Online Shopping Cart Database

Mini Project Report -Database Lab (DSE 2260)

Department of Data Science & Computer Applications



B. Tech Data Science

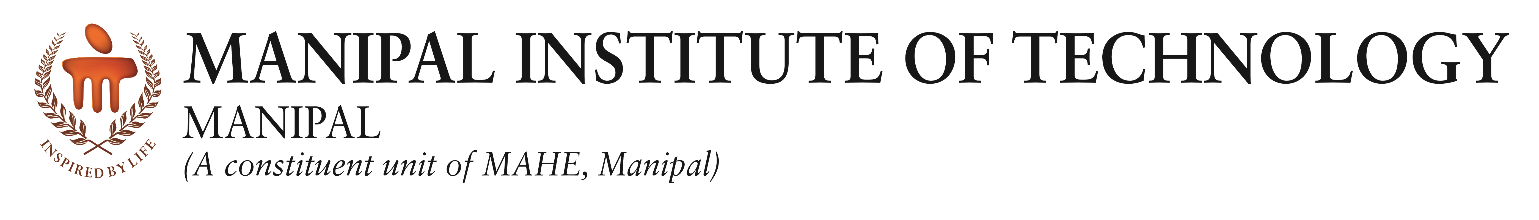
4th Semester – Batch: B1/B2/B3/B4

Submitted By

|  |  |
| --- | --- |
| Kartikeya Tiwari | 200968130 |
| Neel Gangrade | 200968128 |
| Udit Amin | 200968132 |
| Aditi Desai | 200968126 |

**Mentored By**

|  |  |
| --- | --- |
| Vinayak M | Archana H/ Shameem |
| Assistant Professor-Senior | Assistant Professor-Senior |
| DSCA, MIT | DSCA, MIT |



Date: 15-MAY-2022

**CERTIFICATE**

This is to certify that Kartikeya Tiwari (200968130), Udit Amin (200968132), Neel Gangrade (200968128), Aditi Desai (200968126), have successfully executed a mini project title “Online Shopping Cart Database” rightly bringing forth the competencies and skill sets they have gained during the course- Database Lab (DSE 2262 & DSE ), thereby resulting in the culmination of this project.

|  |  |
| --- | --- |
| **Vinayak M** | **Archana H / Shameem** |
| **Assistant Professor-Senior** | **Assistant Professor-Senior** |
| **DSCA, MIT** | **DSCA, MIT** |

**ABSTRACT**

As technology progresses, the world has continued to switch to online mode to fulfil their daily needs. This has given a rise to the online shopping sector for consumers and retailers to connect with each other. In this project, we focus on maintaining a robust database for the end to end purchasing of electronic goods online by users of a web service.

Our methodology for this project was simple, to provide the user with easy access for browsing and purchasing goods and provide the retailer with hassle free access to the said goods while maintaining a cloud database for instant access for verification of payments and replication of the ordered goods and procedure to access said payments and choice of goods.

In the end we were able to create a working database that maintains a live database of all the activities that an online shopping service would require along with providing necessary functionalities to all the parties interacting with the service, retailors and shoppers alike.

It would be pertinent to mention the hardware that we utilised for our purposes. Our primary tool for the creation and maintenance was liveSQL, a database service provided by Oracle. We found the tool extremely handy for the project by providing swift and easy access to the database on all machines and loaded with necessary functionalities

Contents

**Chapter 1** 5

Introduction 5

**Chapter 2** 7

Objectives 7

Proposed System 7

**Chapter 3** 8

Product Details 8

Purchase of Items 8

User Address 8

User Payments 8

User Registering 8

**Chapter 4** 11

Data Dictionary 12

Entity Relationship Diagram 11

Relational Model 16

Schema 12

**Chapter 1**

Introduction

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 or a mobile app. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine, which displays the same product's availability and pricing at different e-retailers.

Online shopping is more popular today than ever before. A report from the U.S. Census Bureau found that in the second quarter of 2021, an estimated $222.5 billion was spent in retail e-commerce sales. In 2011, for comparison, the estimated total for retail e-commerce sales was $47.5 billion. Due in part to new and growing technology, online shopping has become incredibly easy and convenient. It also offers a greater selection than one storefront, opening the doors to products and services that may not be available at a nearby store.

Online stores usually enable shoppers to use "search" features to find specific models, brands or items. Online customers must have access to the Internet and a valid method of payment in order to complete a transaction, such as a credit card, a debit card. When an online store is set up to enable consumers to buy directly from businesses, the process is called business-to-consumer(B2C) online shopping. When an online store is set up to enable businesses to buy from another businesses, the process is called business-to-business (B2B) online shopping. Our project focuses on the B2C online shopping system.

Given the lack of ability to inspect merchandise before purchase, consumers are at higher

risk of fraud than face-to-face transactions. When ordering merchandise online, the item may not work properly, it may have defects, or it might not be the same item pictured in the online photo. To solve these issues, we could start with better quality checks before the shipping process and update the database with the faulty products, so they do not get delivered to a consumer. Merchants also risk fraudulent purchases if customers are using stolen credit cards or fraudulent repudiation of the online purchase. To avoid this issue, we should improve our security measures when it comes to online transactions and payments.

**Chapter 2**

Synopsis

**2.1 Proposed System**

**Problem statement:** Creation of an online platform for users to purchase or sell their electronics products.

The database application allows both buyer and seller to register as the user. The seller can open stores to sell electronic devices and the buyer can purchase products based on shopping cart module.

The database is build based on the basic function including querying products, adding to the shopping cart, and performing payment, collectively used to set up a fully functional online shopping platform for electronics.

* perform registration of both sellers and buyers
* store details of the store and its operation, address and reviews
* make a list of the items bought along with the product details
* provide and accept the preferred mode of payment inputting the required credentials for transaction
* keep track of the buyers address for delivery

**2.2 Objectives**

The Main Objective of the work are.

1. A service to provide a simple solution to Buyers and Sellers across the globe

2. Improve operational efficiency by gathering detailed data and providing the same when required.

3. To automate the process of providing options, description, availability to the potential buyers

4. Provides a medium for interested sellers to display their goods

5. save time, makes paperwork easier, and allow for faster record retrieval when compared to manual data storage.

6. Provide for instantaneous deliver immediately after confirmation of payment.

7. To maintain an account of the discount being provided to the customers

8. Verification of the payment made and send a confirmation of the concerned customer.

9. To verify whether an individual is a seller or buyer respectively

**Chapter 3**

Functional Requirements

**3.1 User Registering**

Every person trying to access the website must have a unique UserID. The same userID will be used by the person to be able to register as both a buyer and a seller.

**3.1.1 New User Registration**

The user must be able to cerate user id and password by supplying appropriate details.

|  |  |
| --- | --- |
| INPUT | UserID, Name, Phone |
| Processing | System has to allot a new unique UserID to customer.  Name must not be left empty.  Only Phone Numbers of Indian Format should be allowed. |
| OUTPUT | User created Successfully message / highlight the information entered which is wrong and allow to renter. |

**3.1.2 Seller/Buyer Registration**

The existing user must be able to login upon entering proper user name and password.

|  |  |
| --- | --- |
| INPUT | userID |
| Processing | Check if the UserID exists in the System already, if it does, allow user to proceed for either as a new buyer or new seller. |
| OUTPUT | IF user is already registered as a buyer/seller the prompt the user so.  ELSE register him as a new user. |

**3.2 User Payments.**

Users should be able to provide their debit card details.

|  |  |
| --- | --- |
| INPUT | Prompt user to enter userID, card details. |
| Processing | User inputs a card number. One user may have only 1 card number at maximum.  However, one card can be used by multiple users. |
| OUTPUT | Card is valid. |

**3.3 User Address**

**3.3.1 Buyer Address**

All buyers should be able to provide a shipping address.

|  |  |
| --- | --- |
| INPUT | Address should be provided along with zipcode. |
| Processing | Zipcode must be a 6 digit number. |
| OUTPUT | IF Zipcode is valid, proceed.  ELSE throw a prompt for invalid zipcode. |

**3.3.2 Seller/Store Address**

All sellers should be able to provide a store address.

|  |  |
| --- | --- |
| INPUT | Address should be provided along with zipcode. |
| Processing | Zipcode must be a 6 digit number. |
| OUTPUT | IF Zipcode is valid, proceed.  ELSE throw a prompt for invalid zipcode. |

**3.4 Product Details**

**3.4.1 Brand Details**

All buyers should be able to provide a shipping address.

|  |  |
| --- | --- |
| INPUT | Brands of the items listed. |

**3.4.2 Products**

All buyers should be able to provide a shipping address.

|  |  |
| --- | --- |
| INPUT | Brand of product, ModelNumber of product, color , no of items & price of each item. |
| Processing | IF Brand of the product must be valid brand present in the system & number of items should be at least 1. |
| OUTPUT | IF Brand is not present,  Either prompt user to register brand OR  Prompt user to check spelling of brand.  IF number of items is less than 1  Prompt for re-entry. |

**3.5 Purchase of Items**

Confirms an order based on the state of payment of the order.

**3.5.1 Payment**

All buyers should be able to provide a shipping address.

|  |  |
| --- | --- |
| INPUT | PaymentID of them item |
| Processing | Given a paymentID, must be a valid paymentID. |
| OUTPUT | IF PaymentID is valid,  Payment Status must be changed to “DONE”  ELSE  Payment Status must be “PENDING”. |

**3.5.2 Payment Failure/ Order Cancellation**

All buyers should be able to provide a shipping address.

|  |  |
| --- | --- |
| INPUT | Order Date of the Order and PaymentID |
| Processing | If a payment is not processed within 7 days of order date, order must be cancelled to prompted to user. |
| OUTPUT | IF SystemDate > OrderDate + 7 and PaymentStatus = “PENDING”  Cancel Order and Prompt User of cancellation and reordering. |

**Chapter 4**

**Detailed Design**

**4.1 ER Diagram**

Diagram

Description automatically generated

**4.2 Schema Diagram**

A picture containing qr code

Description automatically generated

**4.3 Data Dictionary**

**USERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| userid | int | Primary Key | USER\_PK |
| name | varchar(20) |  |  |
| phoneNumber | varchar(20) |  |  |

**BUYER:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| userid | int | Primary Key  Foreign Key references USERS | USERID\_PK\_TO\_USERS |

**SELLER:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| userid | int | Primary Key  Foreign Key references USERS | USERID\_FK\_TO\_USERS |

**DEBITCARD:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| cardNumber | varchar(25) | Primary Key | CARDNUMBER\_PK |
| userid | int | Foreign Key references USERS | USERID\_FK\_TO\_USERS\_DEBIT |

**ADDRESS:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| addrid | int | Primary Key | ADDRID\_PK |
| userid | int | Foreign Key references USERS | USERID\_FK\_TO\_USERS\_ADDRESS |
| name | varchar(50) |  |  |
| contactPhoneNumber | varchar(20) |  |  |
| province | varchar(100) |  |  |
| city | varchar(100) |  |  |
| streetaddr | varchar(100) |  |  |
| postCode | varchar(12) |  |  |

**STORE:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| sid | int | Primary Key | SID\_PK |
| name | varchar(20) |  |  |
| province | varchar(20) |  |  |
| city | varchar(20) |  |  |
| streetaddr | varchar(20) |  |  |
| customerGrade | int |  |  |
| startTime | date |  |  |

**BRAND:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| brandName | varchar(20) | Primary Key | BRANDNAME\_PK |

**PRODUCT:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| pid | int | Primary Key | PID\_PK |
| sid | int | Foreign Key references STORE | SID\_FK\_TO\_STORE |
| brand | varchar(50) | Foreign Key references BRAND | BRAND\_FK\_TO\_BRAND |
| name | varchar(100) |  |  |
| type | varchar(50) |  |  |
| modelNumber | varchar(50) |  |  |
| color | varchar(50) |  |  |
| amount | int |  |  |
| price | int |  |  |

**ORDERITEM:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| itemid | int | Primary Key | ITEMIT\_PK |
| pid | int | Foreign Key  references PRODUCT | PID\_FK\_TO\_PRODUCT |
| price | int |  |  |
| creationTime | date |  |  |

**ORDERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| orderNumber | int | Primary Key | ORDERNUMBER\_PK |
| paymentState | varchar(12) |  |  |
| creationTime | date |  |  |
| totalAmount | int |  |  |

**SAVE\_TO\_SHOPPING\_CART:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| userid | int | Primary Key  Foreign Key references BUYER | USERID\_PK\_TO\_BUYER |
| pid | int | Primary Key  Foreign Key references PRODUCT | PID\_PK\_TO\_PRODUCT |
| addTime | date |  |  |
| quantity | int |  |  |

**CONTAIN:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| orderNumber | int | Primary Key  Foreign Key references ORDERS | ORDERNUMBER\_PK\_TO\_ORDERS |
| itemid | int | Primary Key  Foreign Key references ORDERITEM | ITEMID\_PK\_TO\_ORDERITEM |
| quantity | int |  |  |

**PAYMENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| orderNumber | int | Primary Key  Foreign Key references ORDERS | ORDERNUMBER\_PK\_TO\_ORDERS\_PAYMENT |
| creditcardNumber | varchar(25) | Primary Key  Foreign Key references CREDITCARD | CREDITCARDNUMBER\_PK\_TO\_CREDITCARD |
| payTime | date |  |  |

**DELIVER\_TO:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint Name |
| addrid | int | Primary Key  Foreign Key references ADDRESS | ADDRID\_PK\_TO\_ADDRESS |
| orderNumber | int | Primary Key  Foreign Key references ORDERS | ORDERNUMBER\_PK\_TO\_ORDERS\_DELIVER |
| TimeDelivered | date |  |  |

**MANAGE:**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type(size) | Constraint | Constraint name |
| userid | int | Primary Key  Foreign Key references SELLER | USERID\_PK\_TO\_SELLER\_MANAGE |
| sid | int | Primary Key  Foreign Key references STORE | SID\_PK\_TO\_STORE\_MANAGE |
| SetUpTime | date |  |  |

….

**4.4 Relational Model Implementation**

**TABLES:**

**CREATE TABLE Users** (userid INT NOT NULL, name VARCHAR(20), phoneNumber VARCHAR(20), PRIMARY KEY(userid));

**CREATE TABLE Buyer** (userid INT NOT NULL, PRIMARY KEY(userid), FOREIGN KEY(userid) REFERENCES Users(userid));

**CREATE TABLE Seller** (userid INT NOT NULL, PRIMARY KEY(userid), FOREIGN KEY(userid) REFERENCES Users(userid));

**CREATE TABLE DebitCard (**cardNumber VARCHAR(25) NOT NULL, userid INT NOT NULL, PRIMARY KEY(cardNumber), FOREIGN KEY(userid) REFERENCES Users(userid));

**CREATE TABLE Address** (addrid INT NOT NULL, userid INT NOT NULL, name VARCHAR(50), contactPhone Number VARCHAR(20), province VARCHAR(100), city VARCHAR(100), streetaddr VARCHAR(100), postCode VARCHAR(12), PRIMARY KEY(addrid), FOREIGN KEY(userid) REFERENCES Users(userid));

**CREATE TABLE Store**(sid INT NOT NULL, name VARCHAR(20), province VARCHAR(20), city VARCHAR(20), streetaddr VARCHAR(20), customerGrade INT, startTime DATE, PRIMARY KEY(sid));

**CREATE TABLE Brand**(brandName VARCHAR(20) NOT NULL, PRIMARY KEY (brandName));

**CREATE TABLE Product**(pid INT NOT NULL, sid INT NOT NULL, brand VARCHAR(50) NOT NULL, name VARCHAR(100), type VARCHAR(50), modelNumber VARCHAR(50), color VARCHAR(50), amount INT, price INT, PRIMARY KEY(pid), FOREIGN KEY(sid) REFERENCES Store(sid), FOREIGN KEY(brand) REFERENCES Brand(brandName));

**CREATE TABLE OrderItem** (itemid INT NOT NULL, pid INT NOT NULL, price INT, creationTime DATE, PRIMARY KEY(itemid), FOREIGN KEY(pid) REFERENCES Product(pid));

**CREATE TABLE Orders**(orderNumber INT NOT NULL, paymentState VARCHAR(12), creationTime DATE, totalAmount INT, PRIMARY KEY (orderNumber));

**RELATIONSHIPS:**

**CREATE TABLE Save\_to\_Shopping\_Cart**(userid INT NOT NULL, pid INT NOT NULL, addTime DATE, quantity INT, PRIMARY KEY (userid,pid), FOREIGN KEY(userid) REFERENCES Buyer(userid), FOREIGN KEY(pid) REFERENCES Product(pid));

**CREATE TABLE Contain**(orderNumber INT NOT NULL, itemid INT NOT NULL, quantity INT, PRIMARY KEY (orderNumber,itemid), FOREIGN KEY(orderNumber) REFERENCES Orders(orderNumber), FOREIGN KEY(itemid) REFERENCES OrderItem(itemid));

**CREATE TABLE Payment**(orderNumber INT NOT NULL, creditcardNumber VARCHAR(25) NOT NULL, payTime DATE, PRIMARY KEY(orderNumber,creditcardNumber), FOREIGN KEY(orderNumber) REFERENCES Orders(orderNumber), FOREIGN KEY(creditcardNumber) REFERENCES CreditCard(cardNumber));

**CREATE TABLE Deliver\_To**(addrid INT NOT NULL, orderNumber INT NOT NULL, TimeDelivered DATE, PRIMARY KEY(addrid,orderNumber), FOREIGN KEY(addrid) REFERENCES Address(addrid), FOREIGN KEY(orderNumber) REFERENCES Orders(orderNumber));

**CREATE TABLE Manage**(userid INT NOT NULL, sid INT NOT NULL, SetUpTime DATE, PRIMARY KEY(userid,sid), FOREIGN KEY(userid) REFERENCES Seller(userid), FOREIGN KEY(sid) REFERENCES Store);

**4.5 Queries**

-- 1. select mangalore stores

SELECT \*

from Store

where city = 'Mangalore';

-- 2. update address

UPDATE ADDRESS

SET

city = 'AHMEDABAD',

postCode = 380060,

streetAddr = 'Shyam Villa',

state = 'Gujarat'

WHERE

addrid = 8;

-- 3. Card Info

select u.name, db.cardNumber

from USERS u, DEBITCARD db

where u.userid=db.userid;

-- 4. Brand based on product id:

select brand

from PRODUCT

where pid = 4;

-- 5. Get Details of Users who

SELECT NAME, PHONENUMBER

from users u

where (

u.userId not in(

select b.userId

from Buyer b)

AND

u.userId not in(

select s.userId

from Seller s));

-- 6. Debit Card Updation:

update DEBITCARD

set cardNumber = '1272 3443 8761 0123'

where userid = 14;

--7. Time delivered with address and product name:

select dt.TimeDelivered, dt.addrid, ad.name

from ADDRESS ad, DELIVER\_TO dt

where ad.addrid=dt.addrid and ad.userid=5;

-- 8. find all laptops

SELECT \*

FROM Product

WHERE type='laptop';

-- 9. query the total quantity of products from store with storeid 8 in the shopping cart

SELECT SUM(quantity) AS totalQuantity

FROM Save\_to\_Shopping\_Cart

WHERE Save\_to\_Shopping\_Cart.pid IN (SELECT Product.pid FROM Product WHERE sid=8);

-- 10. query the name and address of orders delivered on 2017-02-17

SELECT name, streetaddr, city

FROM Address

WHERE addrid IN (SELECT addrid FROM Deliver\_To WHERE TimeDelivered = '2017-02-17');

**4.7 Triggers**

-- 1. Once Payment is Confirmed, Add it to database.

CREATE or REPLACE TRIGGER update\_payments

AFTER

INSERT

ON Orders

FOR EACH ROW

WHEN (new.PaymentState = 'Paid')

DECLARE

v\_onum Payment.orderNumber%type;

v\_dc varchar(25);

v\_payTime DATE;

BEGIN

v\_dc := '4457 4992 7391 0219';

select sysdate

into v\_paytime

from dual;

v\_onum := :new.orderNumber;

insert into Payment VALUES(v\_onum , v\_dc , v\_paytime);

END update\_payments;

-- 2. Once Delivery Details are updated, Dispatch Delivery

CREATE OR REPLACE TRIGGER update\_delivery\_details

AFTER

INSERT

ON Orders

FOR EACH ROW

when (new.paymentState = 'Paid')

DECLARE

v\_addrid address.addrid%type;

v\_ordernum orders.orderNumber%type;

BEGIN

v\_ordernum := :new.orderNumber;

v\_addrid := 1;

insert into DELIVER\_TO values(v\_addrid, v\_ordernum, NULL);

END;

-- 3. Send Confirmation once when delivery is done.

CREATE OR REPLACE TRIGGER dispatch\_delivery

AFTER

INSERT

ON deliver\_to

FOR EACH ROW

when (new.TimeDelivered is not NULL)

DECLARE

BEGIN

dbms\_output.put\_line("DELIVERY DONE");

END dispatch\_delivery;

**4.8 Stored Procedures**

-- 1. Check Payment status

CREATE OR REPLACE PROCEDURE PaymentStatus IS

v\_orderNumber ORDERS.orderNumber%TYPE;

v\_paymentState ORDERS.paymentState%TYPE;

v\_totalAmount ORDERS.totalAmount%TYPE;

BEGIN

v\_orderNumber := 72839301;

select paymentState, totalAmount INTO v\_paymentState, v\_totalAmount from ORDERS where orderNumber=v\_orderNumber;

dbms\_output.put\_line('Payment Status: '||v\_paymentState);

dbms\_output.put\_line('Total Amount: '||v\_totalAmount);

END PaymentStatus;

-- 2. Provide Discount on Items

CREATE OR REPLACE PROCEDURE Discount(v\_discount IN ORDERITEM.price%TYPE, v\_pid IN PRODUCT.pid%TYPE) IS

BEGIN

UPDATE ORDERITEM

SET price = price - v\_discount\*price

WHERE pid = v\_pid;

END Discount;

**4.9 Stored Functions**

-- 1. is user is registered as buyer

CREATE OR REPLACE FUNCTION isBuyer(v\_userId IN users.userId%TYPE)

RETURN BOOLEAN IS

isusers BOOLEAN;

cursor c\_buyers is

select userId

from buyer;

BEGIN

isusers := false;

for crsr in c\_buyers loop

if v\_userId = crsr.userId then

isusers := true;

end if;

end loop;

return isusers;

END isBuyer;

-- 2. find if user is registered as seller

CREATE OR REPLACE FUNCTION isSeller(v\_userId IN users.userId%TYPE)

RETURN BOOLEAN IS

isusers BOOLEAN;

cursor c\_sellers is

select userId

from seller;

BEGIN

isusers := false;

for crsr in c\_sellers loop

if v\_userId = crsr.userId then

isusers := true;

end if;

end loop;

return isusers;

END isSeller;

-- PL/SQL to check if User is Registered as a Buyer and Seller Both

DECLARE

v\_userid users.userid%type;

isByr BOOLEAN;

isSlr BOOLEAN;

BEGIN

v\_userid := 1;

isByr := isBuyer(v\_userid);

isSlr := isSeller(v\_userid);

if isByr = true AND isSlr = true then

dbms\_output.put\_line('User is Registered as a Buyer and Seller Both');

elsif isByr = true AND isSlr = false then

dbms\_output.put\_line('User is Registered as a Buyer Only');

elsif isByr = false AND isSlr = true then

dbms\_output.put\_line('User is Registered as Seller Only');

else

dbms\_output.put\_line('User NOT Registered');

end if;

END;

-- Get Cheapest type of product

CREATE OR REPLACE FUNCTION LowestPrice(v\_product IN PRODUCT.type%type) return int is

v\_name users.name%type;

v\_price product.price%type;

Begin

select name, price

into v\_name, v\_price

from PRODUCT

where price = (select MIN(price) from PRODUCT);

return v\_price;

END;

-- PL/SQL Block to print cheapest type of product

DECLARE

v\_type varchar2(20) := 'laptop';

v\_price product.price%type;

BEGIN

v\_price := lowestprice(v\_type);

dbms\_output.put\_line('Cheapest ' || v\_type || ' costs ' || v\_price);

END;

**7. Result**

The application can be used for any Ecommerce application. It is easy to use. It has been thoroughly tested and implemented.

The overall idea of doing this project is to get a real time experience. Learn new technologies.

**8.** **Conclusion**

The concept of Online Shopping has revolutionarized the idea of buying and selling of goods

Customers can purchase items from the comfort of their own homes or work place, without a time barrier.

standing in queues in cash counters are eliminated.

Buyers can also look for the products that are required by them by entering the key words or using search engines.

price comparisons delivery status and order tracking are also available online.

Due to elimination of maintenance, real-estate cost, the retailers are able to sell the products with attractive discounts through online

Few of the large online shopping sites offer store comparison.

This database project on ‘Online Shopping’ is designed to provide a web based application that would make searching, viewing and selection of a product easier. The search engine provides an easy and convenient way to search for products where a user can Search for a product interactively and the search engine would refine the products available based on the user’s input. The user can then view the complete specification of each product. They can also view the product reviews and also write their own reviews.

**9 Scope for future work**

The following things can be done in future.

• The current system can be extended to allow the users to create accounts and

save products in to wish list.

• The users could subscribe for price alerts which would enable them to receive

messages when price for products fall below a particular level.

• The current system is confined only to the shopping cart process. It can be

extended to have an easy to use check out process.

• Users can have multiple shipping and billing information saved. During

checkout they can use the drag and drop feature to select shipping and billing

information.