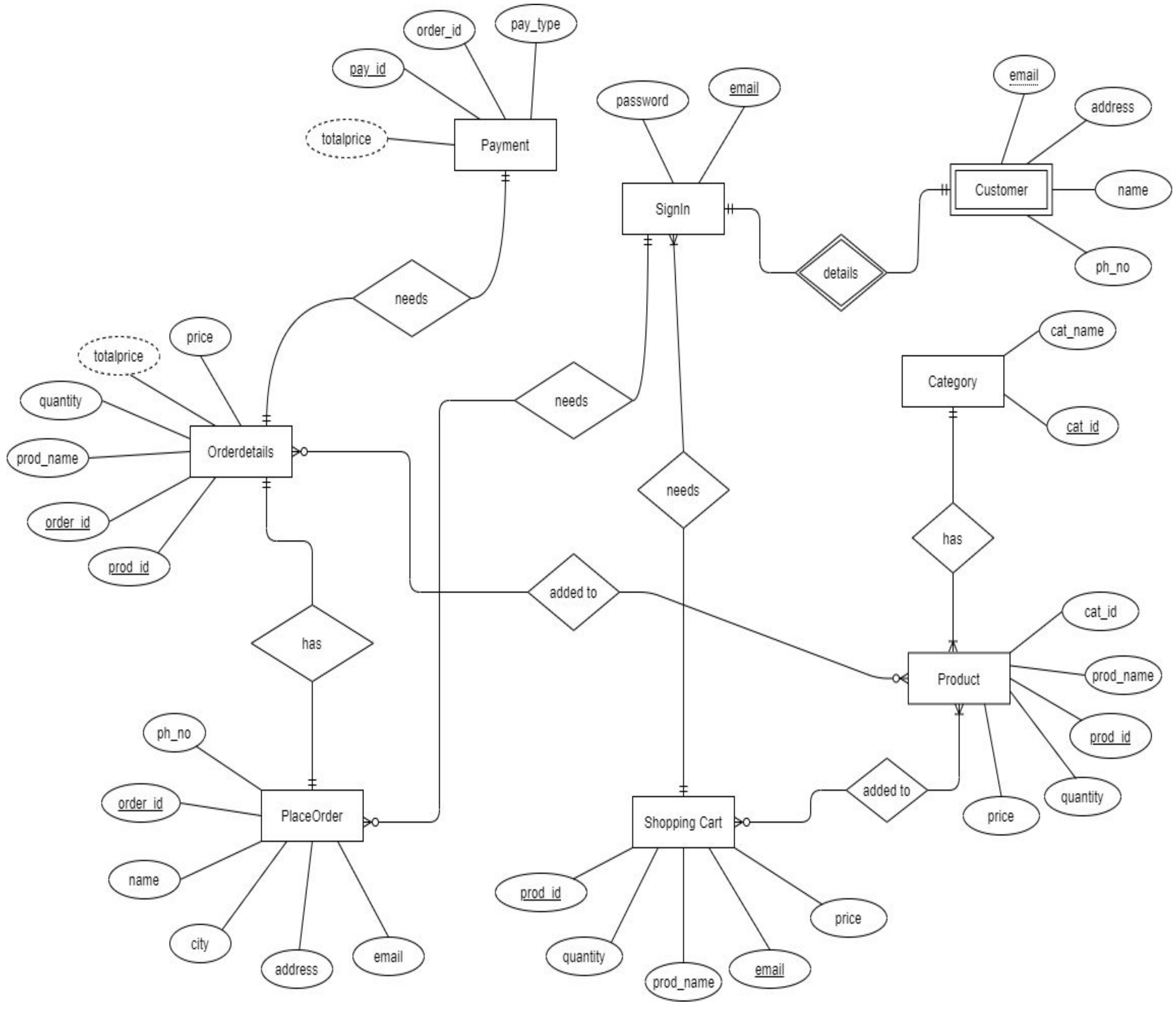


Table Design(Data Dictionary)**:**

Table → SignIn

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
EMAIL	VARCHAR2	50	PRIMARY KEY	Email for Login (Mandatory)
PASSWORD	VARCHAR2	20	NOT NULL	Password for Login (Mandatory)

Table → Customer

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
EMAIL	VARCHAR2	50	FORIEGN KEY, NOT NULL	Customer Email for profile (Mandatory)
NAME	VARCHAR2	50	NOT NULL	Customer name for profile
PH_NO	VARCHAR2	10	NOT NULL	Customer ph no. For profile

ADDRESS	VARCHAR2	100	NOT NULL	Customer address for profile

Table → Category

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
CAT_ID	INT		PRIMARY KEY	Uniquely identify Category
CAT_NAME	VARCHAR2	50	UNIQUE, NOT NULL	Category Name

Table → Product

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
PROD_ID	INT		PRIMARY KEY	Uniquely identify Products
CAT_ID	INT		FORIEGN KEY, NOT NULL	Category Id of Product to identify which category it belongs
PROD_NAME	VARCHAR2	50	UNIQUE, NOT NULL	Product Names
PRICE	INT		NOT NULL	Price of product added

QUANTITY	INT		NOT NULL	Quantity of Product Available

Table → ShoppingCart

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
EMAIL	VARCHAR2	50	PRIMARY KEY,FOREIGN KEY	Email of customer adding products to cart
PROD_ID	INT		PRIMARY KEY,FOREIGN KEY	Product ID of product added to cart
PROD_NAME	VARCHAR2	50	NOT NULL	Product name of product added to cart
PRICE	INT		NOT NULL	Price of product added to cart
QUANTITY	INT		NOT NULL	Quantity of product added to cart

Table → PlaceOrder

Attribute	Data Type	Size	Constraint	Description
ORDER_ID	INT		PRIMARY KEY	Uniquely identify orders
EMAIL	VARCHAR2	50	FOREIGN KEY,NOTNULL	Email of customer to whom order belongs to
NAME	VARCHAR2	50	NOT NULL	Name of customer
CITY	VARCHAR2	10	NOT NULL	City in which order has to be deliver
ADDRESS	VARCHAR2	50	NOT NULL	Address on which order has to be deliver
PH_NO	VARCHAR2	10	NOT NULL	Ph_no of customer to contact during emergency

Table → OrderDetails

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
ORDER_ID	INT		PRIMARY KEY,FOREIGN KEY	Uniquely identify orders which is placed
PROD_ID	INT		PRIMARY KEY,FOREIGN KEY	Uniquely identify products that has to order
PROD_NAME	VARCHAR2	50	NOT NULL	Product name which is ordered
EMAIL	VARCHAR2	50	NOT NULL	Email of customer to whom order belongs to
PRICE	INT		NOT NULL	Price of product which is ordered
QUANTITY	INT		NOT NULL	Quantity of ordered products
TOTALPRICE	INT		NOT NULL	Total amount to be paid by customer after placing order

Table → Payment

Attribute	**Data Type**	**Size**	**Constraint**	**Description**
PAY_ID	INT		PRIMARY KEY	Uniquely identify payment
ORDER_ID	INT		FORIEGN KEY, NOT NULL	Uniquely identify order belongs to whom
PAY_TYPE	VARCHAR2	50	NOT NULL	Mode of payment
TOTALPRICE	INT		NOT NULL	Total amount has to be paid after order is places

Procedures:

Procedure 1:

```
CREATE OR REPLACE PROCEDURE create_user(mail IN
signin.email%TYPE,pass IN signin.password%TYPE, result OUT INT)
AS
BEGIN
    result:=0;
    INSERT INTO signin(email,password) VALUES(mail,pass);
    COMMIT;
    result := 1;
    exception
    WHEN others THEN
    result := 0;
    ROLLBACK;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
create_user(?,?,?)}");
```

Procedure 2:

```
CREATE OR REPLACE PROCEDURE add_user(mail IN
customer.email%TYPE,uname IN customer.name%TYPE,phone IN
customer.ph_no%TYPE,add IN customer.address%TYPE, result OUT
INT) AS
BEGIN
    result:=0
    INSERT INTO customer(email,name,ph_no,address)
VALUES(mail,uname,phone,add);
```

```
    COMMIT;
    result := 1;
    exception
    WHEN others THEN
    result := 0;
    ROLLBACK;
END;
/
CallableStatement ps1 = myConnect.prepareCall("{call
add_user(?,?,?,?,?)}");
```

Procedure 3:

```
CREATE OR REPLACE PROCEDURE check_user(mail IN
signin.email%TYPE,pass IN signin.password%TYPE, result OUT
VARCHAR2) AS
BEGIN
    SELECT email INTO result FROM signin WHERE email=mail AND
password=pass;
    exception
    WHEN NO_DATA_FOUND THEN
    result := 'USER ID AND PASSWORD INCORRECT';
END;
CallableStatement stmt = myConnect.prepareCall("{call
check_user(?,?,?)}");
```

Procedure 4:

```
CREATE OR REPLACE PROCEDURE get_details(mail IN
signin.email%TYPE,uname OUT VARCHAR2,phone OUT VARCHAR2,adr OUT
VARCHAR2,pwd OUT VARCHAR2,result out int) AS
BEGIN
```

```
    result:=0
    SELECT name,ph_no,address INTO uname,phone,adr FROM customer
WHERE email=mail;
    SELECT password INTO pwd FROM signin WHERE email=mail;
    result:=1;
    exception
    WHEN others THEN
    result:=0;
    ROLLBACK;
END;
/
CallableStatement st = myConnect.prepareCall("{call
get_details(?,?,?,?,?)");
```

Procedure 5:

```
CREATE OR REPLACE PROCEDURE update_details(mail IN
signin.email%TYPE, uname IN customer.name%TYPE, phone IN
customer.ph_no%TYPE, adr IN customer.address%TYPE, pwd IN
signin.password%TYPE, result OUT INT) AS
BEGIN
    result:=0;
    UPDATE customer SET name=uname, ph_no=phone,address=adr
WHERE email=mail;
    UPDATE signin SET password=pwd WHERE email=mail;
    result:=1;
    exception
    WHEN others THEN
    result:=0;
    ROLLBACK;
END;
/
CallableStatement st = myConnect.prepareCall("{call
update_details(?,?,?,?,?,?)");
```

Procedure 6:

```
CREATE OR REPLACE PROCEDURE add_product (
        prod_name       IN   product.prod_name%TYPE,
        price           IN   product.price%TYPE,
        quantity        IN   product.quantity%TYPE,
        catt_name       IN   category.cat_name%TYPE,
        result OUT INT
    ) AS
        cursor c_name IS SELECT cat_id FROM category WHERE
cat_name = catt_name;
        r_name c_name%ROWTYPE;
    BEGIN
        OPEN c_name;
        LOOP
            FETCH c_name INTO r_name;
            IF c_name%NOTfound THEN
                        INSERT INTO category
VALUES(dept_seq.nextval,catt_name);
                    ELSE
        INSERT INTO
product(prod_id,cat_id,prod_name,price,quantity) VALUES (
            prod_seq.nextval,
            r_name.cat_id,
            prod_name,
            price,
        quantity
            );
    result := 1;
    END IF;
    exit;
    END LOOP;
    CLOSE c_name;
    COMMIT;
    exception
    WHEN others THEN
    result := 0;
    ROLLBACK;
END add_product;
/
CallableStatement ps = myConnect.prepareCall("{call
add_product(?,?,?,?,?)}");
```

Procedure 7:

```
create or replace procedure add_cart(prodname IN
product.prod_name%TYPE,email IN signin.email%TYPE,quantity IN
product.quantity%TYPE,result out int) as
cursor c_prodid is select * from product where
prod_name=prodname;
r_prodid c_prodid%rowtype;
begin
result:=0
open c_prodid;
    loop
    fetch c_prodid into r_prodid;
    if c_prodid%notfound then
        result:=0;
    exit;
    else
      insert into
shoppingcart(email,prod_id,prod_name,price,quantity)
values(email,r_prodid.prod_id,prodname,r_prodid.price*quantity,q
uantity);
```

```
      result:=1;
      exit;
    end if;
end loop;
close c_prodid;
exception
when others then
result:=0;
end;
/
CallableStatement ps = myConnect.prepareCall("{call
add_cart(?,?,?,?)}");
```

Procedure 8:

```
CREATE OR REPLACE PROCEDURE resetcart(emailin IN
signin.email%TYPE,result OUT INT) AS
BEGIN
    DELETE FROM shoppingcart WHERE email=emailin;
    result:=1;
    exception
    WHEN others THEN
    result:=0;
    ROLLBACK;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
resetcart(?,?)}");
```

Procedure 9:

```
CREATE OR REPLACE PROCEDURE getcart_details(emailin IN
shoppingcart.email%TYPE, cart_details OUT SYS_REFCURSOR) AS
BEGIN
    OPEN cart_details for SELECT prod_name,price,quantity FROM
shoppingcart WHERE email=emailin;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
getcart_details(?,?)}");
```

Procedure 10:

```
CREATE OR REPLACE PROCEDURE getproduct_details(catname IN
category.cat_name%TYPE, prod_details OUT SYS_REFCURSOR) AS
BEGIN
    OPEN prod_details for SELECT prod_name,price FROM product
WHERE cat_id=(SELECT cat_id FROM category WHERE
cat_name=catname);
END;
/
CallableStatement ps = myConnect.prepareCall("{call
getproduct_details(?,?)}");
```

Procedure 11:

```
CREATE OR REPLACE PROCEDURE getuser_details(user_details OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN user_details for SELECT * FROM signin join customer ON
signin.email=customer.email;
END;
/
```

```
CallableStatement ps = myConnect.prepareCall("{call
getuser_details(?)}");
```

Procedure 12:

```
CREATE OR REPLACE PROCEDURE addyourorder(orderid IN INT,emailid
IN signin.email%TYPE, tprice IN INT, result OUT INT) AS
cursor c_addorder IS SELECT * FROM shoppingcart WHERE email =
emailid;
r_addorder c_addorder%ROWTYPE;
BEGIN
result:=0;
OPEN c_addorder;
LOOP
  FETCH c_addorder INTO r_addorder;
    IF c_addorder%NOTfound THEN
    exit;
    ELSE
      INSERT INTO
orderdetails(order_id,prod_id,email,prod_name,price,quantity,tot
alprice);
VALUES(orderid,r_addorder.prod_id,emailid,r_addorder.prod_name,r
_addorder.price,r_addorder.quantity,tprice);
            result:=1;
    END IF;
END LOOP;
CLOSE c_addorder;
exception
WHEN others THEN
result:=0;
```

```
ROLLBACK;
END;
/
CallableStatement ps2 = myConnect.prepareCall("{call
addyourorder(?,?,?,?)}");
```

Procedure 13:

```
CREATE OR REPLACE PROCEDURE orderplace(orderid IN INT,emailid IN
placeorder.email%TYPE,add IN placeorder.address%TYPE,phnnumber
IN placeorder.ph_no%TYPE,name IN customer.name%TYPE, city IN
VARCHAR2, result OUT INT) AS
BEGIN
  result:=0;
  INSERT INTO placeorder
VALUES(orderid,emailid,name,city,add,phnnumber);
    result:=1;
    exception
    WHEN others THEN
    result:=0;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
orderplace(?,?,?,?,?,?,?)}");
```

Procedure 14:

```
CREATE OR REPLACE PROCEDURE getorder_id (mail IN
signin.email%TYPE,ord_id OUT sys_refcursor) AS
```

```
BEGIN
OPEN ord_id for SELECT order_id FROM placeorder WHERE
email=mail;
END;
CallableStatement ps1 = myConnect.prepareCall("{call
getorder_id(?,?)}");
```

Procedure 15:

```
CREATE OR REPLACE PROCEDURE disorder (mail IN
signin.email%TYPE,cnt OUT INT) AS
BEGIN
SELECT DISTINCT COUNT(order_id) INTO cnt FROM placeorder WHERE
email=mail;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
disorder(?,?)}");
```

Procedure 16:

```
CREATE OR REPLACE PROCEDURE getinvoice(ordid IN INT,get_inv OUT
sys_refcursor) AS
BEGIN
OPEN get_inv for SELECT * FROM orderdetails WHERE
order_id=ordid;
END;
/
CallableStatement ps1 = myConnect.prepareCall("{call
getinvoice(?,?)}");
```

Procedure 17:

```
CREATE OR REPLACE PROCEDURE ordidcnt(ordid IN INT,cnt OUT INT)
AS
BEGIN
SELECT COUNT(order_id) INTO cnt FROM orderdetails WHERE
order_id=ordid;
END;
/
CallableStatement ps1 = myConnect.prepareCall("{call
ordidcnt(?,?)}");
```

Procedure 18:

```
CREATE OR REPLACE PROCEDURE getallorders(allorder OUT
sys_refcursor) AS
BEGIN
OPEN allorder for SELECT * FROM orderdetails;
END;
/
CallableStatement ps = myConnect.prepareCall("{call
getallorders(?)}");
```

Procedure 19:

```
CREATE OR REPLACE PROCEDURE getuserorders(mail IN
signin.email%TYPE,allorder OUT sys_refcursor) AS
BEGIN
OPEN allorder for SELECT * FROM orderdetails WHERE email=mail;
END;
CallableStatement ps = myConnect.prepareCall("{call
getuserorders(?,?)}");
```

Procedure 20:

```
CREATE OR REPLACE PROCEDURE add_pay(ord_id IN INT,paytype IN
VARCHAR,tprice IN INT, result OUT INT) AS
BEGIN
INSERT INTO payment(pay_id,order_id,pay_type,totalprice)
VALUES(pay_seq.nextval,ord_id,paytype,tprice);
END;
/
CallableStatement ps3 = myConnect.prepareCall("{call
add_pay(?,?,?,?)}");
```

Procedure 21:

```
CREATE OR REPLACE PROCEDURE getpaydetails(ord_id IN INT,paytype
OUT VARCHAR, tprice OUT INT) AS
BEGIN
    SELECT pay_type,totalprice INTO paytype,tprice FROM payment
WHERE order_id=ord_id;
END;
/
CallableStatement ps = myConnect.prepareCall("call
getpaydetails(?,?,?)");
```

Triggers:

Trigger 1:To Auto Increment Category_Id

```
CREATE SEQUENCE dept_seq START WITH 1
     INCREMENT BY 1
     CACHE 100;
```

```
CREATE OR REPLACE TRIGGER cat_bir
BEFORE INSERT ON category
FOR EACH ROW
BEGIN
  SELECT dept_seq.NEXTVAL
  INTO   :new.cat_id
  FROM   dual;
END;
/
```

Trigger 2:To Auto Increment Product_Id

```
CREATE SEQUENCE prod_seq START WITH 1
     INCREMENT BY 1 CACHE 100;

CREATE OR REPLACE TRIGGER prod_bir
BEFORE INSERT ON product
FOR EACH ROW
BEGIN
  SELECT prod_seq.NEXTVAL
  INTO   :new.prod_id
  FROM   dual;
END;
/
```

Trigger 3:To Auto Increment Payment_Id

```
CREATE SEQUENCE pay_seq START WITH 1
     INCREMENT BY 1 CACHE 100;

CREATE OR REPLACE TRIGGER pay_bir
```

```
BEFORE INSERT ON payment
FOR EACH ROW
BEGIN
  SELECT pay_seq.NEXTVAL
  INTO   :new.pay_id
  FROM   dual;
END;
/
```

Trigger 4: To Auto Increment Order_Id

```
CREATE SEQUENCE ord_seq START WITH 1
     INCREMENT BY 1 CACHE 100;

CREATE OR REPLACE TRIGGER ord_bir
BEFORE INSERT ON placeorder
FOR EACH ROW
BEGIN
  SELECT ord_seq.NEXTVAL
  INTO   :new.order_id
  FROM   dual;
END;
/
```

Trigger 5: To check email and password

```
CREATE OR REPLACE TRIGGER check_signin
BEFORE INSERT ON signin
FOR EACH ROW
DECLARE
     prefix VARCHAR2(1);
     CURSOR cu_signin IS SELECT * FROM signin;
BEGIN
FOR r_signin IN cu_signin LOOP
        IF(r_signin.email = :new.email) THEN
           dbms_output.Put_line('Username already exists....Try
aNOTher one.');
           Raise_Application_Error (-20001, 'Duplicate username
found');
        END IF;
END LOOP;

prefix := substr(:new.password,1,1);
IF(LENGTH(:new.password) != 8) THEN
    dbms_output.Put_line('The length of the password must be 8
characters');
    Raise_Application_Error (-20002, 'Password Length must be
8');
END IF;
IF( (REGEXP_LIKE(prefix, '[a-z]')) OR (REGEXP_LIKE(prefix,
'[0-9]')) OR (REGEXP_LIKE(prefix, '[A-Z]'))) THEN
    dbms_output.Put_line('The first letter of the password must
be special character');
    Raise_Application_Error (-20004, 'first letter of the
password must be special character');
END IF;
END;
/
```

Trigger 6:To check length of Phone no. in customer details

```
CREATE OR REPLACE TRIGGER check_customer
BEFORE INSERT ON customer
FOR EACH ROW
DECLARE
     prefix VARCHAR2(1);
     CURSOR cu_cus IS SELECT * FROM customer;
BEGIN
IF(LENGTH(:new.ph_no) != 10) THEN
    dbms_output.Put_line('The length of the phone number must be
10');
    Raise_Application_Error (-20003, 'Phone number must be size
of 10');
END IF;
END;
/
```

Trigger 7: To subtract quantity from database after order is placed

```
CREATE OR REPLACE TRIGGER quan_sub BEFORE INSERT ON orderdetails
FOR EACH ROW
DECLARE
cursor c_quan IS SELECT quantity FROM product WHERE prod_name =
:new.prod_name;
r_quan c_quan%rowtype;
BEGIN
OPEN c_quan;
  FETCH c_quan INTO r_quan;
  IF (r_quan.quantity<:new.quantity) THEN
    raise_application_error(-20001,'Enough quantity not
available');
  ELSE
```

```
    UPDATE product SET quantity =(r_quan.quantity -
:new.quantity) WHERE prod_name = :new.prod_name;
  END IF;
CLOSE c_quan;
END;
/
```

Trigger 8: To check that quantity which is added is available or not

```
CREATE OR REPLACE TRIGGER check_quan BEFORE INSERT ON
shoppingcart
FOR EACH ROW
DECLARE
cursor c_quan IS SELECT quantity FROM product WHERE prod_name =
:new.prod_name;
r_quan c_quan%rowtype;
BEGIN
OPEN c_quan;
LOOP
FETCH c_quan INTO r_quan;
IF c_quan%notfound THEN
exit;
ELSE
    IF (r_quan.quantity<:new.quantity) THEN
      raise_application_error(-20001,'Enough quantity not
available');
END IF;
END IF;
END LOOP;
CLOSE c_quan;
END;
/
```

Trigger 9:To check length of Phone no. in placeorder

```
CREATE OR REPLACE TRIGGER CHECK_PH
BEFORE INSERT ON placeorder
FOR EACH ROW
DECLARE
BEGIN
IF(LENGTH(:new.ph_no) != 10) THEN
    dbms_output.Put_line('The length of the phone number must be
10');
    Raise_Application_Error (-20003, 'Phone number must be size
of 10');
END IF;
END;
/
```

Functions:

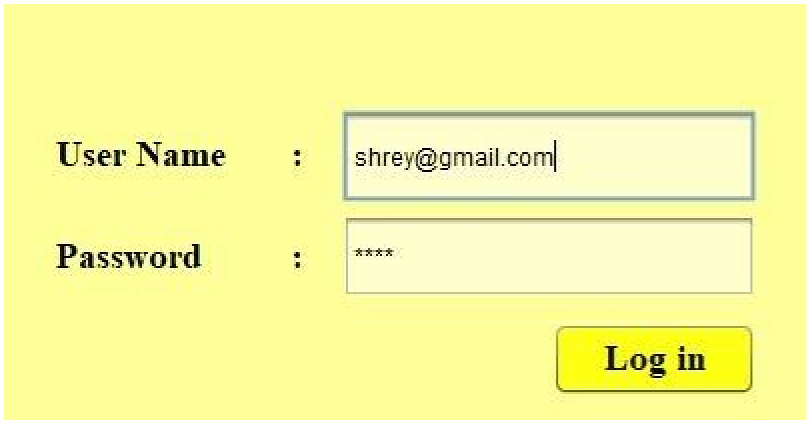
Function 1: To Calculate Total Amount

```
CREATE OR REPLACE FUNCTION tot_amt(shopemail IN
shoppingcart.email%TYPE) RETURN INT AS tot_amt INT;
cursor c_shop IS SELECT price,quantity FROM shoppingcart WHERE
email = shopemail;
r_shop c_shop%ROWTYPE;
BEGIN
tot_amt:=0;
OPEN c_shop;
LOOP
FETCH c_shop INTO r_shop;
IF c_shop%NOTfound THEN
```

```
exit;
ELSE
    tot_amt:=tot_amt+(r_shop.price);
END IF;
END LOOP;
CLOSE c_shop;
RETURN tot_amt;
END;
/
CallableStatement ps1 = myConnect.prepareCall("{? = call
tot_amt(?)}");
```

Images of Output of Functions Triggers and Procedures on the Frontend:

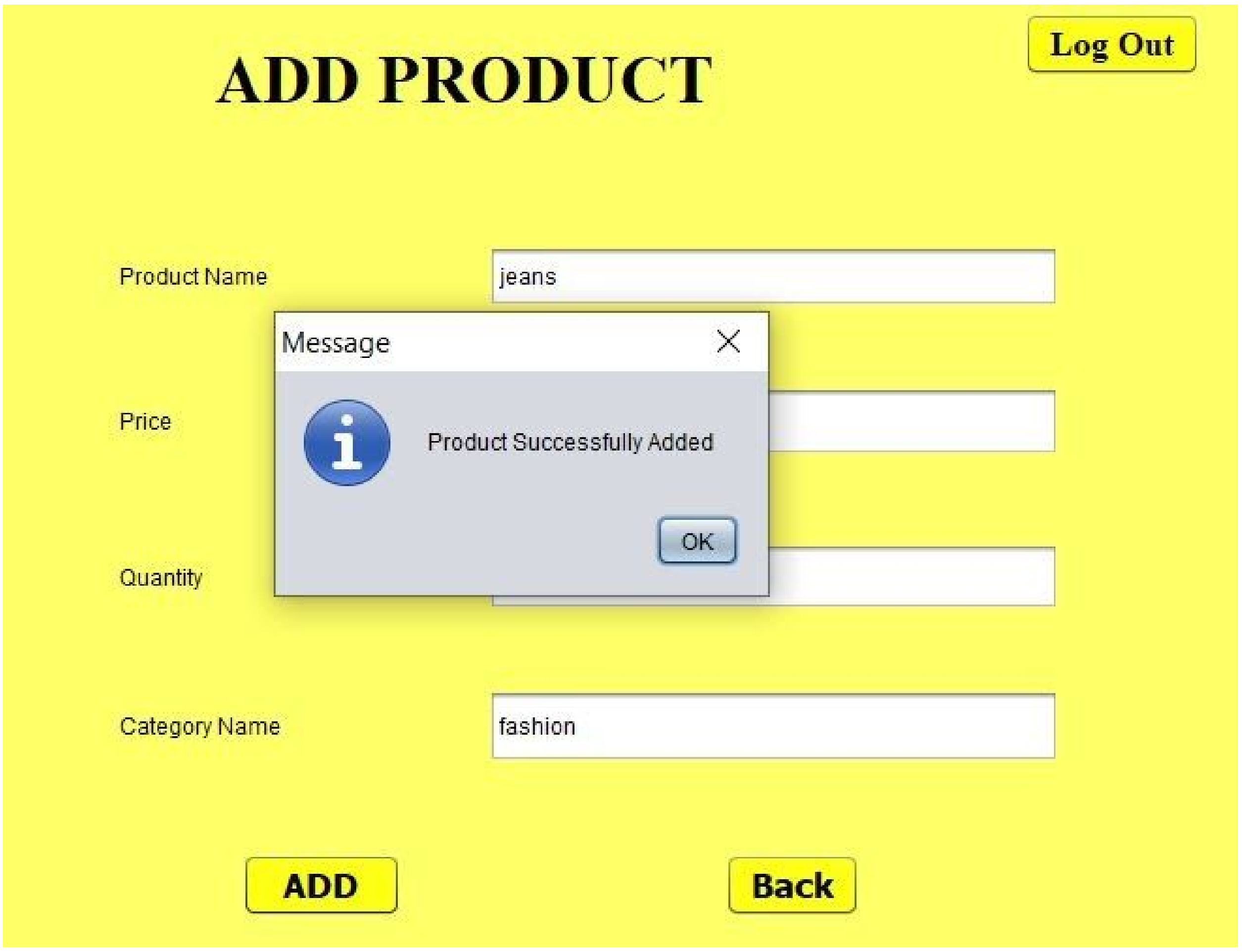
OUTPUT 1:



User Name	:	shrey@gmail.com
Password	:	****

Output of Login Admin to the System.

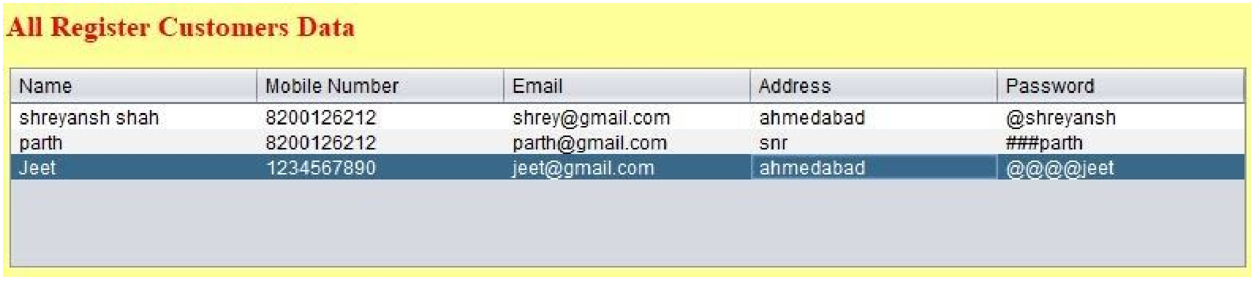
OUTPUT 2:



Product Successfully Added

Output of Admin adding Product in the Fashion Category

OUTPUT 3:

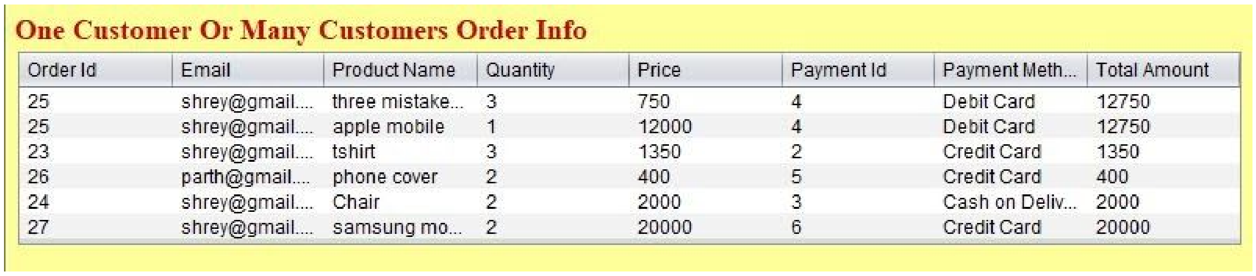


All Register Customers Data

Name	Mobile Number	Email	Address	Password
shreyansh shah	8200126212	shrey@gmail.com	ahmedabad	@shreyansh
parth	8200126212	parth@gmail.com	snr	###parth
Jeet	1234567890	jeet@gmail.com	ahmedabad	@@@@jeet

Output of Admin can see all the Registered User and their Details

OUTPUT 4:

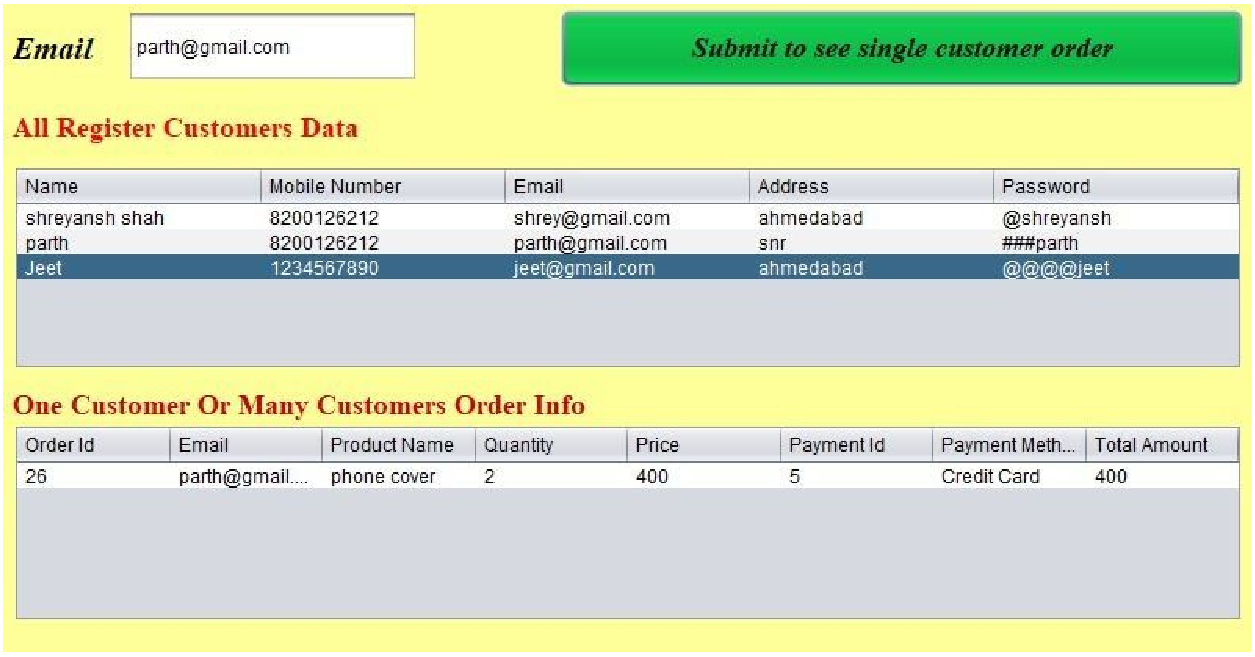


One Customer Or Many Customers Order Info

Order Id	Email	Product Name	Quantity	Price	Payment Id	Payment Meth...	Total Amount
25	shrey@gmail....	three mistake...	3	750	4	Debit Card	12750
25	shrey@gmail....	apple mobile	1	12000	4	Debit Card	12750
23	shrey@gmail....	tshirt	3	1350	2	Credit Card	1350
26	parth@gmail....	phone cover	2	400	5	Credit Card	400
24	shrey@gmail....	Chair	2	2000	3	Cash on Deliv...	2000
27	shrey@gmail....	samsung mo...	2	20000	6	Credit Card	20000

Output of Admin Seeing the Order of a Specific Customer Shrey

OUTPUT 5:



Email: parth@gmail.com

All Register Customers Data

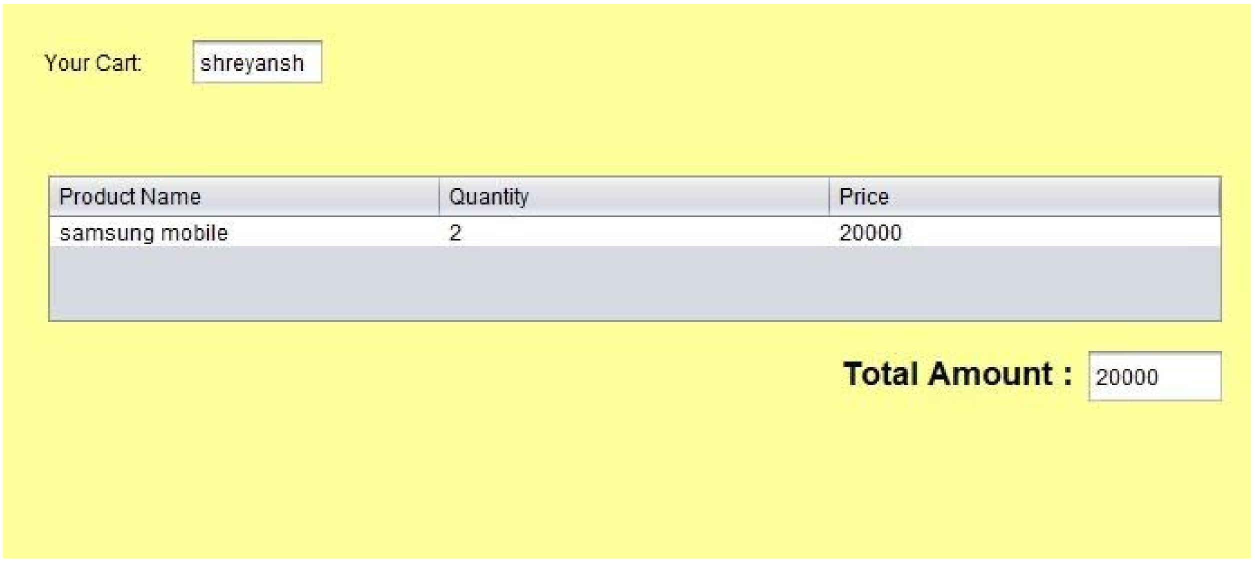
Name	Mobile Number	Email	Address	Password
shreyansh shah	8200126212	shrey@gmail.com	ahmedabad	@shreyansh
parth	8200126212	parth@gmail.com	snr	###parth
Jeet	1234567890	jeet@gmail.com	ahmedabad	@@@@jeet

One Customer Or Many Customers Order Info

Order Id	Email	Product Name	Quantity	Price	Payment Id	Payment Meth...	Total Amount
26	parth@gmail....	phone cover	2	400	5	Credit Card	400

Output of Admin Selecting a Specific User and seeing his information

OUTPUT 6:



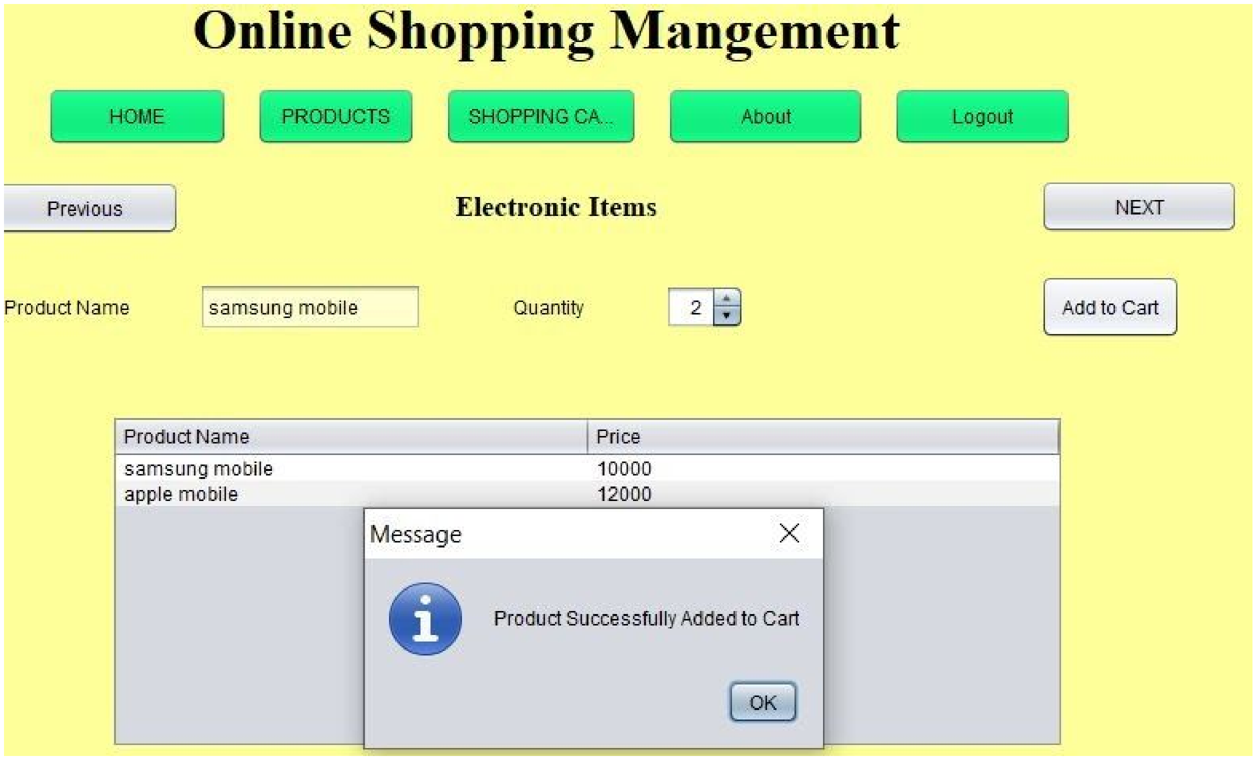
Your Cart: shreyansh

Product Name	Quantity	Price
samsung mobile	2	20000

Total Amount : 20000

Output of a Cart of User Shreyansh

OUTPUT 7:



Online Shopping Mangement

Electronic Items

Product Name samsung mobile Quantity 2

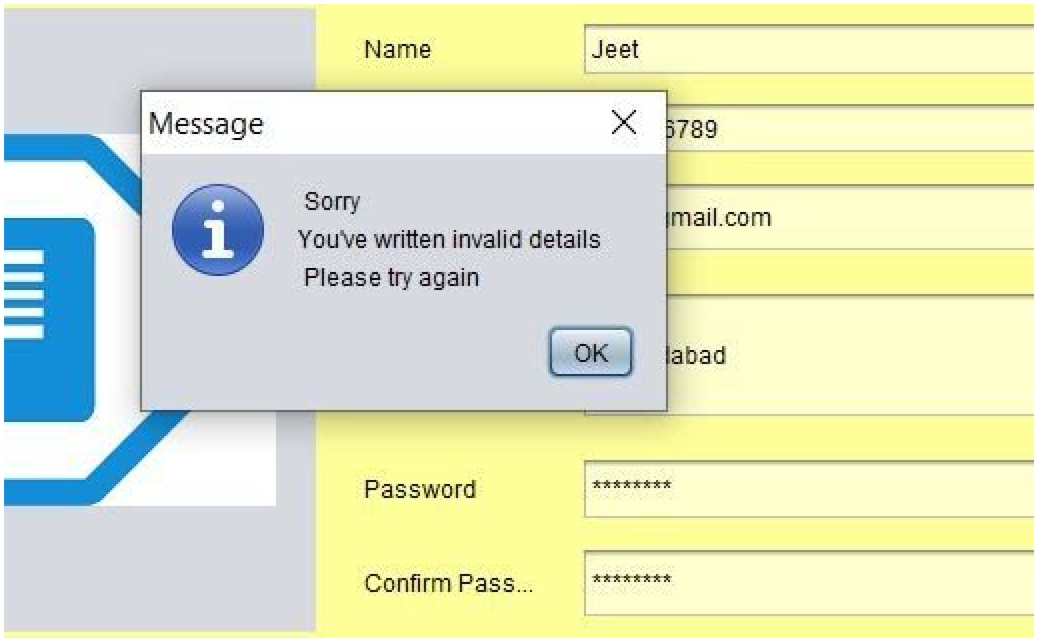
Product Name	Price
samsung mobile	10000
apple mobile	12000

Message

Product Successfully Added to Cart

User adding product to his cart.

OUTPUT 8:



Message

Jeet

6789

mail.com

labad

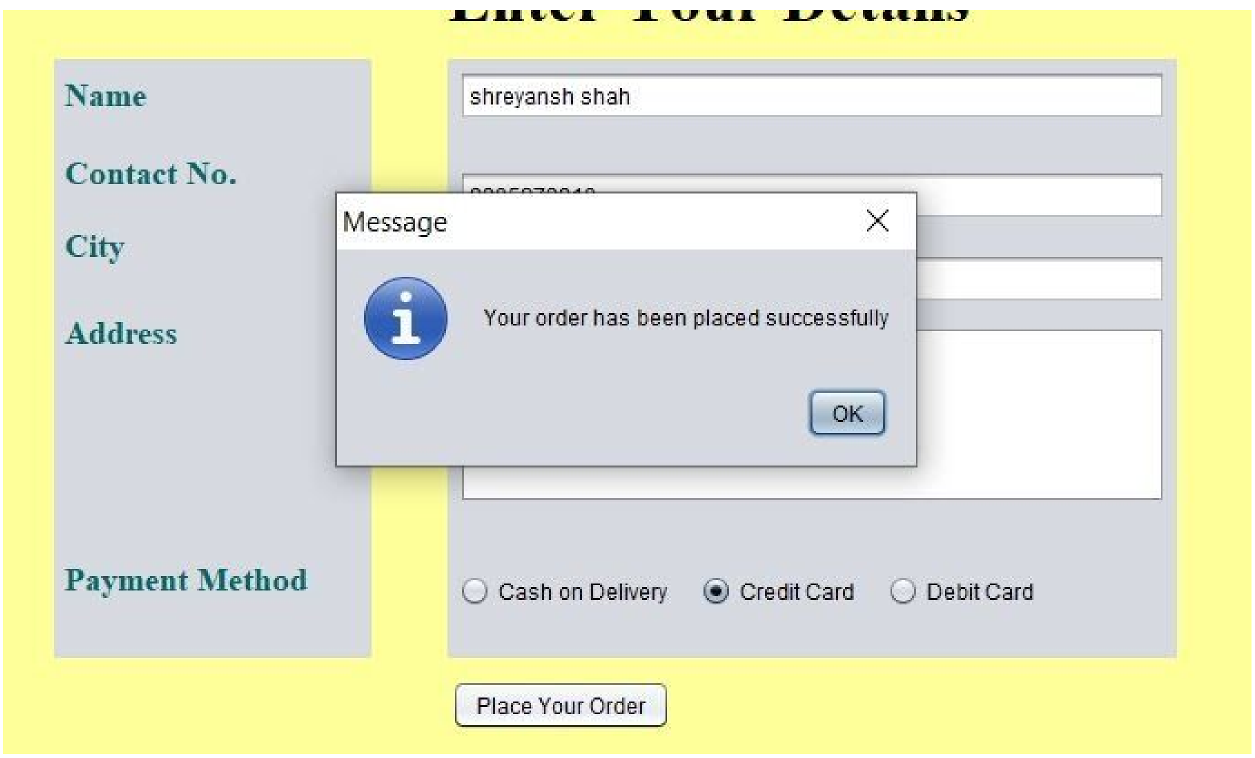
Sorry
You've written invalid details
Please try again

****

****

Validation of Input details of the New User Registering.

OUTPUT 9:



Name

shreyansh shah

Contact No.

City

Address

Message

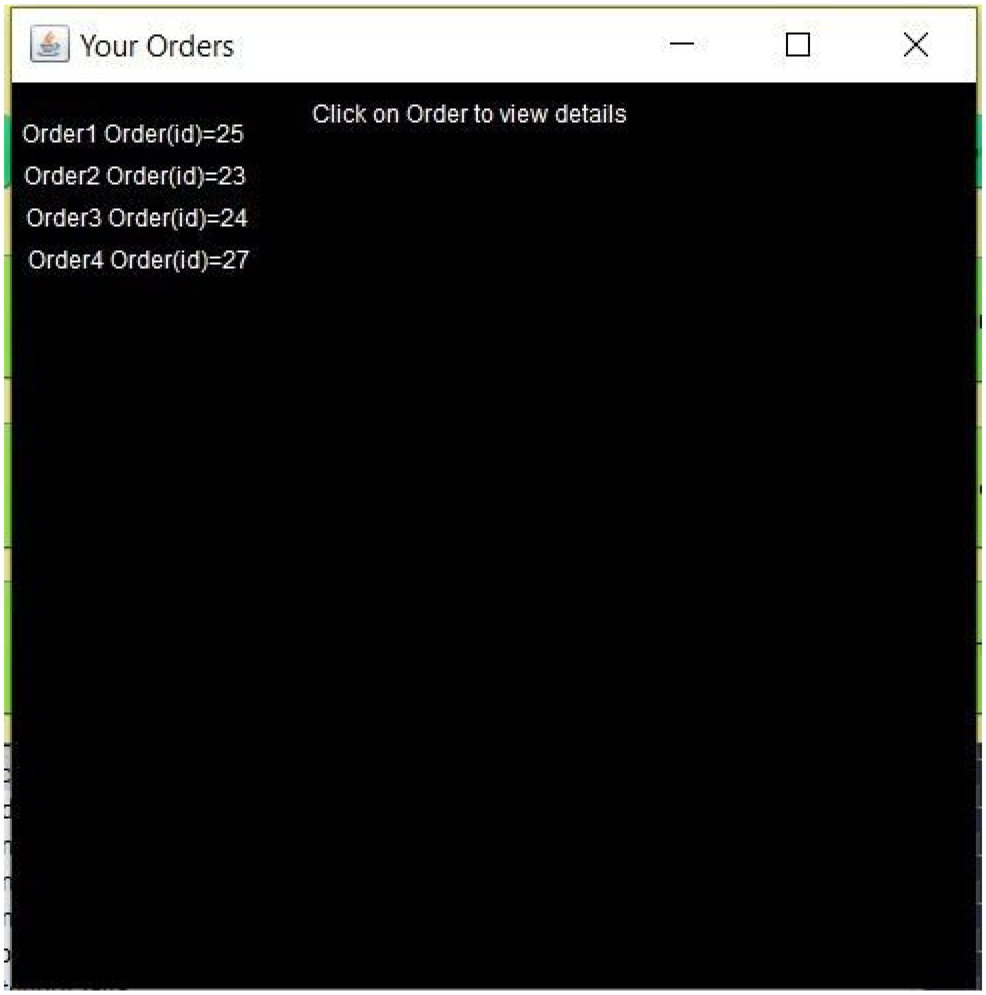
Your order has been placed successfully

Payment Method

( ) Cash on Delivery [✓ Credit Card] ( ) Debit Card

Output of Placing User Shreyansh's Order.

OUTPUT 10:

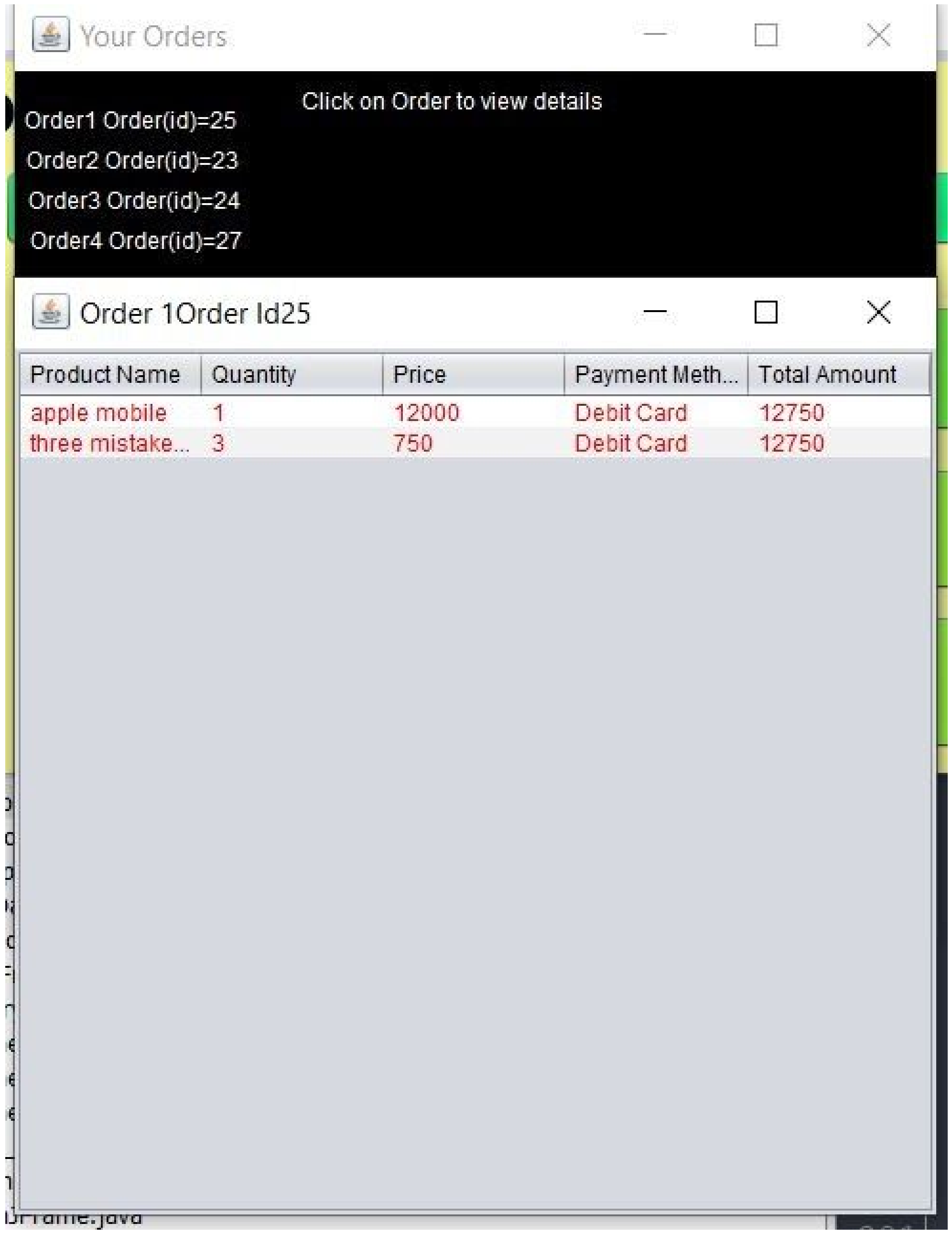

Your Orders

Click on Order to view details

```
Order1 Order(id)=25
Order2 Order(id)=23
Order3 Order(id)=24
Order4 Order(id)=27
```

Total Order of a user.

OUTPUT 11:



Your Orders

Click on Order to view details

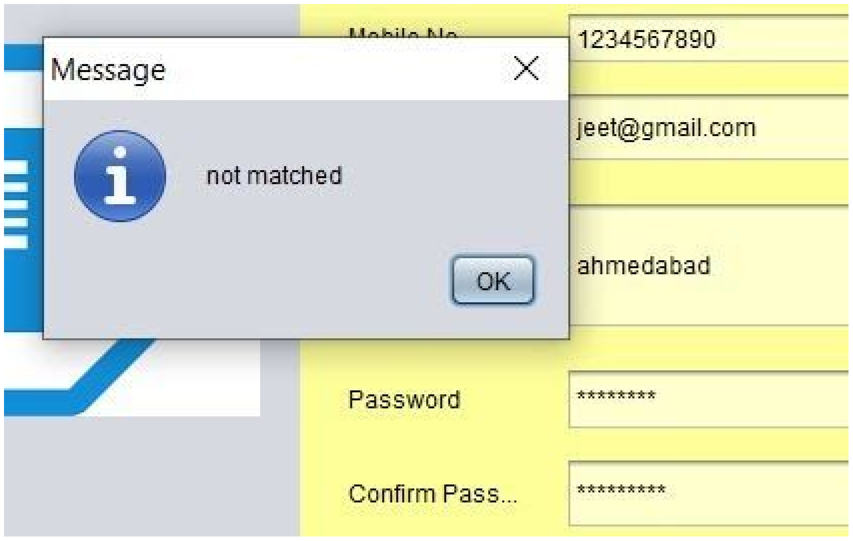
Order1 Order(id)=25
Order2 Order(id)=23
Order3 Order(id)=24
Order4 Order(id)=27

Order 1Order Id25

Product Name	Quantity	Price	Payment Meth...	Total Amount
apple mobile	1	12000	Debit Card	12750
three mistake...	3	750	Debit Card	12750

Viewing history of our Order placed before.

OUTPUT 12:



Mobile No 1234567890

jeet@gmail.com

ahmedabad

Password ********

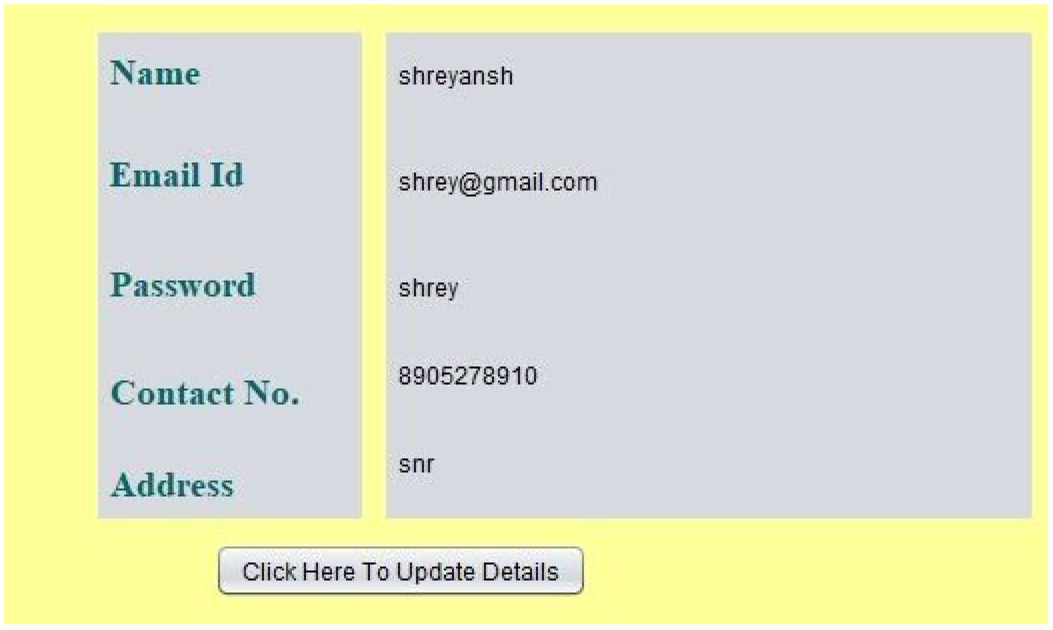
Confirm Pass... *********

Message

not matched

Confirming Password inserted before.

OUTPUT 13:

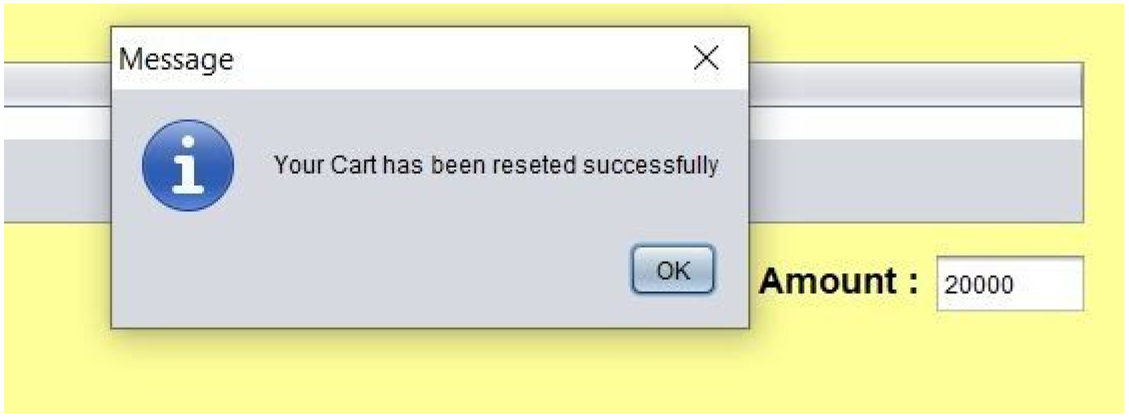


Name	shreyansh
Email Id	shrey@gmail.com
Password	shrey
Contact No.	8905278910
Address	snr

Updating user's details

OUTPUT 14:

OUTPUT 14:

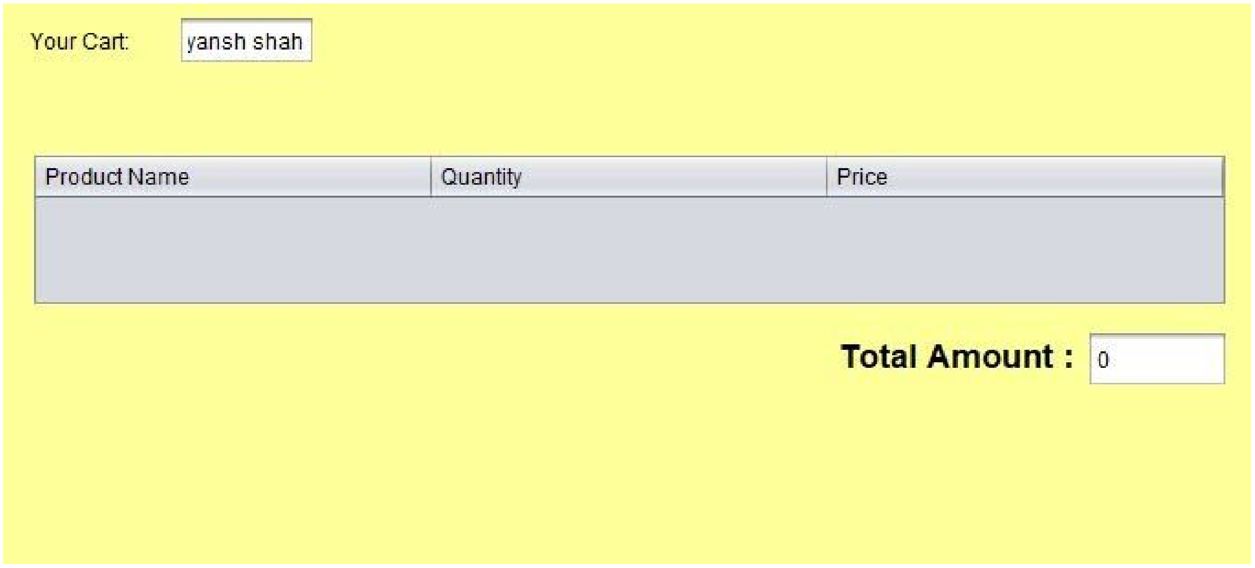

Message

Your Cart has been reseted successfully

Amount : 20000

Resetting our cart

OUTPUT 15:

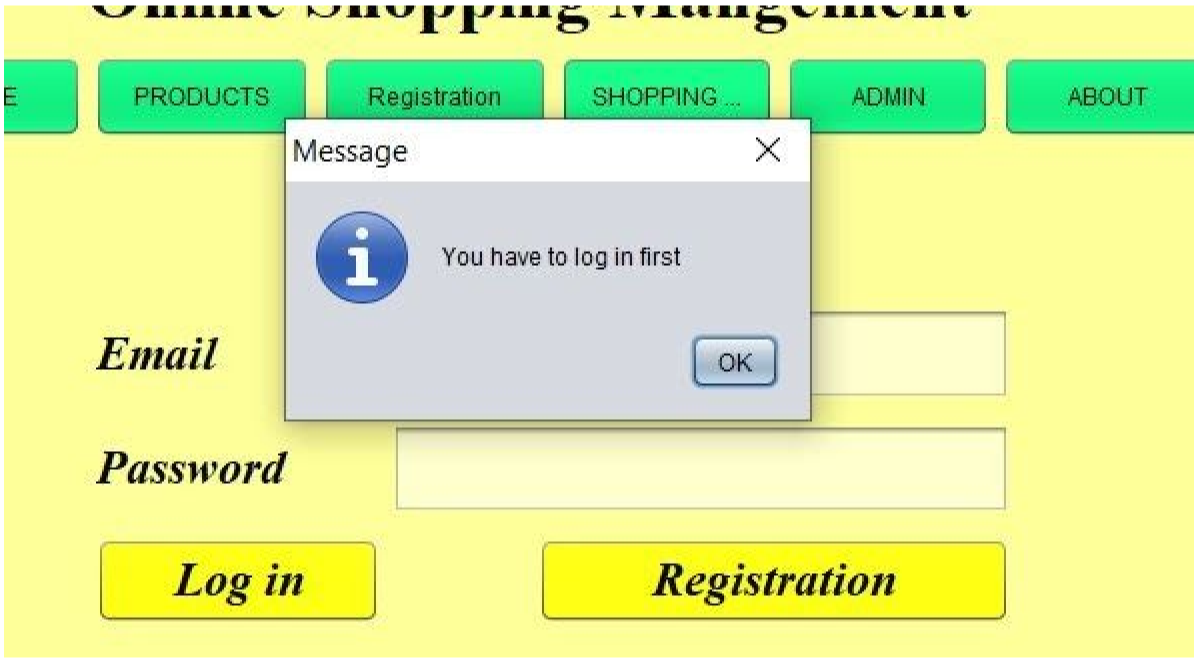

Your Cart: yansh shah

Product Name	Quantity	Price

Total Amount : 0

Cart of the User Reseted.

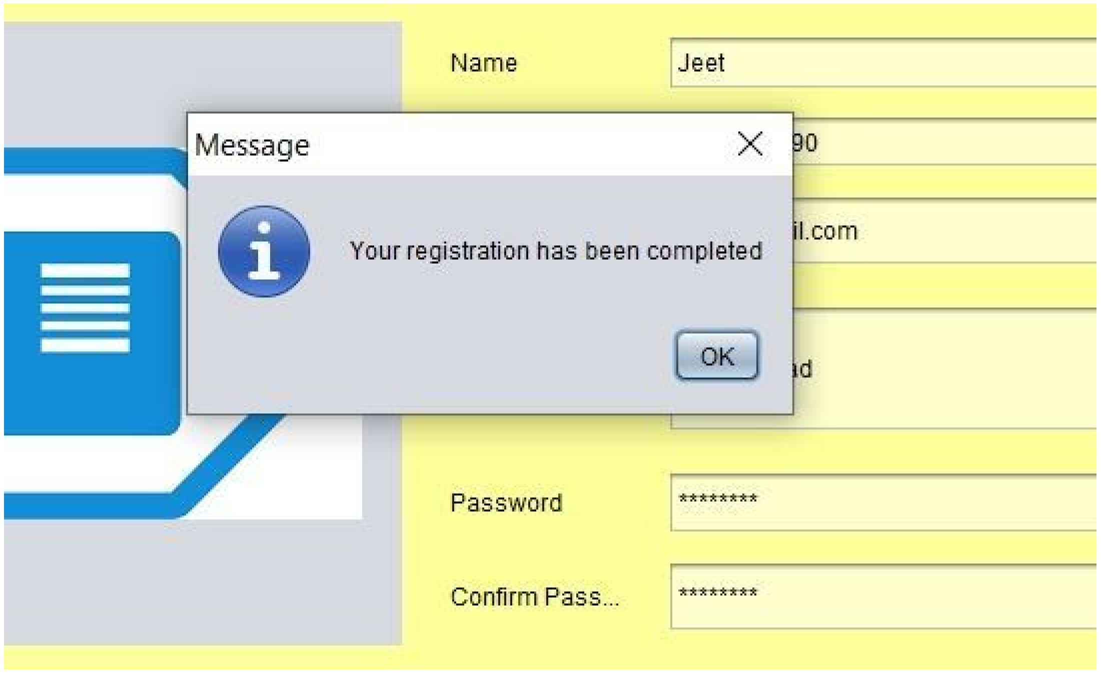
OUTPUT 16:



You have to log in first

Validation for using the System you must have to login first.

OUTPUT 17:



Name: Jeet

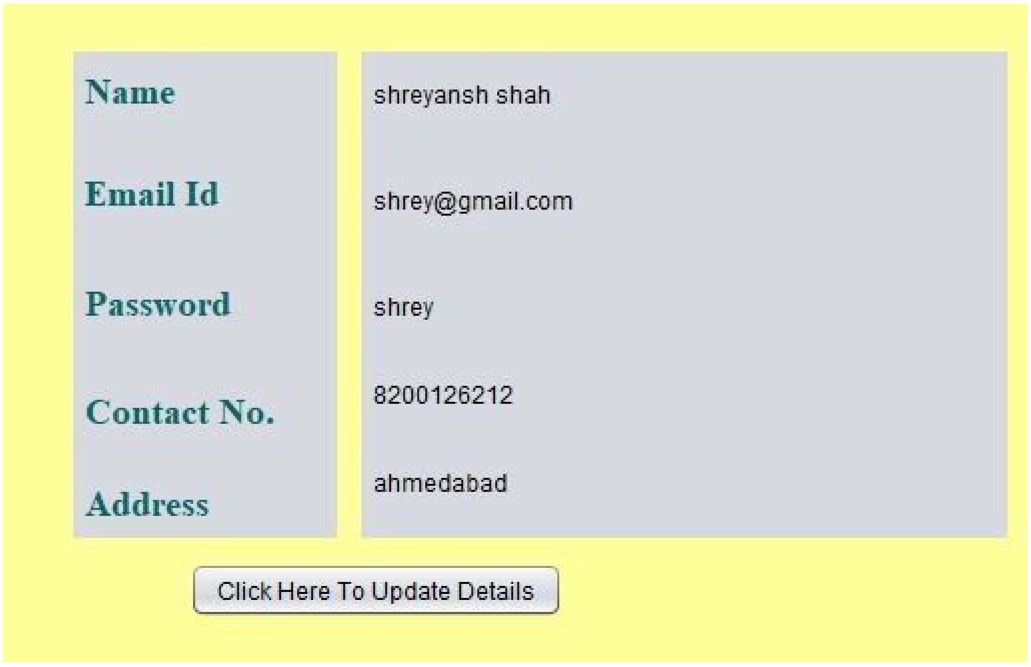
Your registration has been completed

Password: ********

Confirm Pass...: ********

Registration of the User Completed.

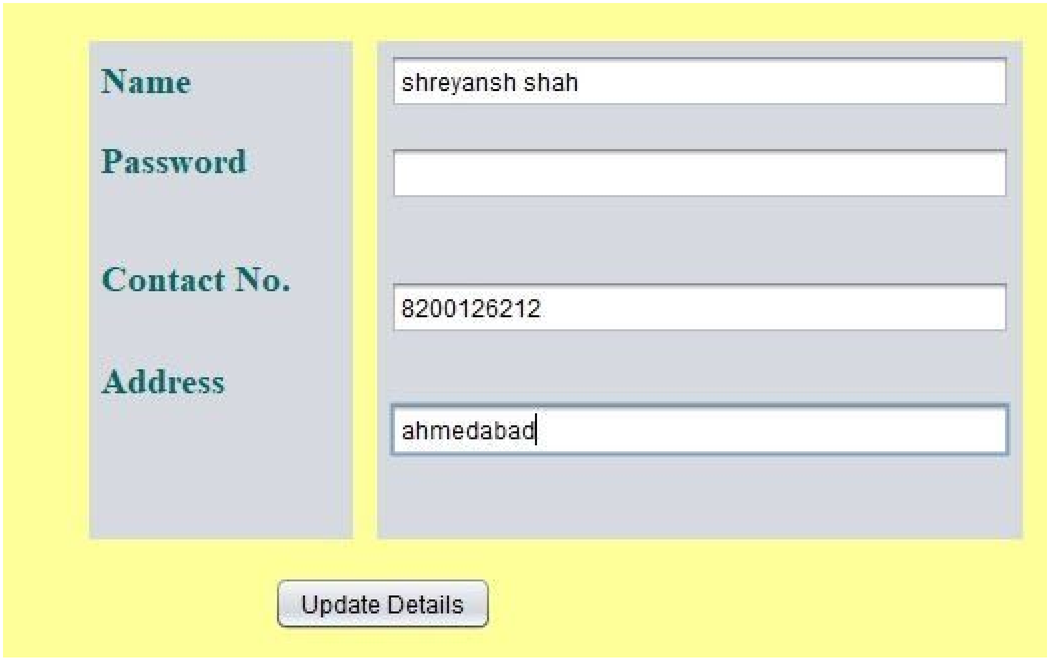
OUTPUT 18:



Name	shreyansh shah
Email Id	shrey@gmail.com
Password	shrey
Contact No.	8200126212
Address	ahmedabad

Updating the User's Details.

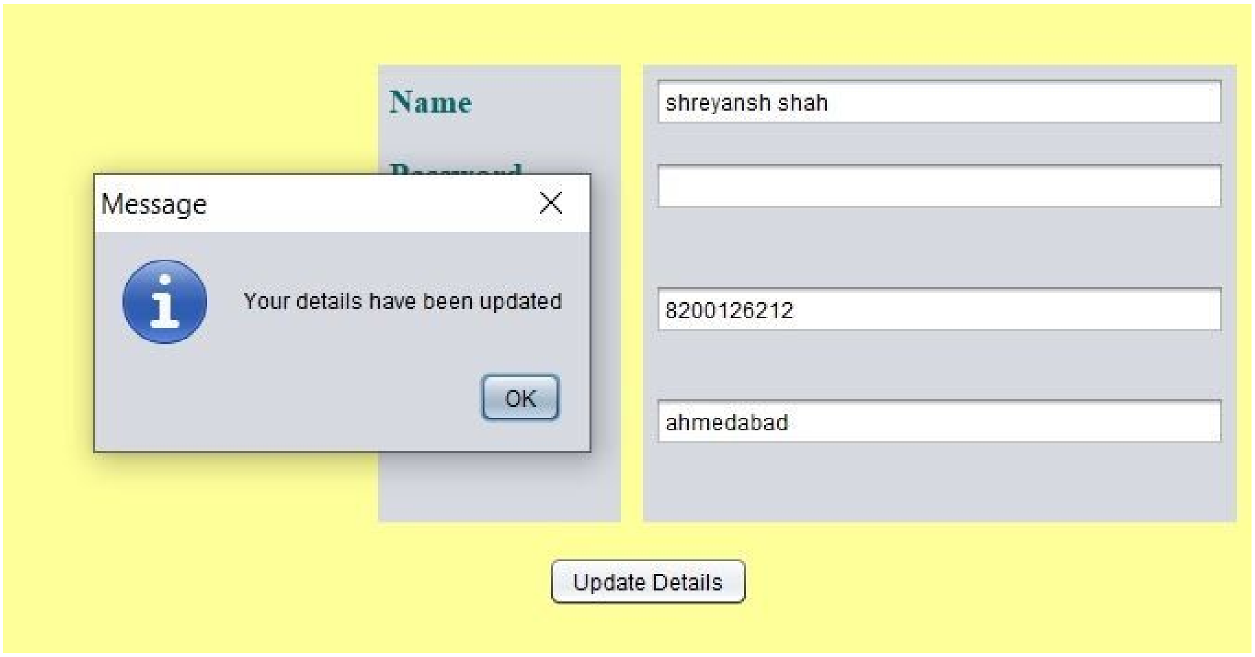
OUTPUT 19:



Name	shreyansh shah
Password	
Contact No.	8200126212
Address	ahmedabad

Checking Validations

OUTPUT 20:



Your details have been updated

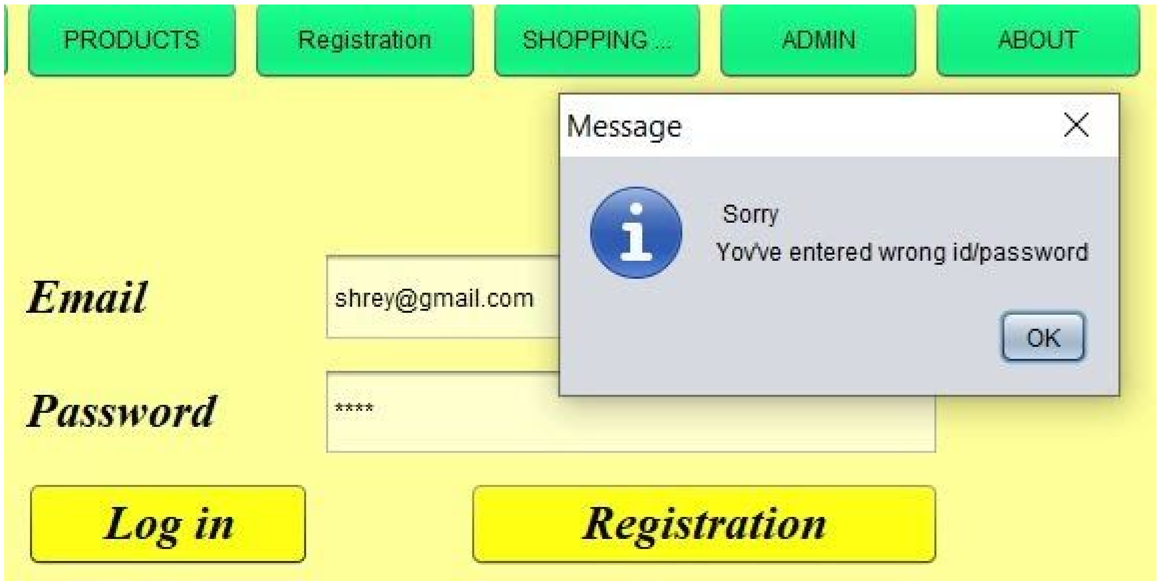
shreyansh shah

8200126212

ahmedabad

Output when user have successfully updated the detail.

OUTPUT 21:



Sorry
Yov've entered wrong id/password

shrey@gmail.com

****

Validations when you have input the wrong password during login to the system.