

# Database Design And Implementation For E-Commerce

Sourabh Aggarwal (111601025), Nikhil Kumar Yadav (111601013)

March 27, 2019

## Contents

<b>1</b>	<b>Contribution</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>1</b>
<b>3</b>	<b>Requirements</b>	<b>1</b>
<b>4</b>	<b>Entity Relation Diagram</b>	<b>3</b>
4.1	Enhanced ERD . . . . .	3
4.2	Simple ERD . . . . .	4
<b>5</b>	<b>Database Schema And Normalization</b>	<b>5</b>
<b>6</b>	<b>Roles, Triggers, Views</b>	<b>9</b>
6.1	Roles . . . . .	9
6.2	Views . . . . .	9
6.2.1	Customer Views . . . . .	9
6.2.2	Seller Views . . . . .	10
6.2.3	Shipper Views . . . . .	10
6.3	Triggers . . . . .	11
<b>7</b>	<b>Functions And Procedures</b>	<b>13</b>
7.1	Customer Procedures . . . . .	13
7.2	Seller Procedures . . . . .	15
7.3	Shipper Procedures . . . . .	16
<b>8</b>	<b>Use Cases</b>	<b>16</b>

<b>9 Progress And TODOS</b>	<b>19</b>
9.1 GUI . . . . .	19
<b>10 Useful links</b>	<b>19</b>

# 1 Contribution

Since we are a two member team, each of us was involved constantly in each phase of the project, be it GUI, Procedures, Report etc. So, the division of task within ourselves isn't that straight forward to describe, just understand that we did everything together.

For detailed look at the different phases of our project, please refer github.

# 2 Introduction

In this project, our aim was to come up with a reasonably scalable database along with basic GUI for E-Commerce purpose.

We started by listing down various requirements presented in the next section. After that we proceed on building an ER Diagram to fulfill the same. After that it was time to implement all this in SQL. In due time, we managed to put various important features provided by almost all E-Commerce site in our project.

So the following report touches on each of these aspects in brief and sequential manner.

# 3 Requirements

Following is the list of requirements.

1. Idea of roles. Model should have role for each of Customer, Seller and Shipper. Each user is thus assigned its role.
2. Company maintains the details of its users like their name, address, phone number and email-id.
3. Each user has security credentials like their username and password.
4. It should be possible for each user to update their information including their password.
5. Customer requirements
  - (a) Customers should be able to browse all products.

- (b) They should be able to add those products in their cart. Thus each Customer should possess a cart into which they can add products for purchasal.
- (c) Note that each product could be sold by different sellers, show customer should be able to select a product corresponding to the seller he or she wished to buy from.
- (d) Once all the items are added in cart, Customer should then be able to purchase all those items in cart at once.
- (e) For each of the product the Customer has bought, it should be possible for customer to pass the rating and review for both the product as well as seller.

#### 6. Seller requirements

- (a) Each Seller possesses a rating which he or she should be able to see. This rating is given and updated by the users of role Customer who purchased the product from Seller.
- (b) Seller should list down the products he or she is willingly to sell. Each product should have:
  - i. Product Name
  - ii. Product Image
  - iii. Price
  - iv. Quantity which the seller possess of the product
  - v. Pickup address
  - vi. Description
  - vii. Rating (Product rating is again updated by the users of role Customer)
- (c) It should be possible for seller to add or update his or her products.
- (d) It should be possible for seller to see his or her past sellings.
- (e) It should be possible for seller to see the products which he or she has listed.
- (f) It should be possible for seller to see their earnings in a specific duration.

- (g) Seller should be able to browse Shippers so that he or she can decide and contact various Shippers.
  - (h) It should be possible for seller to see the latest purchases of his or her products which are pending to be shipped, thus seller should also have an interface to update customer with the sold product's shipment Tracking ID.
7. Shipper requirements
- (a) It should be possible for shipper to see his or her past shipments (using our interface) including source and destination of the product.

## 4 Entity Relation Diagram

### 4.1 Enhanced ERD

Enhanced Entity Relation Diagram can be represented in various notations, below shows the notation which we have followed.

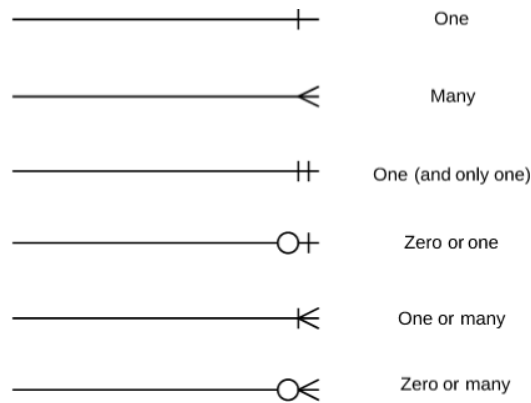


Figure 1: Entity Relation Diagram Notation

Following this notation, our ERD is represented below.

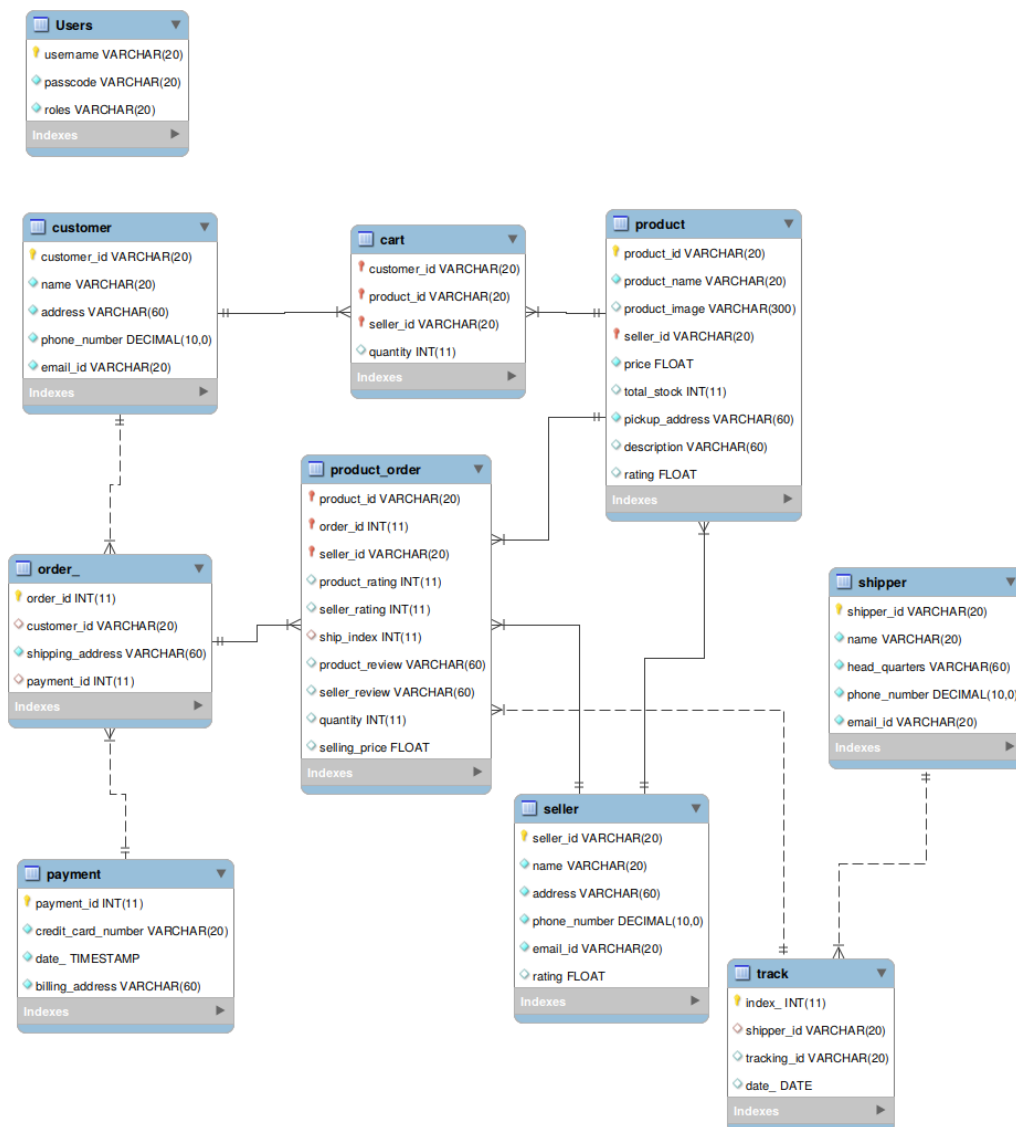


Figure 2: Entity Relationship Diagram for e-commerce

## 4.2 Simple ERD

TODO

## 5 Database Schema And Normalization

In this section we describe various aspects of our schema and also mention that it is indeed normalized (BCNF).

*Notation:* A dependency  $A \rightarrow B$  is called relevant if all other dependencies from  $A$  are of the form  $A \rightarrow C$  where  $C \subseteq B$ .

- A table for basic details of customer.

```
/* Here: customer_id -> R is the only relevant
dependency and hence it is in BCNF */
create table customer (
  customer_id VARCHAR (20) primary key not null,
  name VARCHAR (20) not null,
  address VARCHAR (60) not null,
  phone_number DECIMAL (10) UNSIGNED not null,
  email_id VARCHAR (20) not null
);
```

- A table for basic details of seller.

```
/* Here: seller_id -> R is the only relevant
dependency and hence it is in BCNF */
/* Rating will be updated with the help of triggers.
*/
create table seller (
  seller_id varchar (20) primary key not null,
  name varchar (20) not null,
  address varchar (60) not null,
  phone_number decimal (10) UNSIGNED NOT NULL,
  email_id VARCHAR (20) not null,
  rating float
);
```

- A table for basic details of shipper.

```
/* Here: shipper_id -> R is the only relevant
dependency and hence it is in BCNF */
create table shipper (
  shipper_id varchar (20) primary key not null,
  name varchar (20) not null,
  head_quarters varchar (60) not null,
  phone_number decimal (10) UNSIGNED not null,
```

```
email_id VARCHAR (20) not null
);
```

- Having described these basic tables, we can now describe table for products. Note that each product can be sold by different sellers in different price and quantity, thus, primary key is formed by both product\_id and seller\_id.

```
/* Here: (product_id, seller_id) -> R is the only
relevant dependency and hence it is in BCNF */
/* Rating will be updated with the help of triggers.
*/
create table product (
    product_id varchar (20) not null,
    product_name varchar (20) not null,
    seller_id varchar (20) not null,
    price float not NULL,
    total_stock int,
    pickup_address varchar (60) not null,
    description varchar (60),
    rating float,
    foreign key (seller_id) references seller
    (seller_id) on delete cascade,
    primary key (product_id, seller_id)
);
```

- When user makes a payment, we want to store payment details for which we have the following table.

```
/* Here: payment_id -> R is the only relevant
dependency and hence it is in BCNF */
create table payment (
    payment_id VARCHAR (20) primary key not null,
    credit_card_number VARCHAR (20) not null,
    date_ timestamp,
    billing_address varchar(60) not null
);
```

- User will have a front end feature to add items in cart. When the user is ready to buy, it will generate an *order\_id* for all those products which he or she chose. Note that *order\_id* will be generated *only* when user successfully does the payment.



```

/* Here: order_id -> R is the only relevant
dependency and hence it is in BCNF */
create table order_ (
  order_id VARCHAR (20) primary key not null,
  customer_id VARCHAR (20),
  shipping_address varchar(60) not null,
  payment_id VARCHAR (20),
  foreign key (customer_id) references customer
(customer_id) on delete set null,
  foreign key (payment_id) references payment
(payment_id) on delete set null
);

```

- After generating the payment, we have to put the details of the bought items along with their order\_id.

```

/* Here: (product_id, order_id, seller_id) -> R is
the only relevant dependency and hence it is in BCNF
*/
create table product_order (
  product_id varchar(20) not null,
  order_id varchar (20) not null,
  seller_id varchar (20),
  product_rating int check (product_rating in (NULL,
1, 2, 3, 4, 5)),
  seller_rating int check (seller_rating in (NULL, 1,
2, 3, 4, 5)),
  ship_index int,
  product_review varchar (60),
  seller_review varchar (60),
  quantity int,
  selling_price float,
  primary key (product_id, order_id, seller_id),
  foreign key (product_id) references product
(product_id) on delete cascade,
  foreign key (order_id) references order_ (order_id)
on delete cascade,
  foreign key (seller_id) references seller
(seller_id) on delete cascade,
  foreign key (ship_index) references track (index_)
on delete set null
);

```

- Note that we used a foreign key in the above table which we haven't defined yet, which is *ship\_index* . It is basically a unique identifier for each ordered product serving as an index of track table which we will use to track our items.

```
/* Here: index_ -> R is the only relevant dependency
and hence it is in BCNF */
create table track (
  index_ INT AUTO_INCREMENT primary key not null,
  shipper_id varchar (20),
  tracking_id varchar (20),
  foreign key (shipper_id) references shipper
    (shipper_id) on delete set null
);
```

- We also have an auxiliary table for keeping track of users with their old passwords as mysql.user encrypts the passwords and there is no way to get it back also this table is required for validation when the user tries to log in the system.

```
-- Clearly this table is in BCNF.
create table Users (
  username VARCHAR (20) primary key not null,
  passcode VARCHAR (20) not null,
  roles VARCHAR (20) not null
);
```

## 6 Roles, Triggers, Views

### 6.1 Roles

As mentioned before, we have three roles, viz., Customer, Seller and Shipper. And of course above them all we have database administrator, "root", viz. "dbadmin". Each role will be able to access views, functions, procedures written for it and granted to it.

### 6.2 Views

Almost all what we wanted to achieve was possible with the help of procedures, views, thus have little to offer but still we are listing here all the views that we wrote irrespective of whether they are actually part of our GUI.

#### 6.2.1 Customer Views

1. View that will allow customer to view its profile, "customerProfile".
2. View that will allow customer to see the total cost of his/her various orders, i.e. total money spent on the platform till date, "orderPrice".
3. View that will allow customer to see and add products to his cart, "showCart".
4. View that will tell the customer details corresponding to his/her all order\_id mentioning complete order details (order\_id, shipping\_address, date\_, total\_price) except the products in that order, "previousOrders".
5. View that will allow customer to just see his various order\_id, "listOrders".
6. View that will give us shipment index (ship\_index), order ID, product ID from product\_order relation corresponding to our orders, "trackID".
7. View that will augment the above view with tracking ID as well, "packageStatus".

### 6.2.2 Seller Views

1. View that will allow seller to view its details, "sellerProfile".
2. View that will allow seller to see his/her various products, "sellerProducts".
3. View that will allow seller to see various orders which he or she have sold (seller\_id, product\_id, quantity, selling\_price, date\_), "sellerOrders".

### 6.2.3 Shipper Views

1. View that will allow shipper to view its details, "shipperProfile".
2. View that will allow shipping details (pickup\_address, shipping\_address, tracking\_id), "shipperTrack".

And their details is best understood with the help of the following code:

```
CREATE ROLE dbadmin;
CREATE ROLE customer;
CREATE ROLE seller;
CREATE ROLE shipper;

GRANT ALL PRIVILEGES ON AmaKart.* TO dbadmin;

-- Make sure that any view on which a role gets access on
should have the filter "SELECT user()"
GRANT ALL PRIVILEGES ON AmaKart.customer_add TO customer;
GRANT SELECT ON AmaKart.previousOrders TO customer;
GRANT SELECT ON AmaKart.listOrders TO customer;
GRANT SELECT ON AmaKart.packageStatus TO customer;

GRANT SELECT ON AmaKart.sellerProducts TO seller;
GRANT SELECT ON AmaKart.sellerOrders TO seller;

GRANT SELECT ON AmaKart.shipperTrack TO shipper;
```

## 6.3 Triggers

```
-- When a product is sold, we want to mention its
selling_price as later the seller can update the price
DELIMITER //
CREATE TRIGGER setPrice BEFORE INSERT on product_order
FOR EACH ROW BEGIN
    SET NEW.selling_price = (SELECT price FROM product WHERE
        product_id = NEW.product_id and seller_id = NEW.seller_id);
END//
DELIMITER ;

-- When a customer passes a rating for product we have to
update it in our product table
DELIMITER //
CREATE TRIGGER updateRatingProduct AFTER UPDATE on
product_order
FOR EACH ROW BEGIN
    IF NEW.product_rating != NULL THEN
        UPDATE product SET rating = (SELECT AVG(product_rating)
            FROM product_order WHERE product_id = NEW.product_id) WHERE
            product_id = NEW.product_id;
    END IF;
END//
DELIMITER ;

-- When a customer passes a rating for seller we have to
update it in our seller table
DELIMITER //
CREATE TRIGGER updateRatingSeller AFTER UPDATE on product_order
FOR EACH ROW BEGIN
    IF NEW.seller_rating != NULL THEN
        UPDATE seller SET rating = (SELECT AVG(seller_rating) FROM
            product_order WHERE seller_id = NEW.seller_id) WHERE
            seller_id = NEW.seller_id;
    END IF;
END//
DELIMITER ;
```

```
-- When a product is sold, we need to add an entry to our
track table for the same
DELIMITER //
CREATE TRIGGER addTrack BEFORE INSERT on product_order
FOR EACH ROW BEGIN
    INSERT INTO track () Values ();
    SET NEW.ship_index = (SELECT MAX (index_) FROM track);
END//
DELIMITER ;
```

## 7 Functions And Procedures

Below you can see details description and implementation of various Procedures and Functions, due to the nature of operations mentioned in our requirements we have preferred procedures in most cases over functions, thus if we have used function, we explicitly mention it.

### 7.1 Customer Procedures

1. Procedure to see purchases between some duration, "seePurchases-ByDate(IN startTime TIMESTAMP, IN endTime TIMESTAMP)" -> Will return complete details from order, payment, product and product\_order table.
2. Procedure to see items in cart, "getProductsFromCart ()" -> which selects everything from view "showCart".
3. Procedure for customer to checkout items present in cart, "purchaseEverythingInCart(IN oid varchar(20))" -> which takes in order id and then for each product present in cart, will add the corresponding entry to product\_order table.
4. Procedure for customer to remove product from cart, "removeProduct-Cart(IN pid varchar(20), IN sid varchar(20))" -> which takes in that products product\_id and seller\_id and thus delete such product from cart (deletion is performed using view showCart).
5. Procedure for customer to update a product in cart (i.e. change quantity of the chosen product), "CREATE PROCEDURE updateProduct-Cart(IN pid varchar(20), IN sid varchar(20), IN N INT)".
6. Procedure for customer to make an order, "makeorder(IN cnum varchar(20), IN badd varchar(20), IN cid varchar(20), IN sadd varchar(20))" which takes a credit card number, "cnum", billing address, "badd", customer ID, "cid" and a shipping address, "sadd" and first adds an entry into payment table by selecting date using "NOW()" function and then since payment table had payment\_id which was set to automatic increment so we after fetching it insert the corresponding entries into order\_table and then again by fetching order\_id as it was auto increment, we call already discussed procedure, "purchaseEverythingInCart".

7. Procedure to add product to cart, "addProductToCart(IN cid varchar(20), IN pid varchar(20), IN sid varchar(20), IN q int)" which will decide whether to update the quantity if the product is already there in cart, or to add this new record into our cart.
8. Procedure to see latest N purchases, "seeLatestNPurchases(IN N INT)" which takes in N and then gives rows corresponding to payment, order\_, product\_order, product, table and track (by taking join) ordered decreasingly by payment date.
9. Procedure to see Purchases between dates, "seePurchasesByDate(IN startTime TIMESTAMP, IN endTime TIMESTAMP)" -> works same as before just that it gives entries between the given time.
10. Procedure to see products within price range, "queryProductsTim(IN productName varchar(20), IN lowRange FLOAT, IN highRange FLOAT)" -> which after receiving suitable parameters, return entries from product sorted by price between the given range.
11. Procedure to see reviews of a given product, "ProductReviews(IN pid varchar(20), IN sid varchar(20))" -> will fetch ratings and reviews of the given product from product\_order, etc.
12. Procedure to add review for a product, "addReviewProduct(IN pid varchar(20), IN oid varchar(20), IN sid varchar(20), IN rev varchar(60))" -> as only those who have purchased successfully the product can add reviews, we need order ID for verification.
13. Procedure to add review for a seller, "addReviewSeller(IN pid varchar(20), IN oid varchar(20), IN sid varchar(20), IN rev varchar(60))" -> again we need order ID for verification.
14. Procedure to add rating for product, "addRatingProduct(IN pid varchar(20), IN oid varchar(20), IN sid varchar(20), IN rating INT)".
15. Procedure to add rating for seller, "addRatingSeller(IN pid varchar(20), IN oid varchar(20), IN sid varchar(20), IN rating INT)".
16. Procedure to see products sorted by rating, "queryProductsRat(IN productName varchar(20))".
17. Procedure to update customer info, custUpdateInfo(IN customer\_id varchar(20), IN password VARCHAR(20), IN name varchar(20), IN address VARCHAR(60), IN phone\_number DECIMAL(10) UNSIGNED, IN emailId VARCHAR(20)).



## 7.2 Seller Procedures

1. **Seller Function:** Function to return the total earning of a seller between supplied dates, "sellerStatsBetweenDate(startTime TIMESTAMP, endTime TIMESTAMP)".
2. Procedure for seller to see sold but not shipped products, soldButNotShipped(IN seller\_id varchar(20)) -> fetches entries from product\_order table which have corresponding shipper\_id NULL.
3. Procedure for seller to ship a sold product, "shipSoldProduct(IN gproduct\_id varchar(20), IN gorder\_id varchar(20), IN gseller\_id varchar(20), IN gshipper\_id varchar(20), IN gtracking\_id varchar(20), IN gdate DATE)" -> will ship the product (update shipper\_id and tracking\_id in track table) by taking relevant input.
4. Procedure for seller to see his rating, "getRating(IN seller\_id varchar(20))".
5. Procedure for seller to check whether there is already a product with given seller\_id and product\_id (this is useful when seller is adding a new product), "sellerCheckExistProd(IN product\_id varchar(20), IN seller\_id varchar(20))" -> will return a row if there is a corresponding product.
6. Procedure for seller to add new product, "addProduct(IN product\_id varchar(20), IN seller\_id varchar(20), IN product\_name varchar(20), IN product\_image varchar(300), IN price float, IN total\_stock int, IN pickup\_address varchar(60), IN description varchar(60))" -> adds an entry in product table by taking suitable input details.
7. Procedure for seller to update his specific product details, "updateProductInfo(IN product\_id varchar(20), IN seller\_id varchar(20), IN product\_name varchar(20), IN product\_image varchar(300), IN price float, IN total\_stock int, IN pickup\_address varchar(60), IN description varchar(60))" -> Note that it is possible to call this procedure by giving empty string for entries whose update is not deemed fit.
8. Procedure to update seller's info, "sellerUpdateInfo(IN seller\_id varchar(20), IN passwordd VARCHAR(20), IN named varchar(20), IN addressd VARCHAR(60), IN phone\_number DECIMAL(10) UNSIGNED, IN email\_id VARCHAR(20))" -> Note that it is possible to call this procedure by giving empty string for entries whose update is not deemed fit.

9. Procedure for seller to see his or her past sold products within a specific time duration, "seeSellingsBetweenDuration(IN startTime TIMESTAMP, IN endTime TIMESTAMP)" -> fetches the relevant entries from product\_order table.
10. Procedure to see latest N sellings, "seeLatestNSellings(IN N INT)" -> fetches the relevant entries from product\_order table.
11. Procedure to see seller's already listed similar products with increasing price, "selQuerySimProducts(IN productName varchar(20))".
12. Procedure to see seller's similar products sorted by rating, "selQueryProductsRat(IN productName varchar(20))".

### 7.3 Shipper Procedures

1. Procedure to update shipper's info, "shipperUpdateInfo(IN shipper\_id varchar(20), IN passwordd VARCHAR(20), IN named varchar(20), IN addressd VARCHAR(60), IN phone\_number DECIMAL(10) UNSIGNED, IN email\_id VARCHAR(20))" -> works in the same way as other user's update info.
2. Procedure for shipper to see his or her past shipments within a specific time duration, "seeShipmentsBetweenDuration(IN startTime DATE, IN endTime DATE)" -> fetches targeted entries from track table.
3. Procedure to see latest N Shipments, "seeLatestNShipments(IN N INT)" -> fetched targeted entries from track table.

## 8 Use Cases

Lets for this case assume that we a supplier. We first want to register and then add products to sell.

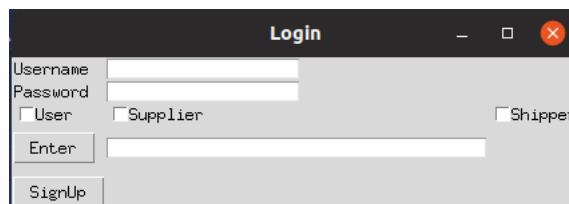
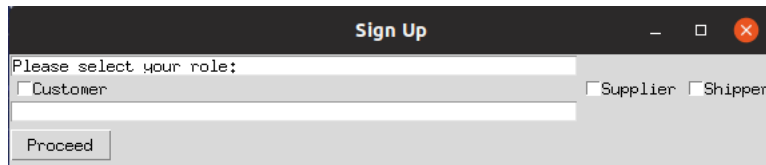


Figure 3: Login page.

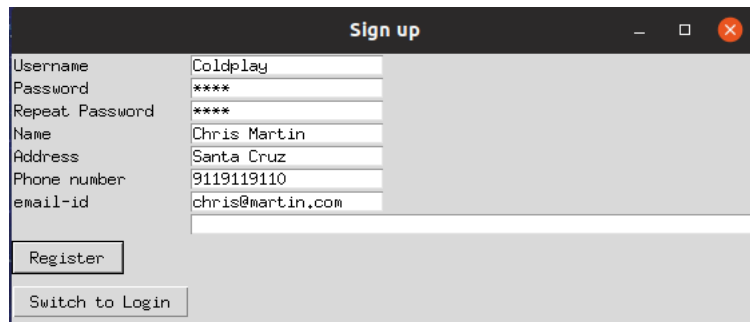
So at the login page we will select SignUp. It will then open up the SignUp page.



The screenshot shows a web browser window titled "Sign Up". Inside the window, there is a label "Please select your role:" followed by three radio button options: "Customer", "Supplier", and "Shipper". The "Supplier" option is selected. Below the options is a "Proceed" button.

Figure 4: SignUp page(Selecting role).

Since we are Supplier we will select supplier option and then proceed.



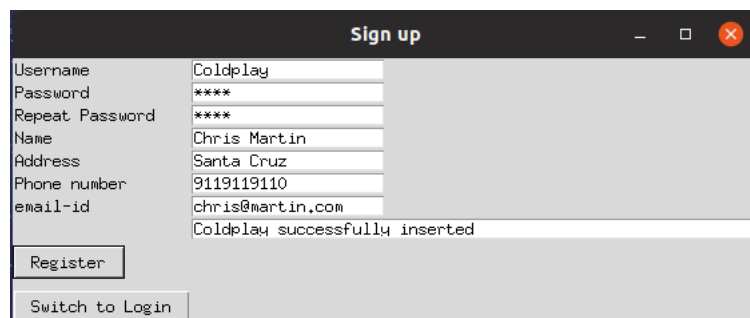
The screenshot shows the "Sign up" page with the following details entered in the form fields:

Field	Value
Username	Coldplay
Password	****
Repeat Password	****
Name	Chris Martin
Address	Santa Cruz
Phone number	9119119110
email-id	chris@martin.com

Below the form fields are two buttons: "Register" and "Switch to Login".

Figure 5: SignUp page(Entering Details).

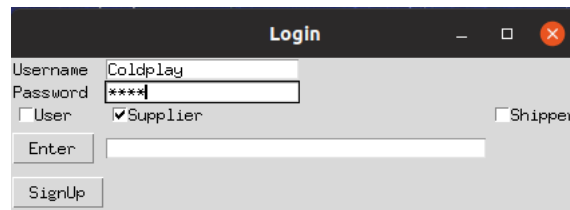
After entering the details press the Register button. It will display that the user is successfully created.



The screenshot shows the "Sign up" page with the same details as Figure 5. Below the form fields, a message "Coldplay successfully inserted" is displayed. The "Register" and "Switch to Login" buttons are still visible.

Figure 6: SignUp page(Conformation).

After registering goto the login page by pressing the Switch to Login button.

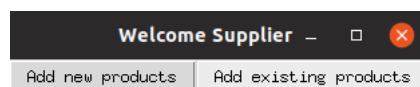


A screenshot of a web application's login page. The page has a dark header with the title "Login" and standard window controls. Below the header, there are input fields for "Username" (containing "Coldplay") and "Password" (containing "\*\*\*\*"). There are three radio buttons: "User" (unchecked), "Supplier" (checked), and "Shipper" (unchecked). Below the radio buttons are two buttons: "Enter" and "SignUp".

Figure 7: Login page(Entering Details).

After login you will be taken to a welcome page which will have two options for you

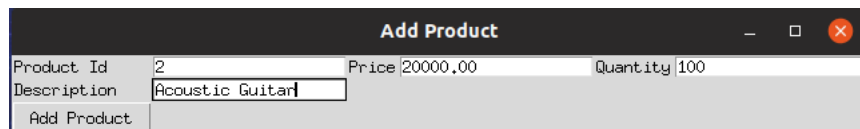
- Either to add new products in the market.
- Or to change the quantites or price of existing products.



A screenshot of a web application's "Welcome Supplier" page. The page has a dark header with the title "Welcome Supplier" and standard window controls. Below the header, there are two buttons: "Add new products" and "Add existing products".

Figure 8: Welcome page for Supplier.

So lets add a new product. You will be taken to a page which will ask to fill the details of the product.



A screenshot of a web application's "Add Product" page. The page has a dark header with the title "Add Product" and standard window controls. Below the header, there are four input fields: "Product Id" (containing "2"), "Price" (containing "20000,00"), "Quantity" (containing "100"), and "Description" (containing "Acoustic Guitar"). Below the input fields is a button labeled "Add Product".

Figure 9: Add new product.

Now the product has been added successfully.

## 9 Progress And TODOS

### 9.1 GUI

- Signup is working perfectly (just that inplace of integer if string is given then its an issue, can write a function to check it though).
- Login is working perfectly.
- 

## 10 Useful links

Complete project - [Link](#)  
GUI source code - [Link](#)