Jose Miranda
Sabrina Kundu
Reagan Zhou
Professor Duke
10/12/19
CMSC 508

Phase 2 Project Proposal

GitHub link:
https://github.com/cmsc-vcu/semester-project-team-15/tree/master

Sell-Or-Buy Database

The database will be used for an application that allows users to buy and sell items and the items will be shipped to their location. The database stores users' information and product information which will be viewed by consumers. This provides an ecommerce solution that other companies would not have. Sellers are able to post products, which will be inserted into the database. Buyers can order these items. Once the order is processed, the details will be inserted into the database. For example, a seller may want to post a clothing item, which will then be inserted into the database including details about the item. In turn, this will be shown to the consumer.

Accounts are created and stored in the database as either a Customer, Delivery Driver, or Seller. Accounts will contain the credentials of the user, which will help identify them by their unique ID. User's will be able to access their account to post, order and view deliveries.

Consumers can place orders which will then be stored in the database. Consumers are able to view their orders and order history. They will also be able to modify or remove orders. The application is used for users that do not want to leave their house to buy products. Rather than buying something in person, users will be able to work on other things while the products they are trying to buy ships easily to their homes.

Sellers are able to sell as a company or independently. Sellers post their products onto the data base, which will then be viewable by consumers for purchase. The application lets companies and independent sellers to easily have contact with their customers. The sellers would not have to create their own website or delivery service. Sellers will be able to post their products and have them easily purchased by customers. This could help companies that are just starting up that want to get a head start with selling their products.

Products are posted by sellers onto the database. These products are modifiable by the seller and viewable by the consumer. Details of the product will be shown when entered by the seller. Some examples of products sold are clothing, electronics, and furniture. Non-perishable items

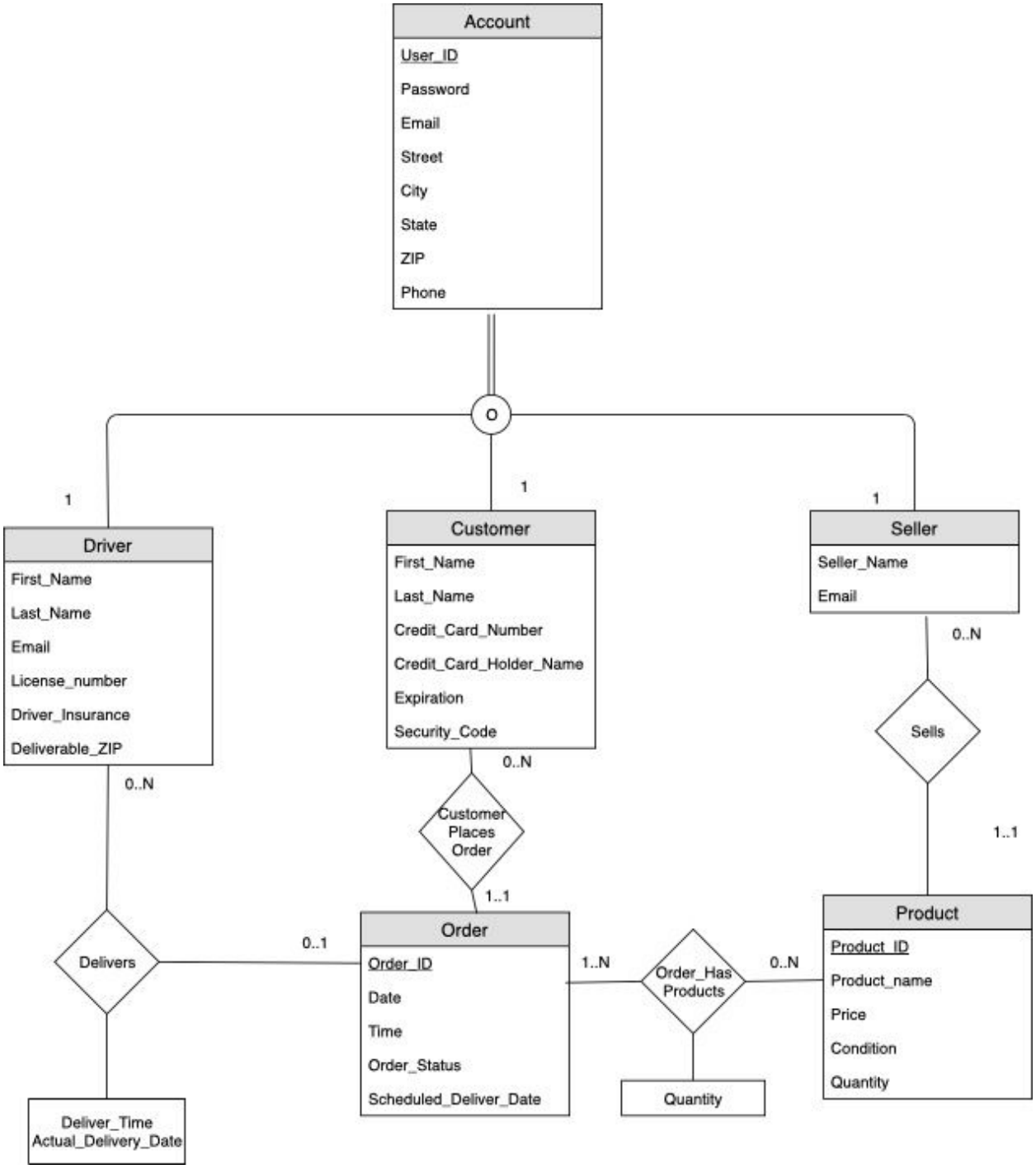
are not available for sale. Products may be used or new. The prices of the products are decided by the seller.

Orders are posted by the customers onto the database when they add products to their order. Products contained in an order will have a quantity value if a user wants to order multiple of the same product. The products are removable from the order and is also able to be empty. Customers will be able to cancel orders within 24 hours of the order being placed.

The delivery service ships the products. The customers are able to view the delivery service name and time of arrival. Customers are able to choose whether they prefer standard or express shipping. This gives customers an easy access to delivery services and also if they want their product faster. Customers will be able to view the status of their order whether it is shipped or delivered.

Security has been a focus and we implemented verification of a valid email and valid password when prompting users to create accounts to prevent SQL injections. This prevents users from dropping tables or altering the database in a way that is unintended. Also to provide further security we check data types before inserting to make sure it matches the specified column's data type. We utilized the MySQLi extension that allows us to connect to the database and also gave us access to functions to interact with the database when it came to inserting, updating, or interacting with the database in any way.

Entity-relationship diagram



Logical Design

Primary Keys appear like this → Primary Keys
Attributes that are primary keys and foreign keys appear like this →Primary_keys/Foreign_Keys

Account(User_Id, Password, Email, Street, City, Phone, State, ZIP)
- Functional Dependency: User_ID → Password, Email, Street, ZIP, Phone

Driver (User_ID, First_Name, Last_Name, email ,License_Number,Driver_Insurance, Deliverable_ZIP)
- Functional Dependency: User_ID → First_Name, Last_Name, email, License_Number, Driver_Insurance, Deliverable_ZIP

Seller(User_ID, Seller_Name, email)
- Functional Dependency: User_ID → Seller_Name, email

Customer(User_ID, Credit_Card_Number, Credit_Card_Holder_Name, Expiaration, Security_Code)
- Functional Dependency: User_ID → Credit_Card_Number, Credit_Card_Holder_Name, Expiration, Security_Code

Product(Product_ID, Product_Name, Price, Condition, Quantity)
- Functional Dependency: Product_ID → Product_Name, Price, Condition, Quantity

Order(Order_ID, User_ID, Date, Time, Order_Status, Scheduled_Delivery_Date)
- Functional Dependency: Order_ID → User_ID, Date, Time, Order_Status, Scheduled_Delivery_Date

Order_Products(Order_ID, Product_ID, Quantity)
- Functional Dependency: Order_ID, Product_ID → Quantity

Delivers(Delivery_Drive.User_ID, Order_ID, Actual_Delivery_Date, Delivery_TIme)
- Functional Dependency: Deliver_Driver.User_ID, Order_ID → Actual_Delivery_Date, Delivery_Time

Order was decomposed to Order and Order_Products to satisfy 4NF because multiple different products in an order could be associated with an account which causes a multivalued dependency.

SQL Script

```
CREATE TABLE accounts(
  user_id int not null AUTO_INCREMENT,
  pass_word varchar(225) not null,
  email varchar(225) not null,
  street varchar(225) not null,
  city varchar(225) not null,
  state varchar(225) not null,
  zip double(5,0) not null,
  phone double(10,0) not null,
  status_type varchar(225) not null,
  primary key(user_id),
  foreign key (status_type) references acc_status(status_type)
);

CREATE TABLE seller(
  user_id int not null primary key,
  email varchar(30) not null unique,
  seller_name varchar(30)not null,
  foreign key(user_id) references accounts(user_id)
);

CREATE TABLE driver(
  user_id int not null primary key,
  email varchar(30) not null unique,
  first_name varchar(30) not null,
  last_name varchar(30) not null,
  license_num double(30,0) not null,
  insurance varchar(30) not null,
  deliver_zip double(5,0) not null,
  foreign key(user_id) references person(user_id)
);

CREATE TABLE customer(
  user_id int not null primary key,
  first_name varchar(30) not null,
  last_name varchar(30) not null,
  credit_card double(9,0) not null,
  card_holder varchar(30) not null,
  expirtion varchar(30) not null,
  security_code double(5,0) not null,
  foreign key(user_id) references accounts(user_id)
```

```
);

CREATE TABLE product(
  product_id int not null AUTO_INCREMENT,
  user_id int not null,
  product_name varchar(30) not null,
  price double(30,2) not null,
  conditions varchar(30) not null,
  primary key(product_id),
  foreign key(user_id) references accounts(user_id)
);

CREATE TABLE orders(
  order_id int not null AUTO_INCREMENT,
  user_id int not null,
  product_id int not null,
  order_date date not null,
  order_status varchar(30),
  schduled_delivery varchar(30),
  primary key(order_id, user_id, product_id),
  foreign key(user_id) references accounts(user_id),
  foreign key(product_id) references product(product_id)
);

CREATE TABLE orders_products(
  order_id int not null,
  product_id int not null,
  quantity int not null,
  primary key(order_id, product_id),
  foreign key(product_id) references product(product_id),
  foreign key(order_id) references orders(order_id)
)

CREATE TABLE delivers(
  order_id int not null,
  user_id int not null,
  delivery_time double(4,0),
  actual_delivery_date varchar(30),
  foreign key(order_id) references orders(order_id),
  foreign key(user_id) references driver(user_id)
);

delimiter //
```

```
CREATE TRIGGER `trig_quantity_check` BEFORE INSERT ON `product`
FOR EACH ROW
BEGIN
IF NEW.quantity < 0 THEN
SET NEW.quantity=0;
END IF;
END
//
delimiter;
```