

Database Design And Implementation For E-Commerce

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

March 27, 2019

Contents

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

9 Use Cases	16
10 Progress And TODOS	20
10.1 GUI	20
11 Useful links	20

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.

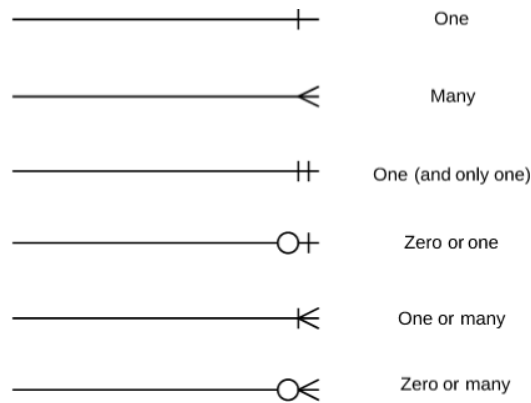


Figure 1: Entity Relation Diagram Notation

Following this notation, our ERD is represented below.

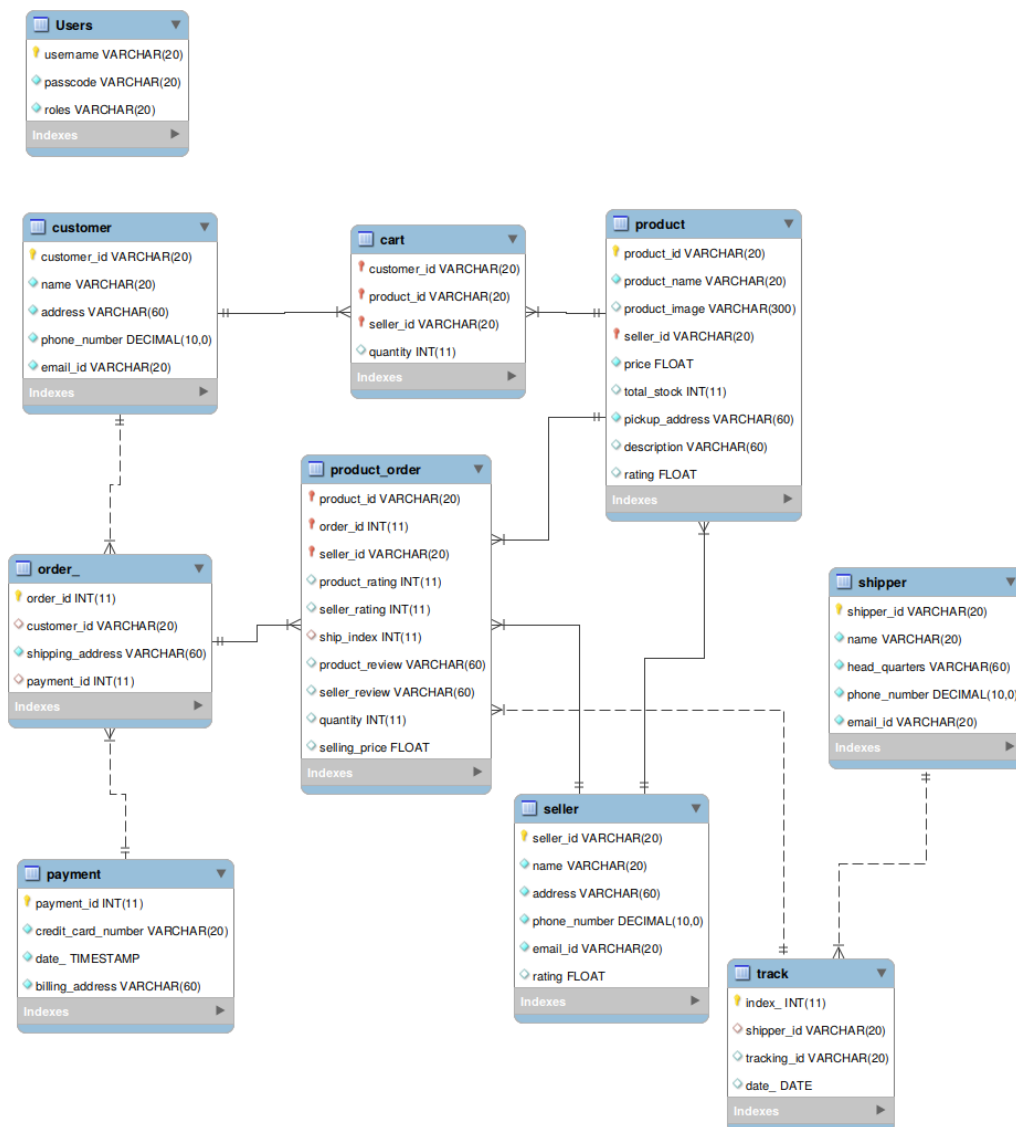


Figure 2: Entity Relationship Diagram for e-commerce

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). All sections except this don't have code but we feel it is necessary to include code in this section for better understanding of various relations.

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,
    product_image varchar(300),
    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 int AUTO_INCREMENT 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 */
/* Initially payment id can be null and then later once
the customer does the payment, trigger will add the
payment id */
create table order_ (
    order_id int AUTO_INCREMENT primary key not null,
    customer_id VARCHAR (20),
    shipping_address varchar(60) not null,
    payment_id int,
    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 *order_id* (by successful completion of 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 int 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),
    date_ DATE,
    foreign key (shipper_id) references shipper (shipper_id)
    on delete set null
);

```

- A relation for *cart*.

```

/* Here: (customer_id, product_id, seller_id) -> R is
the only relevant dependency and hence it is in BCNF */
create table cart (
    customer_id varchar(20),
    product_id varchar(20),
    seller_id varchar(20),
    quantity int,
    primary key (customer_id,product_id,seller_id),
    foreign key (customer_id) references
    customer(customer_id) on delete cascade,
    foreign key (product_id,seller_id) references
    product(product_id,seller_id) on delete cascade
);

```

- 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.

```

-/* Here: username -> R is the only relevant dependency
and hence it 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, Views and Triggers

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:

6.3 Triggers

1. When a product is sold, we want to mention its selling_price as later the seller can update the price, "setPrice". (sets NEW.selling_price entry in product_order).
2. When a customer passes a rating for product we have to update it in our product table, "updateRatingProduct" (update is done by averaging over all ratings).
3. When a customer passes a rating for seller we have to update it in our seller table, "updateRatingSeller" (update is naturally done in "seller" relation by averaging over all ratings passed to our seller).
4. When a product is sold, we need to add an entry to our track table for the same and also we have to reduct the total_stock of that product offered by that seller. Both of these are done together by our trigger, "stockCheckandaddTrack" (naturally it will have shipper_id, tracking_id field NULL as of now and this info will be used to determine whether the product has been shipped or not)
5. To check whether the products added in cart are valid or not. In case it is valid we proceed else we raise an exception. Trigger name "validCartinsert".

6. Same as above but for update operation on cart, "validCartupdate".

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

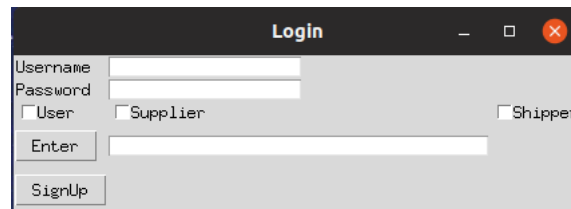
1. Procedure to update shipper's info, "shipperUpdateInfo(IN shipper_id varchar(20), IN passwordd VARCHAR(20), IN named varchar(20), IN addressd VARCHAR(60), IN phone_number DECIMAL(10) UNSIGNED, IN email_id VARCHAR(20))" -> works in the same way as other user's update info.
2. Procedure for shipper 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 Further Reading

- Complete implementation of schema, views, procedures, triggers, etc. can be found here.

9 Use Cases

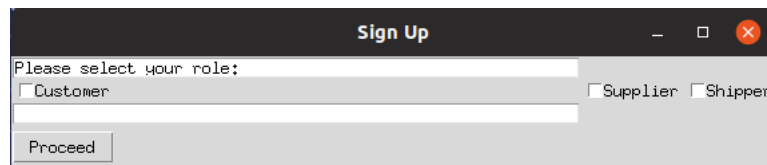
Lets for this case assume that we a supplier. We first want to register and then add products to sell.



A screenshot of a web application window titled "Login". It contains a form with the following elements: a "Username" text input field, a "Password" text input field, three radio buttons labeled "User", "Supplier", and "Shipper", an "Enter" button, and a "SignUp" button.

Figure 3: Login page.

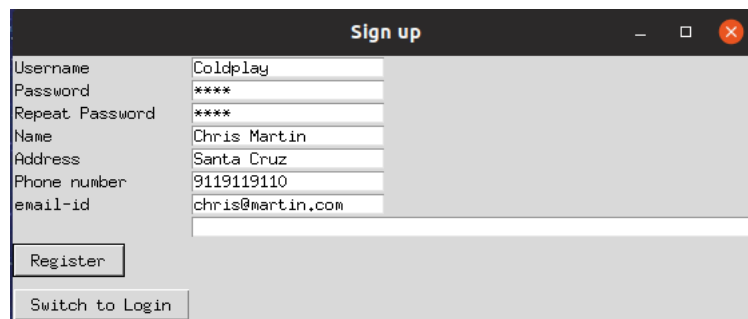
So at the login page we will select SignUp. It will then open up the SignUp page.



A screenshot of a web application window titled "Sign Up". It contains a form with the following elements: a label "Please select your role:", three radio buttons labeled "Customer", "Supplier", and "Shipper", a text input field, and a "Proceed" button.

Figure 4: SignUp page(Selecting role).

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



A screenshot of a web application window titled "Sign up". It contains a form with the following elements: a "Username" text input field with the value "Coldplay", a "Password" text input field with the value "****", a "Repeat Password" text input field with the value "****", a "Name" text input field with the value "Chris Martin", an "Address" text input field with the value "Santa Cruz", a "Phone number" text input field with the value "9119119110", an "email-id" text input field with the value "chris@martin.com", a "Register" button, and a "Switch to Login" button.

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 a 'Sign up' window with the following fields and values:

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

Below the fields, a message states: 'Coldplay successfully inserted'. At the bottom, there are two buttons: 'Register' and 'Switch to Login'.

Figure 6: SignUp page(Conformation).

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

The screenshot shows a 'Login' window with the following fields and values:

Username	Coldplay
Password	****
<input type="checkbox"/> User <input checked="" type="checkbox"/> Supplier <input type="checkbox"/> Shipper	

Below the fields, there are two buttons: 'Enter' and 'SignUp'.

Figure 7: Login page(Entering Details).

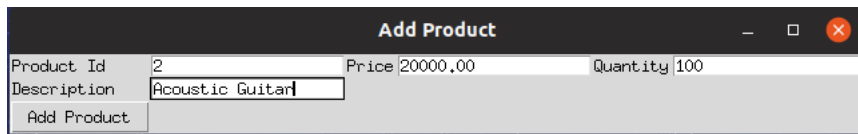
After login you will be taken to a welcome page which will have two options for you

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

The screenshot shows a 'Welcome Supplier' window with 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.



Add Product			
Product Id	2	Price	20000.00
		Quantity	100
Description	Acoustic Guitar		
Add Product			

Figure 9: Add new product.

Now the product has been added successfully.

10 Progress And TODOS

10.1 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.
-

11 Useful links

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