E-Commerce: Online Grocery Delivery
INFSCI 2710: Database Management Group Project
Fall 2016

Erin Price (eep27)
Xiaoshi Guo (XIG40)
Zhou Zhu (zhou_zhu)

Introduction to the System

A short overview of the system including identification of the various types of users, administrators, etc. who will be accessing the system in various ways.

Overview

The website offers a service to have groceries delivered to a customer's address. In order for this service to function, the system must have various users and permissions. The system has four main roles including customer, store/region manager, salesperson, and admin. As an overview of how the system works with these user roles, the first page to the system is a login screen. According to user name and user role which are stored in the user table, user can be redirected log in to the appropriate screen. For example, if the user's role is a customer, then they will be taken to a front-end view where they can do tasks such as browse and purchase groceries. However, if the user's role is a store/region manager, salesperson, or admin, then they will be redirected to a back-end view. These views are discussed later in the report. Each role needs to be broken down into specific permissions.

Customer Role

Generally, a user is considered a customer and is given customer permissions after they sign up to the delivery service. A customer is able to create, view, and edit their own profile with fields that include first name, last name, address, email, phone number, and more. Then the user can browse the database of products that the delivery service offers through the front-end. The navigation is based on the product categories. Customers can search and view products in the system by name, category, description, keywords, price, brand, rating, and promotion. They can also see a detailed page for each product, including the product's name, description, rating, price, brand, and category.

After browsing the website and seeing what products are available, the customer can use the shopping cart. The shopping cart allows the customer to add and delete goods into their digital cart. They can also change the quantity of the product from the shopping cart page. After the customer shops on the website, they have the ability to make the order by confirming to buy the goods in the cart. There is a check-out process they go through, and then they have permissions to browse their own transactions in the transaction history. Finally, the customer can add ratings to the products so that their shopping experience is tailored to their favorite and least favorite purchases.

Store/Region Manager Role

The goal of the website is to deliver groceries to a customer's door, so there must be a local warehouse/store that is managed. Each store will have a store manager to manage the store’s products as well as view the transactions related to the store. Similarly, location will need a region manager to maintain stock, transactions, products and more. Store managers and region managers can see the back-end view of the website in order to manage their stores’ warehouse and deliveries. They can make changes to the product information, inventory, customer, transaction, and salesperson (delivery person).

Salesperson Role

A salesperson for this application is a delivery person. This person will deliver groceries from the warehouse to the customer's door. They can log into the back-end view of the website to browse the transactions and products. A salesperson can also view certain reports, specifically about their individual sales.

Admin Role

The admin role is meant for those who oversee the entire business system, including database. They can add, edit, and delete every aspect to the grocery delivery business website. When they login, admins see the back-end and can approve managers' accounts as well as have all access to inventory, customers, transactions, salespersons, and reports.

Assumptions About the System

A list of assumptions that you have made about the system.

- Passwords will be stored in plaintext instead of encrypted values due to time constraints.
- Payment methods such as credit cards and electronic checks will not actually be real data due to time constraints. There are PCI security standards that would have to be implemented as well as setting up a bank account and payment processor.
- The site would use SSL in a real setting.
- Users will not be willing to give certain data such as income or marriage status on a grocery delivery website.
- Regions are states. Stores are cities.
- A city name can determine the only unique state.
- A zip code can determine the only unique street.
- Different city may have the same zip code.
- States, cities, zip codes and streets will be managed by admin and managers, customers can only choose one of them, they cannot add new instance in these information.

Graphical Schema using the ER Diagram

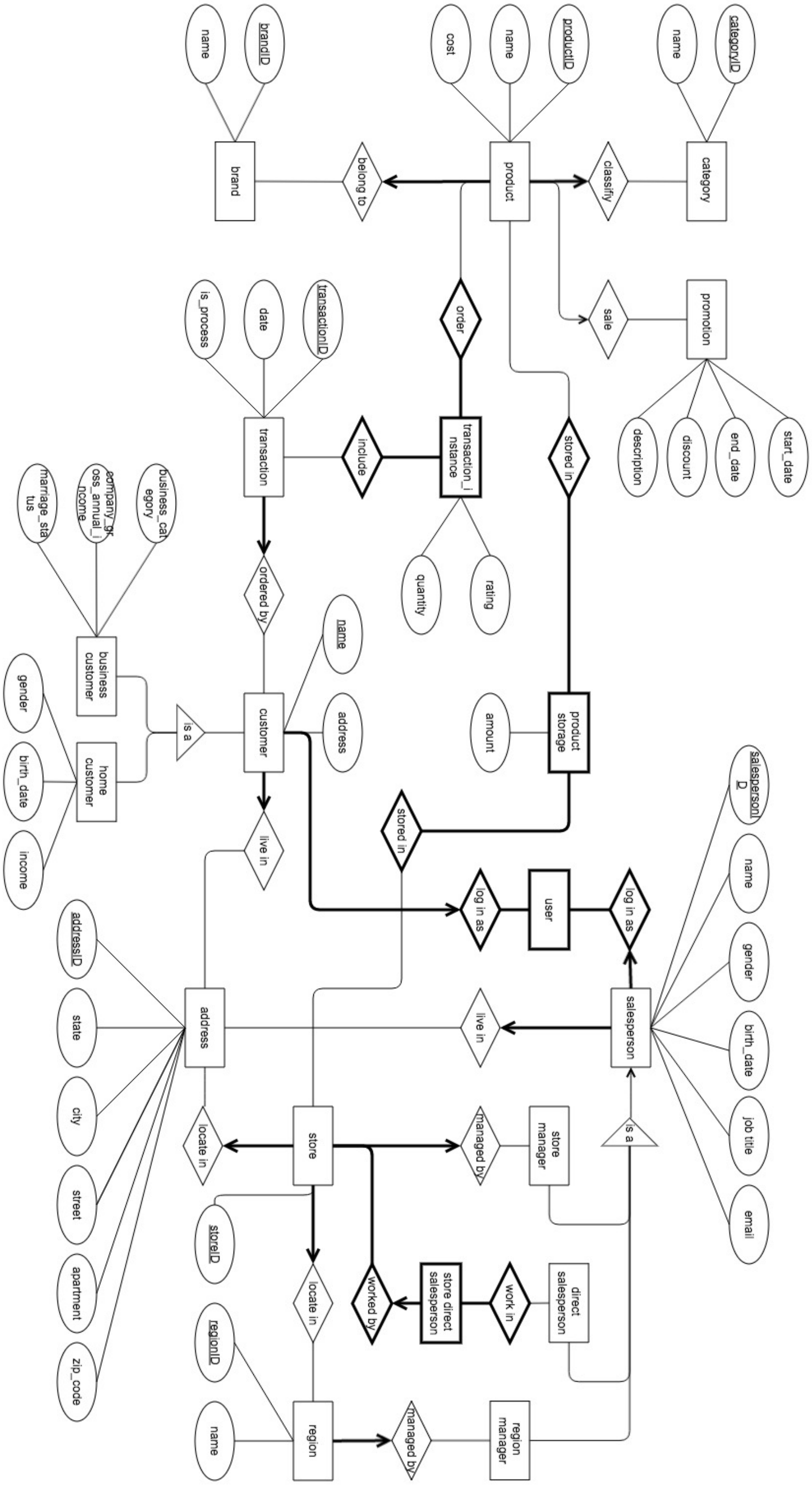
A graphical schema of the database using the ER diagram with a short description of each entity set, relationship set and their corresponding attributes.

ER Diagram
See next page

Description

There are 19 entities in the ER diagram, which shows relationships between all of those entities included in the grocery business. All relationships depend on the business assumption, including connections required by business, by logic and by common sense. There is a special relation, namely “is-a”, which denotes that some of entities are children of a certain entity, for example, store manager, direct salesperson and region manager are all salesperson. Additionally, the link with arrow means that every certain tuple in the entity in that relationship could only shows at most one time, and the bold link means very tuple in the entity should appear in that relationship.

Besides entities and relationships, there are attributes for every entity to describe its properties. The properties with underlines are primary keys to the certain entity.


brandID
name
brand
belong to
productID
name
cost
product
classify
category
categoryID
name
order
sale
promotion
start_date
end_date
discount
description
stored in
transactionID
date
is_process
transaction
include
transaction_instance
rating
quantity
business_category
company_gross_annual_income
marriage_status
business customer
ordered by
name
address
customer
is a
home customer
gender
birth_date
income
product storage
amount
stored in
live in
log in as
user
log in as
salespersonID
name
gender
birth_date
job title
email
salesperson
live in
is a
addressID
state
city
street
apartment
zip_code
address
locate in
store
managed by
store manager
storeID
locate in
worked by
store direct salesperson
work in
direct salesperson
regionID
name
region
managed by
region manager

Relational Schema from the E-R Diagram

A set of relational schema resulting from the E-R diagram with identification of primary and foreign keys.

Relational Schema
See next page

Description

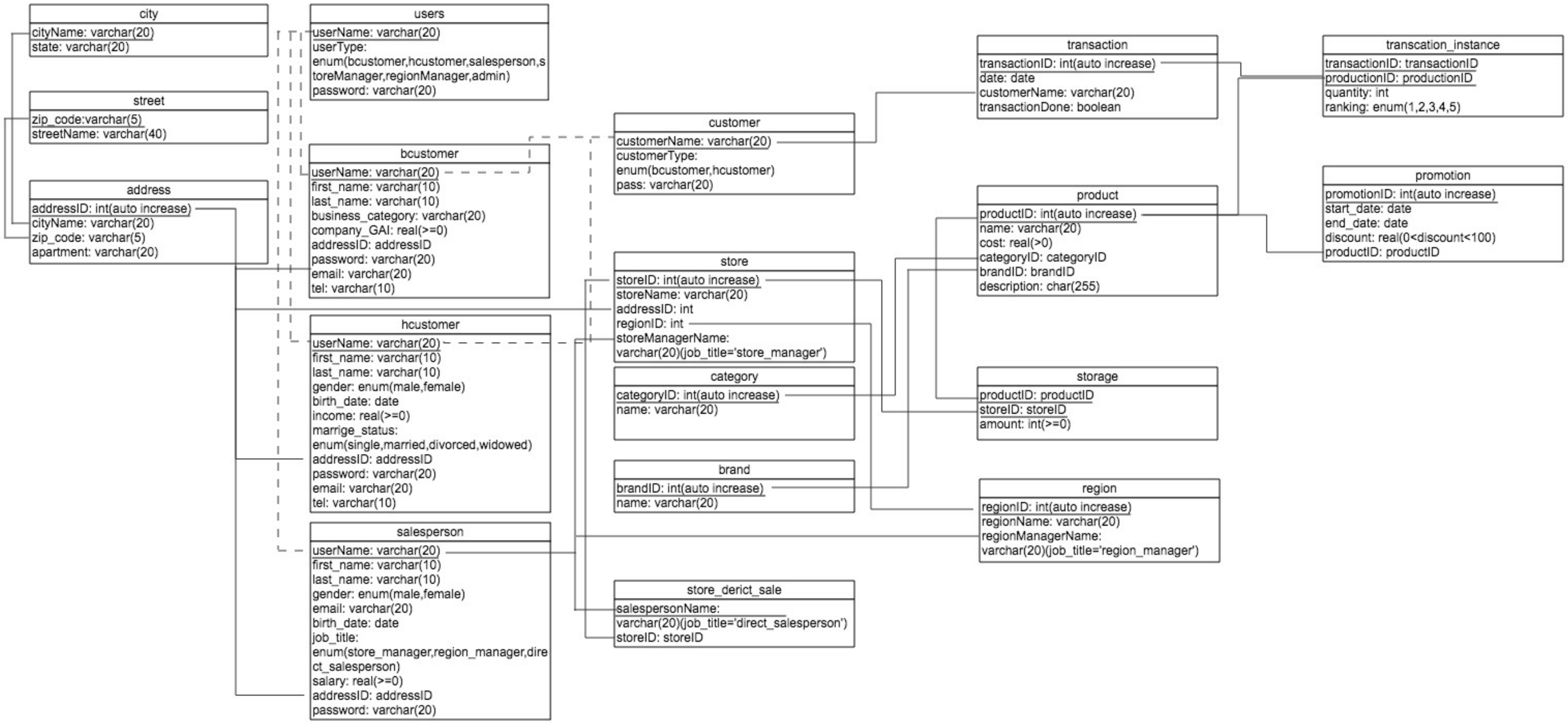
According to the ER diagram shown before, and use BCNF normal forms to refine the tables represent in ER, we get the relational schema as following. This relational schema is already the real table we build in our database.

Every entity always corresponds to a table in relational schema, but we take account of the BCNF normal forms. In terms of BCNF, some single tables are separated to several tables. For example, because some constrain in address which is one entity in ER diagram, we separated in to three tables, they are "city", "street" and address.

The relationships in ER diagram are descripted as foreign key between tables. Specially, there are some dotted links between tables' attributes, they are dummy foreign keys cannot implement by database's "foreign key" function, which should be one to one, but we realize that logical foreign key constrain by using triggers.

Furthermore, take real situation in business into account, we use username instead of userID as the primary in all the tables related to user.

The same as the ER diagram, the underline illustrates the primary key in that table.


city
cityName: varchar(20)
state: varchar(20)
street
zip_code:varchar(5)
streetName: varchar(40)
address
addressID: int(auto increase)
cityName: varchar(20)
zip_code: varchar(5)
apartment: varchar(20)
users
userName: varchar(20)
userType:
enum(bcustomer,hcustomer,salesperson,s
toreManager,regionManager,admin)
password: varchar(20)
bcustomer
userName: varchar(20)
first_name: varchar(10)
last_name: varchar(10)
business_category: varchar(20)
company_GAI: real(>=0)
addressID: addressID
password: varchar(20)
email: varchar(20)
tel: varchar(10)
hcustomer
userName: varchar(20)
first_name: varchar(10)
last_name: varchar(10)
gender: enum(male,female)
birth_date: date
income: real(>=0)
marrige_status:
enum(single,married,divorced,widowed)
addressID: addressID
password: varchar(20)
email: varchar(20)
tel: varchar(10)
salesperson
userName: varchar(20)
first_name: varchar(10)
last_name: varchar(10)
gender: enum(male,female)
email: varchar(20)
birth_date: date
job_title:
enum(store_manager,region_manager,dire
ct_salesperson)
salary: real(>=0)
addressID: addressID
password: varchar(20)
customer
customerName: varchar(20)
customerType:
enum(bcustomer,hcustomer)
pass: varchar(20)
store
storeID: int(auto increase)
storeName: varchar(20)
addressID: int
regionID: int
storeManagerName:
varchar(20)(job_title='store_manager')
category
categoryID: int(auto increase)
name: varchar(20)
brand
brandID: int(auto increase)
name: varchar(20)
store_derict_sale
salespersonName:
varchar(20)(job_title='direct_salesperson')
storeID: storeID
transaction
transactionID: int(auto increase)
date: date
customerName: varchar(20)
transactionDone: boolean
product
productID: int(auto increase)
name: varchar(20)
cost: real(>0)
categoryID: categoryID
brandID: brandID
description: char(255)
storage
productID: productID
storeID: storeID
amount: int(>=0)
region
regionID: int(auto increase)
regionName: varchar(20)
regionManagerName:
varchar(20)(job_title='region_manager')
transcation_instance
transactionID: transactionID
productionID: productionID
quantity: int
ranking: enum(1,2,3,4,5)
promotion
promotionID: int(auto increase)
start_date: date
end_date: date
discount: real(0<discount<100)
productID: productID

Refine Tables from ER Diagram to Relational Schema

1. address(addressID, state, city, street, apartment, zip_code)
addressID -> state
addressID -> city
city -> state
addressID -> street
addressID -> apartment
addressID -> zip_code
street -> zip_code

We seperate these address-related attributes to three tables:
1.1 address(addressID, cityID, street, apartment)
{addressID -> cityID, addressID -> street, addressID -> apartment}

For {addressID -> cityID, street, apartment}, addressID is the key, the relation is in BCNF.

1.2 city(cityID, cityName, state)
{cityID -> cityName, cityID -> state}

For {cityID -> cityName, state}, cityID is the key, the relation is in BCNF.

1.3 street(streetID, streetName, zip_code)
{streetID -> streetName, streetID -> zip_code}

For {streetID -> streetName, zip_code}, streetID is the key, the relation is in BCNF.

2. user(userID, userType, userTypeID, userName, password)
userID -> userType
userID -> userTypeID
userID -> userName
userID -> password
userName -> userTypeID
userName -> password

For {userID -> userType, userTypeID, userName, password, userName -> userType, password}, we need to seperate the table into two.

2.1 user_name(userID, userName)
userID -> userName
We can see that userID is the key, the relation is in BCNF.

2.2 user(userID, userType, userTypeID, password)
userID -> userType

userID -> userTypeID
userID -> password

For {userID -> userType, userTypeID, password}, userID is the key, the relation is in BCNF.

3. bcustomer(b_customerID, first_name, last_name, business_category, company_GAI, addressID)
b_customerID -> first_name
b_customerID -> last_name
b_customerID -> business_category
b_customerID -> company_GAI
b_customerID -> addressID

For {b_customerID -> first_name, last_name, business_category, company_GAI, addressID}, we know that b_customerID is the key, the relation is in BCNF.

4. hcustomer(h_customerID, first_name, last_name, gender, birth_date, income, marrige_status, addressID)
h_customerID -> first_name
h_customerID -> last_name
h_customerID -> gender
h_customerID -> birth_date
h_customerID -> income
h_customerID -> marrige_status
h_customerID -> addressID

For {h_customerID -> first_name, last_name, gender, birth_date, income, marrige_status, addressID}, h_customerID is the key, so the relation is in BCNF.

5. salesperson(salespersonID, first_name, last_name, gender, email, birth_date, job_title, salary, addressID)
salespersonID -> first_name
salespersonID -> last_name
salespersonID -> gender
salespersonID -> email
salespersonID -> birth_date
salespersonID -> job_title
salespersonID -> salary
salespersonID -> addressID

Similar to b_customer and h_customer, because {salespersonID -> first_name, last_name, gender, email, birth_date, job_title, salary, addressID}, we can see that salespersonID is the key, so the relation is in BCNF.

6. customer(customerID, customerType, typeID)
customerID -> customerType
customerID -> typeID

For {customerID -> customerType, typeID}, the customerID is the key, the relation is in BCNF.

7. store(storeID, addressID, regionID, salespersonID)
storeID -> addressID, regionID, salespersonID

We can see that storeID is the key, so the relation is in BCNF.

8. category(categoryID, name)
categoryID -> name

We can see that categoryID is the key, so the relation is in BCNF.

9. brand(brandID, name)
brandID -> name

We can see that brandID is the key, so the relation is in BCNF.

10. store_derict_sale(salespersonID, storeID)
salespersonID, storeID -> salespersonID, storeID

Both attributes in the relation compose the key, so the relation is in BCNF.

11. transaction(transactionID, date, customerID)
transactionID -> date
transactionID -> customerID

For transactionID is the key, the relation is in BCNF.

12. product(productID, name, cost, categoryID, brandID)
productID -> name
productID -> cost
productID -> categoryID
productID -> brandID

Obviously, productID is the key in the relation, so the relation is in BCNF.

13.storage(productID, storeID, amount)
productID, storeID -> amount
The set of (productID, storeID) is the key, so the relations is in BCNF.

14. region(regionID, name, salespersonID)
regionID -> name
regionID -> salespersonID

region ID is key, so the relation is in BCNF.

15. transaction_instance(transactionID, productionID, ranking)
transactionID -> productionID, ranking

transactionID is key, so the relation is in BCNF.

16. promotion(promotionID, start_date, end_date, discount, productID)
promotionID -> start_date
promotionID -> end_date
promotionID -> discount
promotionID -> productID

For {promotionID -> start_date, end_date, discount, productID}, we know that promotionID is the key, so the relation is in BCNF.

DDL Statements to Create the Relational Schema

The DDL statements to create the relational schema in some appropriate Normal Form, with identification and justification of the Normal Form.
Create tables according to diagrams

```
CREATE DATABASE grocery_website;
CREATE TABLE grocery_website.city(
  cityName VARCHAR(20) PRIMARY KEY,
  state VARCHAR(20)
);
CREATE TABLE grocery_website.street(
  zip_code VARCHAR(5) PRIMARY KEY,
  streetName VARCHAR(40)
);
CREATE TABLE grocery_website.address(
  addressID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  cityName VARCHAR(20),
  zip_code VARCHAR(5),
  apartment VARCHAR(20),
  FOREIGN KEY(cityName) REFERENCES grocery_website.city(cityName),
  FOREIGN KEY(zip_code) REFERENCES grocery_website.street(zip_code)
);
CREATE TABLE grocery_website.users(
  userName VARCHAR(20),
  userType ENUM(
    'bcustomer',
    'hcustomer',
    'salesperson',
    'store_manager',
    'region_manager',
    'direct_salesperson',
    'admin'
  ),
  pass VARCHAR(20),
  PRIMARY KEY(userName)
);
CREATE TABLE grocery_website.customer(
  customerName VARCHAR(20),
  customerType ENUM('bcustomer', 'hcustomer'),
  pass VARCHAR(20),
  PRIMARY KEY(customerName)
);
CREATE TABLE grocery_website.bcustomer(
  b_customerUsername VARCHAR(20),
  first_name CHAR(10),
  last_name CHAR(10),
  business_category CHAR(20),
  company_GAI REAL,
  addressID INT,
  pass VARCHAR(20),
  email VARCHAR(20),
  tel VARCHAR(10),
  PRIMARY KEY(b_customerUsername),
  FOREIGN KEY(addressID) REFERENCES grocery_website.address(addressID)
);
CREATE TABLE grocery_website.hcustomer(
  h_customerUsername VARCHAR(20),
  first_name CHAR(10),
  last_name CHAR(10),
  gender ENUM('male', 'female'),
  birth_date DATE,
  income REAL,
  marriage_status ENUM(
    'single',
    'married',
    'divorced',
    'widowed'
  ),
  addressID INT,
  pass VARCHAR(20),
  email VARCHAR(20),
  tel VARCHAR(10),
  PRIMARY KEY(h_customerUsername),
  FOREIGN KEY(addressID) REFERENCES grocery_website.address(addressID)
);
CREATE TABLE grocery_website.salesperson(
  salespersonName VARCHAR(20),
  first_name CHAR(10),
  last_name CHAR(10),
  gender ENUM('male', 'female'),
  email CHAR(20),
  birth_date DATE,
  job_title ENUM(
    'store_manager',
    'region_manager',
    'direct_salesperson'
  ),
  salary REAL,
  addressID INT,
  pass VARCHAR(20),
  PRIMARY KEY(salespersonName),
  FOREIGN KEY(addressID) REFERENCES grocery_website.address(addressID)
);
CREATE TRIGGER grocery_website.bcus_cus_trigger AFTER
INSERT
ON
  grocery_website.bcustomer FOR EACH ROW
INSERT
INTO
  grocery_website.customer
VALUES(
  NEW.b_customerUsername,
  'bcustomer',
  NEW.pass
);
CREATE TRIGGER grocery_website.hcus_cus_trigger AFTER
INSERT
ON
  grocery_website.hcustomer FOR EACH ROW
INSERT
INTO
  grocery_website.customer
VALUES(
  NEW.h_customerUsername,
  'hcustomer',
  NEW.pass
);
CREATE TRIGGER grocery_website.cus_usr_trigger AFTER
INSERT
ON
  grocery_website.customer FOR EACH ROW
INSERT
INTO
  grocery_website.users
VALUES(
  NEW.customerName,
  NEW.customerType,
  NEW.pass
);
CREATE TRIGGER grocery_website.sales_trigger AFTER
INSERT
```

```
ON
  grocery_website.salesperson FOR EACH ROW
INSERT
INTO
  grocery_website.users
VALUES(
  NEW.salespersonName,
  NEW.job_title,
  NEW.pass
);
CREATE TABLE grocery_website.region(
  regionID INT NOT NULL AUTO_INCREMENT,
  regionManagerName VARCHAR(20),
  regionName VARCHAR(20),
  PRIMARY KEY(regionID),
  FOREIGN KEY(regionManagerName) REFERENCES
grocery_website.salesperson(salespersonName)
);
CREATE TABLE grocery_website.store(
  storeID INT NOT NULL AUTO_INCREMENT,
  storeName VARCHAR(20),
  addressID INT,
  regionID INT,
  storeManagerName VARCHAR(20),
  PRIMARY KEY(storeID),
  FOREIGN KEY(storeManagerName) REFERENCES
grocery_website.salesperson(salespersonName),
  FOREIGN KEY(regionID) REFERENCES grocery_website.region(regionID)
);
CREATE TABLE grocery_website.category(
  categoryID INT NOT NULL AUTO_INCREMENT,
  gwcname CHAR(20),
  PRIMARY KEY(categoryID)
);
CREATE TABLE grocery_website.brand(
  brandID INT NOT NULL AUTO_INCREMENT,
  gwbname CHAR(20),
  PRIMARY KEY(brandID)
);
CREATE TABLE grocery_website.store_direct_sale(
  salespersonName VARCHAR(20),
  storeID INT,
  PRIMARY KEY(salespersonName, storeID),
  FOREIGN KEY(storeID) REFERENCES grocery_website.store(storeID)
```

```
);
CREATE TABLE grocery_website.transaction(
  transactionID INT NOT NULL AUTO_INCREMENT,
  DATE DATE,
  customerName VARCHAR(20),
  in_process BOOLEAN,
  PRIMARY KEY(transactionID),
  FOREIGN KEY(customerName) REFERENCES grocery_website.customer(customerName)
);
CREATE TABLE grocery_website.product(
  productID INT NOT NULL AUTO_INCREMENT,
  gwpname CHAR(20),
  cost REAL,
  categoryID INT,
  brandID INT,
  description CHAR(255),
  PRIMARY KEY(productID),
  FOREIGN KEY(categoryID) REFERENCES grocery_website.category(categoryID),
  FOREIGN KEY(brandID) REFERENCES grocery_website.brand(brandID)
);
CREATE TABLE grocery_website.storage(
  productID INT,
  storeID INT,
  amount INT,
  PRIMARY KEY(productID, storeID),
  FOREIGN KEY(productID) REFERENCES grocery_website.product(productID),
  FOREIGN KEY(storeID) REFERENCES grocery_website.store(storeID)
);
CREATE TABLE grocery_website.transaction_instance(
  transactionID INT,
  productionID INT,
  quantity INT,
  ranking ENUM('1', '2', '3', '4', '5'),
  PRIMARY KEY(transactionID, productionID),
  FOREIGN KEY(transactionID) REFERENCES grocery_website.transaction(transactionID),
  FOREIGN KEY(productionID) REFERENCES grocery_website.product(productID)
);
CREATE TABLE grocery_website.promotion(
  promotionID INT NOT NULL AUTO_INCREMENT,
  start_date DATE,
  end_date DATE,
  discount REAL,
  productID INT,
  PRIMARY KEY(promotionID),
  FOREIGN KEY(productID) REFERENCES grocery_website.product(productID)
);
```

Front-End Design and Front-End to Back-End Connection

A description of your front-end design as well as the front-end to back-end connection.

Front-End Design

See next page

Description

The front-end of the application begins with the login page. There is an option to login with a username and password as well as sign up for an account. If the user does not have an account, they follow the path to sign up as home customer or business customer, including a form and confirmation page. After logging in, all customer roles will continue to the front-end path. Here, there are various options. The customer can browse, search, add/remove products from the shopping cart, view their shopping cart, view purchases, and check out. The paths are outlined in the diagram.

Main Page: Featured products, navigation by category, search, shopping cart, profile, log out
Search Result Page: Name, price (search by price from lower to higher), rating, category, brand, add to cart
Product Page: Name, description, add to cart, rating
Shopping Cart: List of products, remove, quantity, price, subtotal price, days of shipment, address editions, confirm
Transaction History: List of transactions, products purchased, rating (star system)

Back-End Design

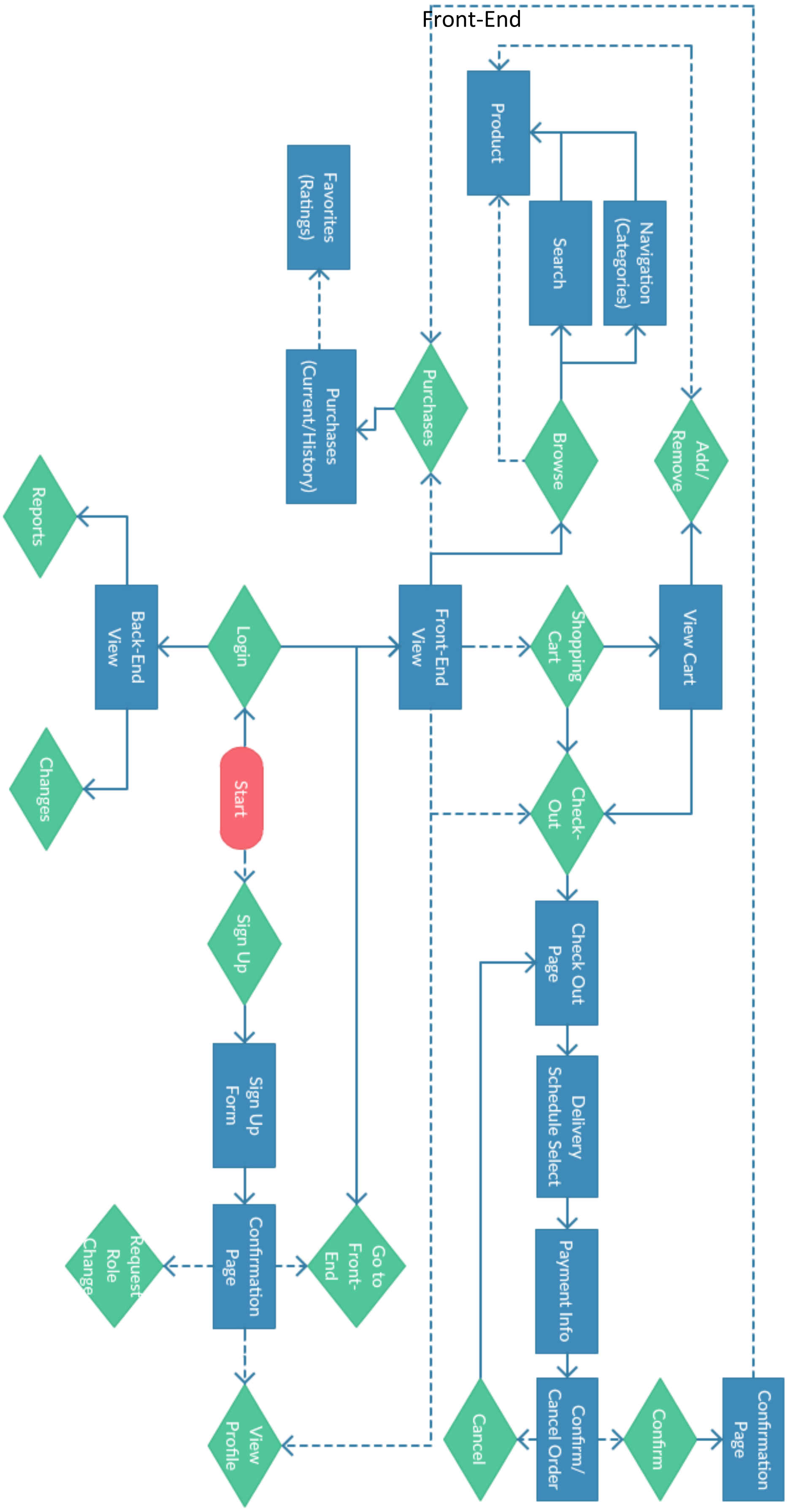
See page after front-end diagram

Description

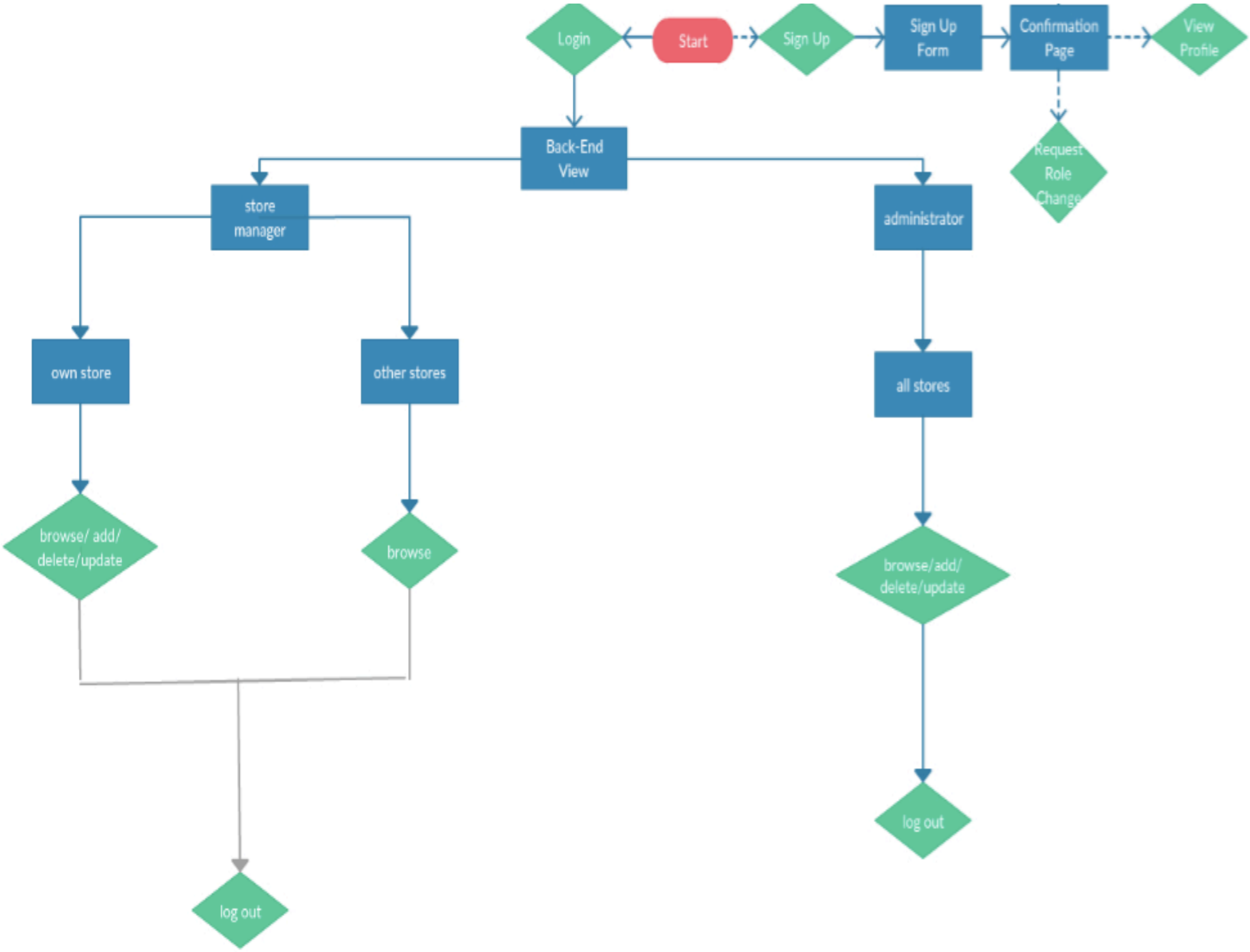
The back-end of the application begins with the login page, including a username and password. After logging in, all manager roles will continue to the back-end path. Here, there are various options. Store managers can browse the information of each product in all stores, and add/delete/update products in their own stores as well as browse the information in other stores. Administrators can browse the information of each product in stores, and add/delete/update stores and user types. The paths are outlined in the diagram.

(Store Manager View)
Main page: Slogan, store’s name, other stores’ name
My Store page: Product ID, product name, price, rating, category and brand, log out, add, delete, update
Others’ stores page: Product ID, product name, price, rating, category and brand, log out, add, delete, update

(Admin View)
Main page: Slogan, all stores' name
All Stores Page: Store ID, store name, region, state, city, zip code, store manager, log out, add, delete, update
Product: browse
Promotion: browse
Category: Browse
Brand: Browse
Transaction: Browse
Address: Browse
User: User name, user type, password, update, update, delete, add, log out


Front-End
Start
Login
Sign Up
Sign Up Form
Confirmation Page
Go to Front-End
Request Role Change
View Profile
Back-End View
Reports
Changes
Front-End View
Purchases
Purchases (Current/History)
Favorites (Ratings)
Browse
Search
Navigation (Categories)
Product
Shopping Cart
View Cart
Add/ Remove
Check-Out
Check Out Page
Delivery Schedule Select
Payment Info
Confirm/ Cancel Order
Cancel
Confirm
Confirmation Page

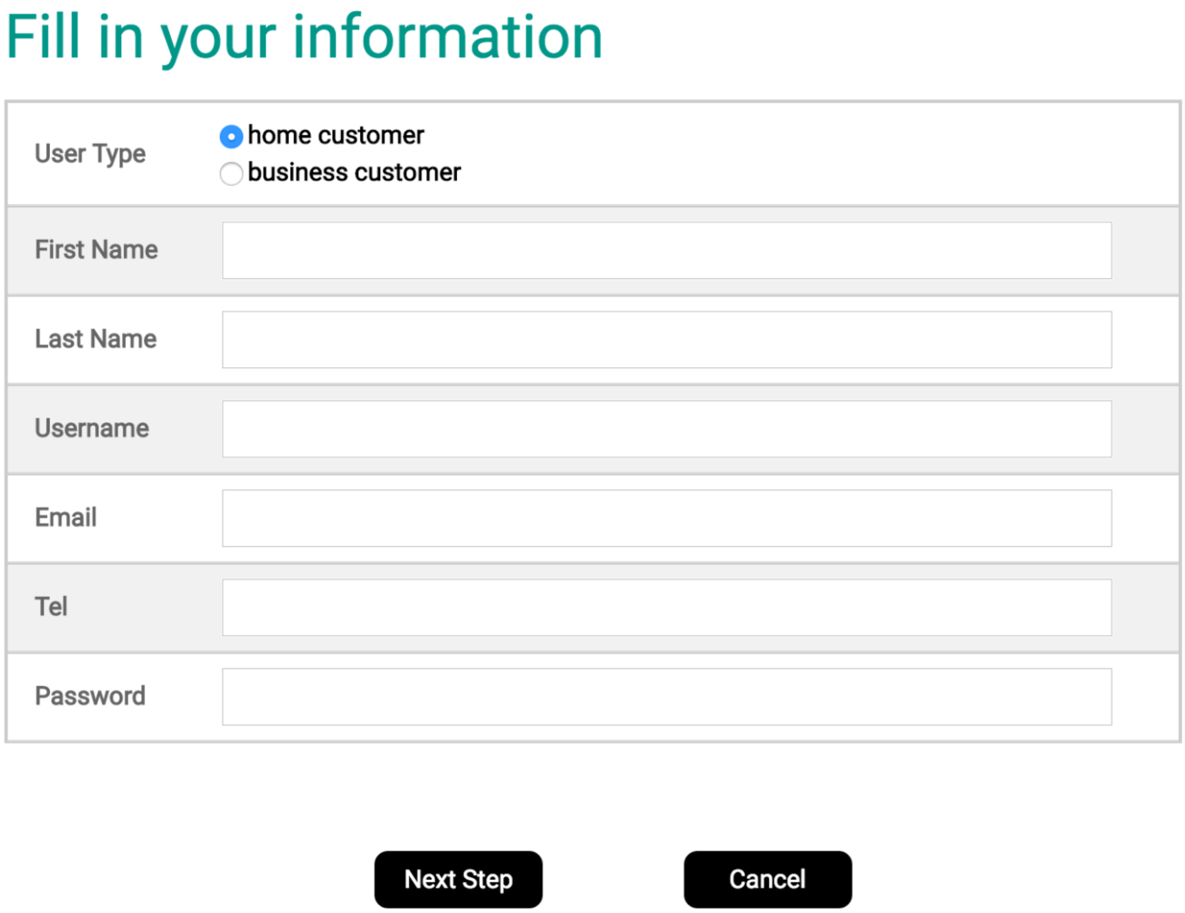
Back-End


Login
Start
Sign Up
Sign Up Form
Confirmation Page
View Profile
Request Role Change
Back-End View
store manager
administrator
own store
other stores
all stores
browse/ add/ delete/update
browse
browse/add/ delete/update
log out
log out

Overview of System Implementation

A brief overview of the system implementation with example screen shots.

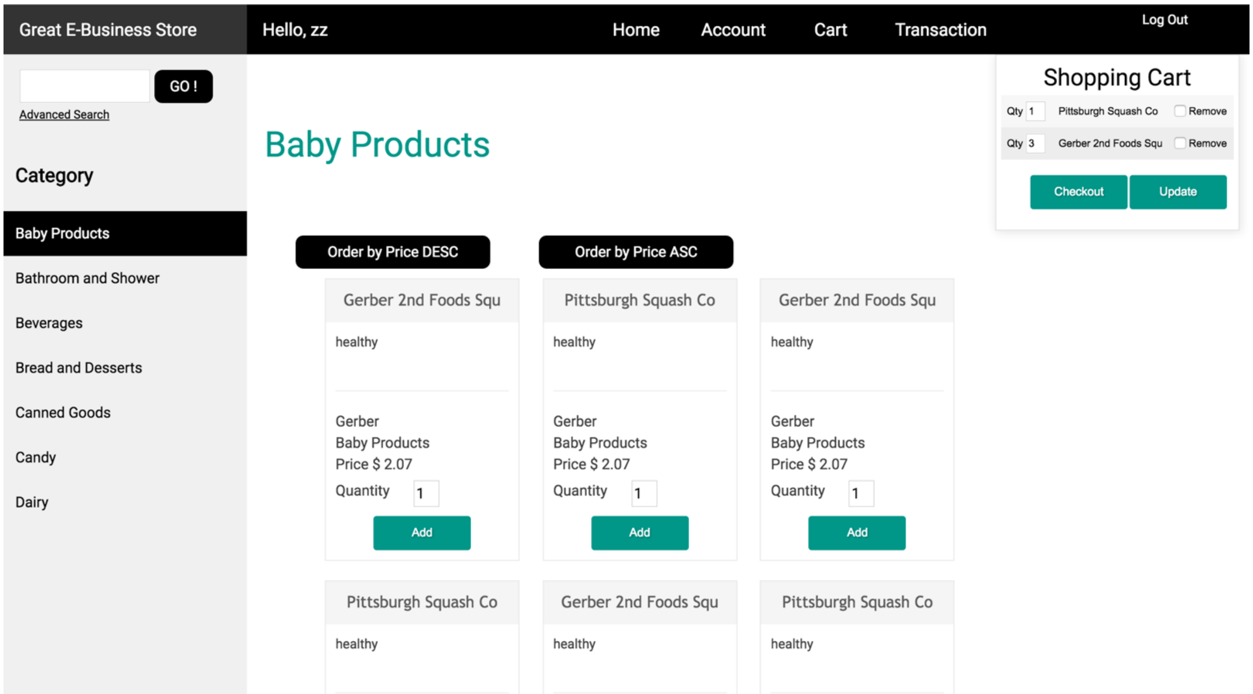
Create account, users can choose the customer type and fill in other information to sign up.



Fill in your information

User Type	● home customer ○ business customer
First Name	
Last Name	
Username	
Email	
Tel	
Password	

Customer view products, can view by price descending or ascending.
Customer implementation with cart. Shopping cart will always show at the right corner in the page no matter where the customer scrolls the page or change the category page as long as there are products in cart.



Baby Products

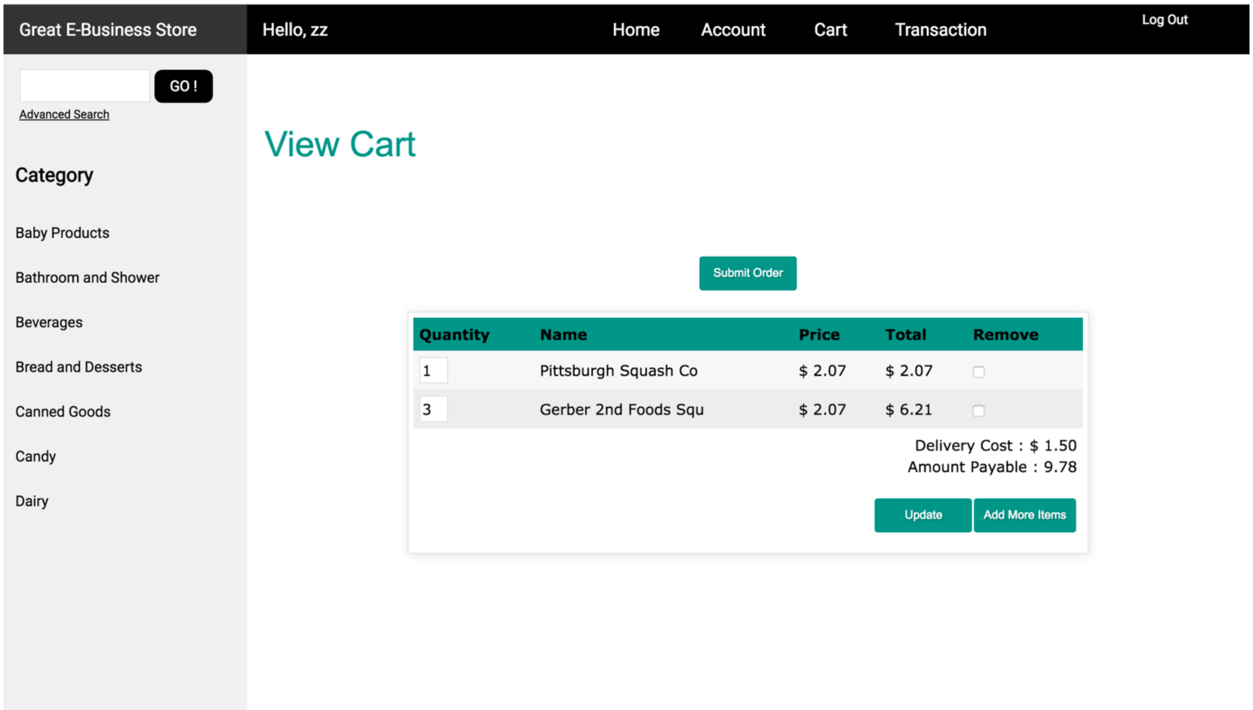
Shopping Cart

Qty	1	Pittsburgh Squash Co	□ Remove
Qty	3	Gerber 2nd Foods Squ	□ Remove

Gerber 2nd Foods Squ	Pittsburgh Squash Co	Gerber 2nd Foods Squ
healthy	healthy	healthy
Gerber Baby Products Price $ 2.07 Quantity 1	Gerber Baby Products Price $ 2.07 Quantity 1	Gerber Baby Products Price $ 2.07 Quantity 1

Pittsburgh Squash Co	Gerber 2nd Foods Squ	Pittsburgh Squash Co
healthy	healthy	healthy

Customer view cart.

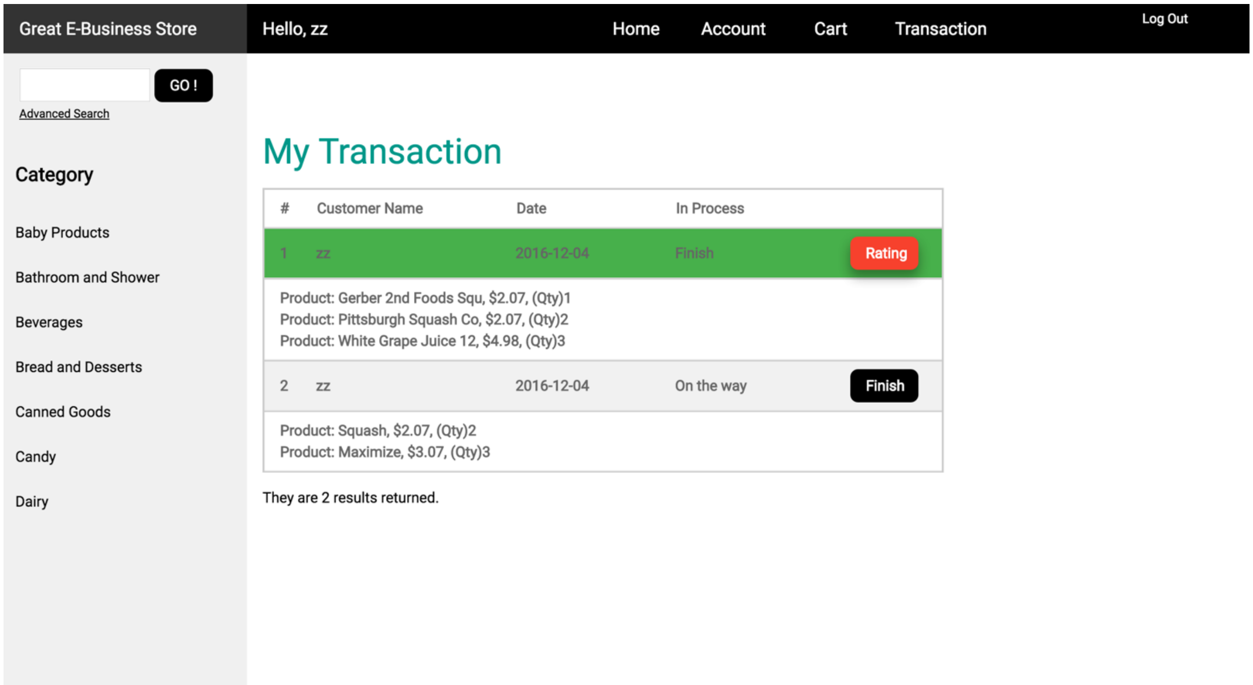


View Cart

Quantity	Name	Price	Total	Remove
1	Pittsburgh Squash Co	$ 2.07	$ 2.07	□
3	Gerber 2nd Foods Squ	$ 2.07	$ 6.21	□

Delivery Cost : $ 1.50
Amount Payable : 9.78

Customer view transaction. They can confirm the complement of transactions, and rating the transaction.

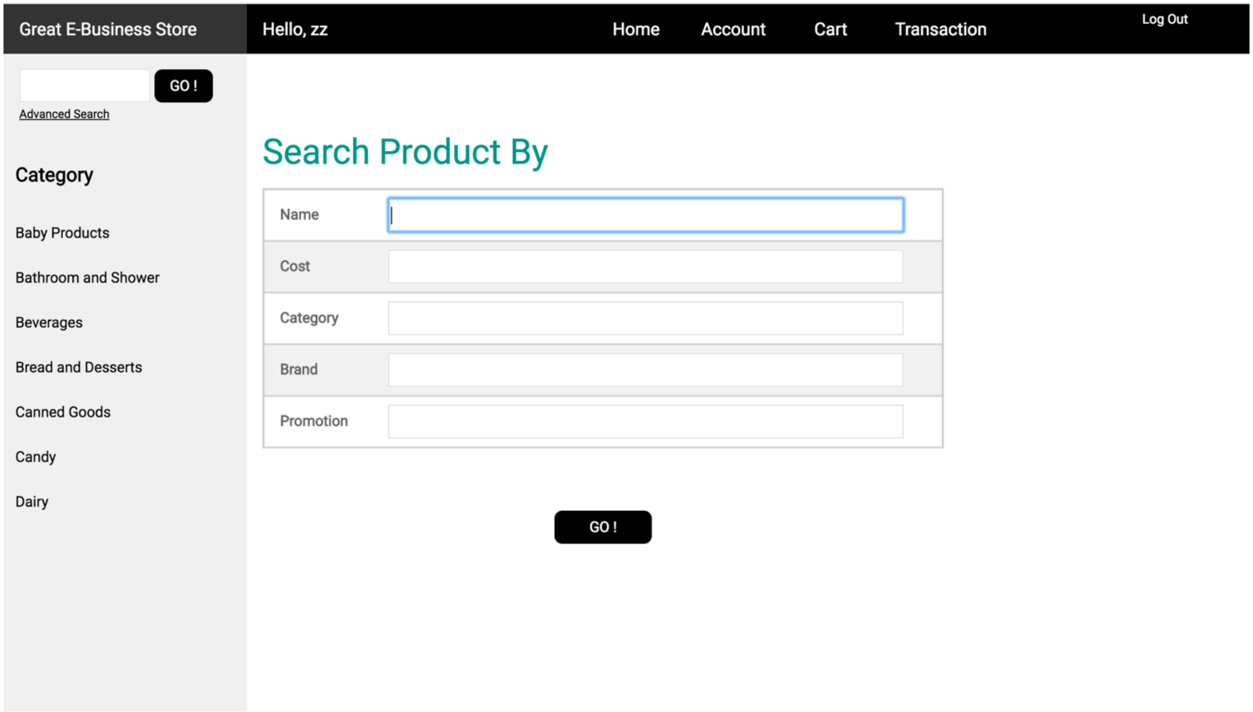


My Transaction

#	Customer Name	Date	In Process	
1	zz	2016-12-04	Finish	Rating
Product: Gerber 2nd Foods Squ, $2.07, (Qty)1 Product: Pittsburgh Squash Co, $2.07, (Qty)2 Product: White Grape Juice 12, $4.98, (Qty)3				
2	zz	2016-12-04	On the way	Finish
Product: Squash, $2.07, (Qty)2 Product: Maximize, $3.07, (Qty)3				

They are 2 results returned.

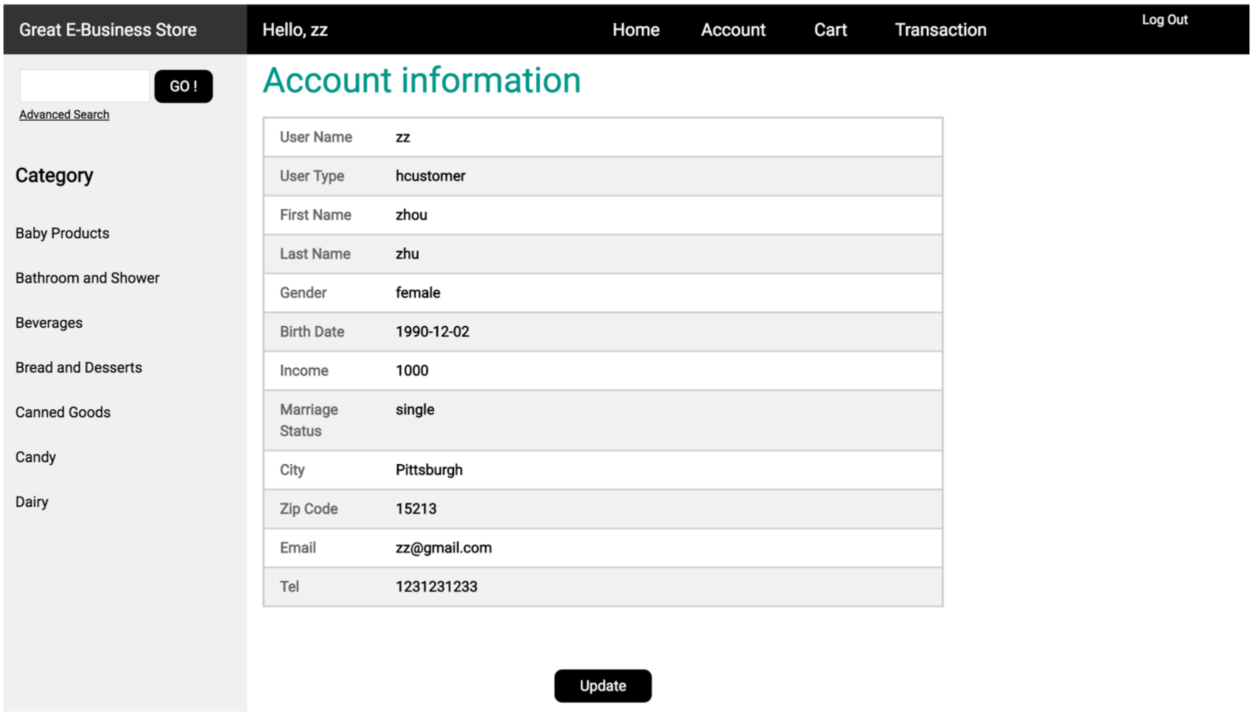
Customer can use multiple search and fuzzy search.



Search Product By

Name	
Cost	
Category	
Brand	
Promotion	

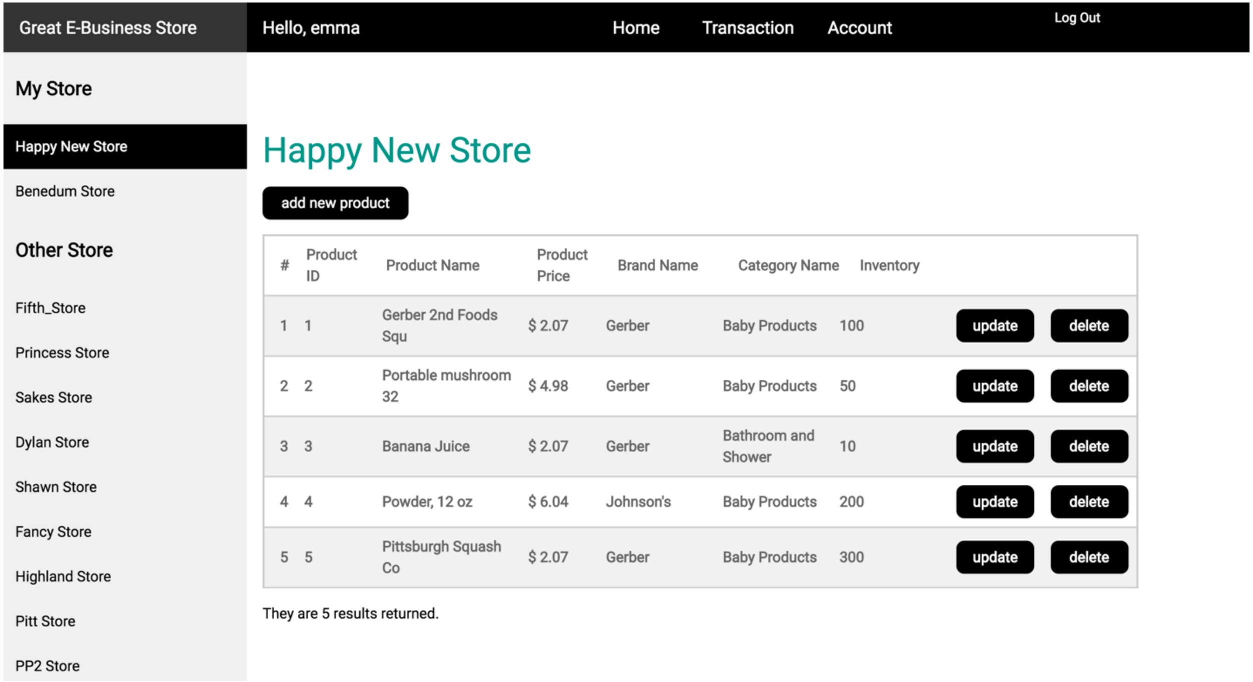
Customer can view their own account information and update it.



Account information

User Name	zz
User Type	hcustomer
First Name	zhou
Last Name	zhu
Gender	female
Birth Date	1990-12-02
Income	1000
Marriage Status	single
City	Pittsburgh
Zip Code	15213
Email	zz@gmail.com
Tel	1231231233

Store manager and region manager can manipulate the products in their stores, and only view others stores' products information. They can also view transactions in their stores.
View the list of products in certain store.

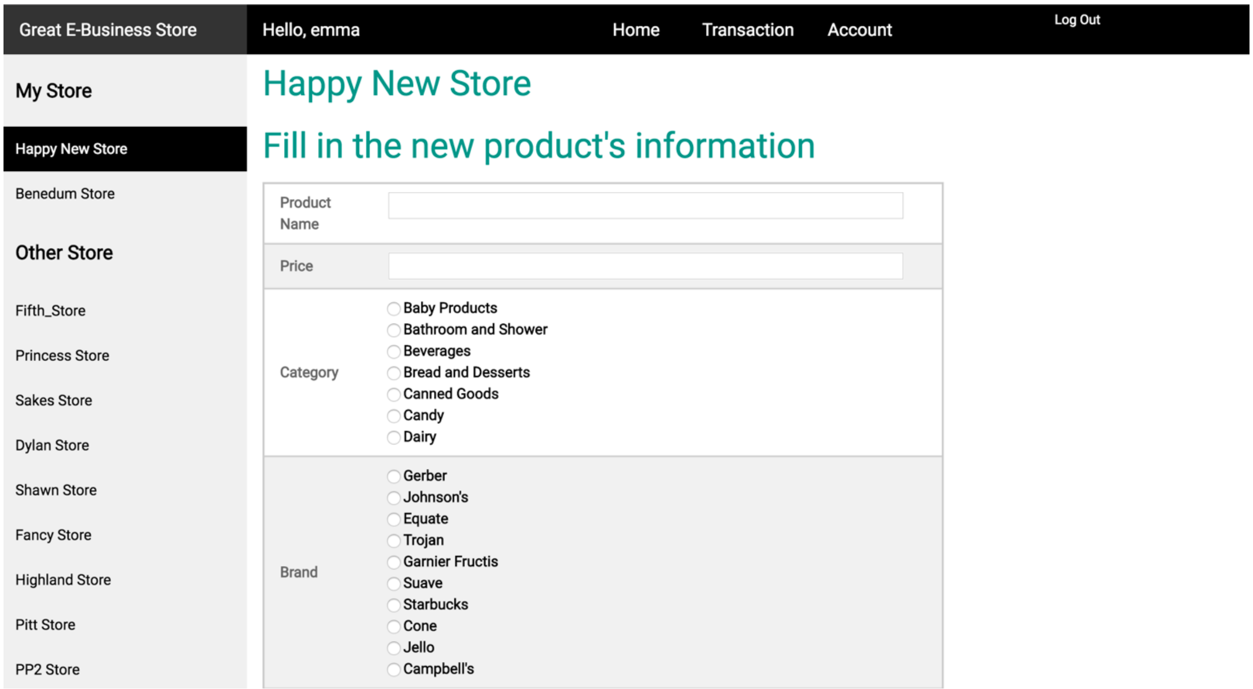


Happy New Store

#	Product ID	Product Name	Product Price	Brand Name	Category Name	Inventory		
1	1	Gerber 2nd Foods Squ	$ 2.07	Gerber	Baby Products	100	update	delete
2	2	Portable mushroom 32	$ 4.98	Gerber	Baby Products	50	update	delete
3	3	Banana Juice	$ 2.07	Gerber	Bathroom and Shower	10	update	delete
4	4	Powder, 12 oz	$ 6.04	Johnson's	Baby Products	200	update	delete
5	5	Pittsburgh Squash Co	$ 2.07	Gerber	Baby Products	300	update	delete

They are 5 results returned.

Add a product to a certain store.



Great E-Business Store | Hello, emma | Home | Transaction | Account | Log Out

My Store

Happy New Store

Benedum Store

Other Store

Fifth_Store

Princess Store

Sakes Store

Dylan Store

Shawn Store

Fancy Store

Highland Store

Pitt Store

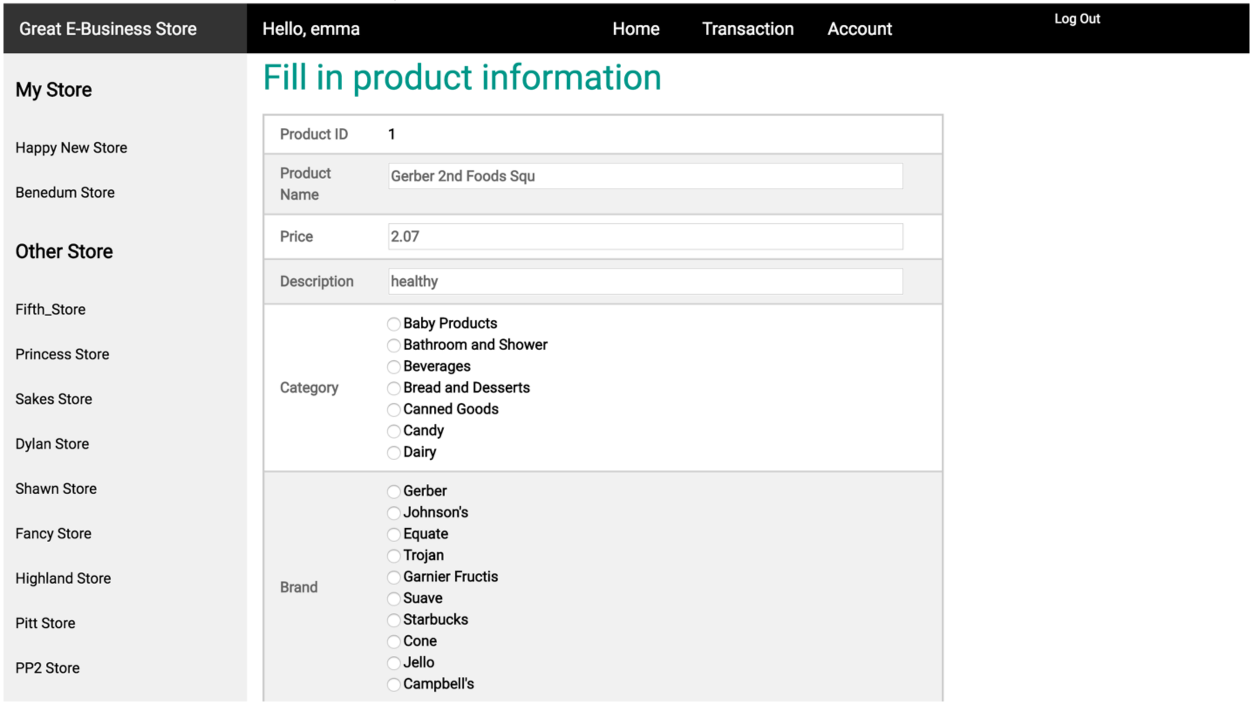
PP2 Store

Happy New Store

Fill in the new product's information

Product Name	
Price	
Category	○ Baby Products ○ Bathroom and Shower ○ Beverages ○ Bread and Desserts ○ Canned Goods ○ Candy ○ Dairy
Brand	○ Gerber ○ Johnson's ○ Equate ○ Trojan ○ Garnier Fructis ○ Suave ○ Starbucks ○ Cone ○ Jello ○ Campbell's

Revise the information of the product.



Great E-Business Store | Hello, emma | Home | Transaction | Account | Log Out

My Store

Happy New Store

Benedum Store

Other Store

Fifth_Store

Princess Store

Sakes Store

Dylan Store

Shawn Store

Fancy Store

Highland Store

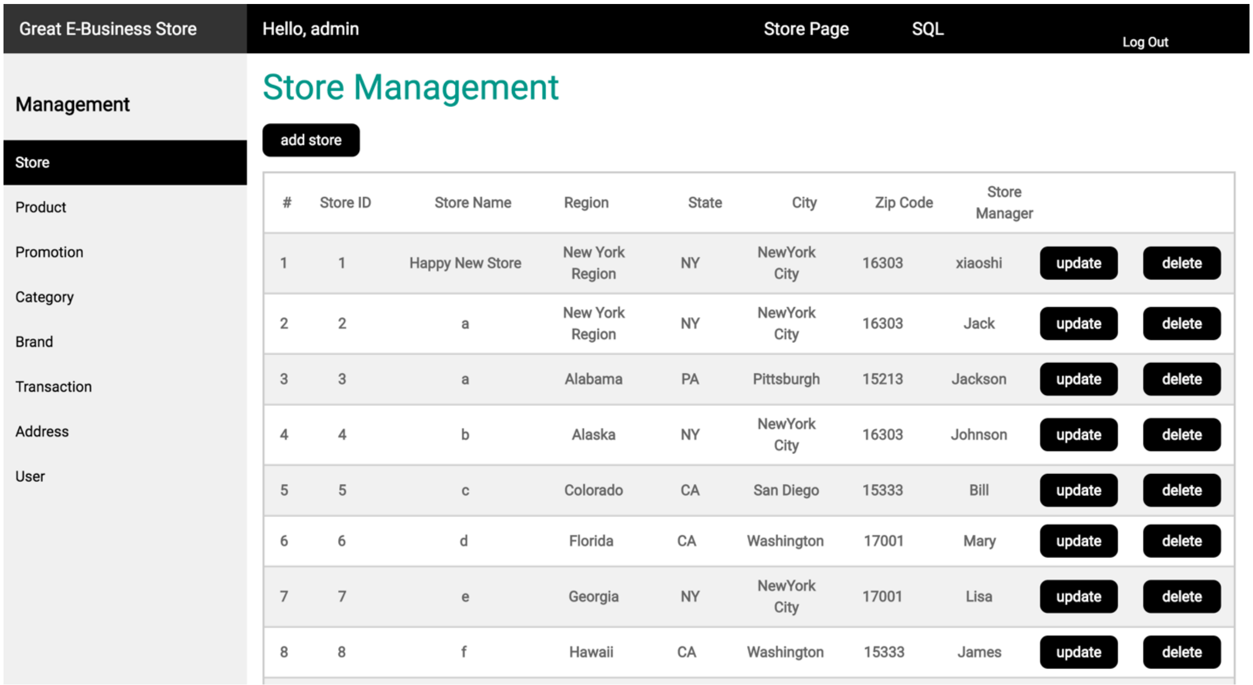
Pitt Store

PP2 Store

Fill in product information

Product ID	1
Product Name	Gerber 2nd Foods Squ
Price	2.07
Description	healthy
Category	○ Baby Products ○ Bathroom and Shower ○ Beverages ○ Bread and Desserts ○ Canned Goods ○ Candy ○ Dairy
Brand	○ Gerber ○ Johnson's ○ Equate ○ Trojan ○ Garnier Fructis ○ Suave ○ Starbucks ○ Cone ○ Jello ○ Campbell's

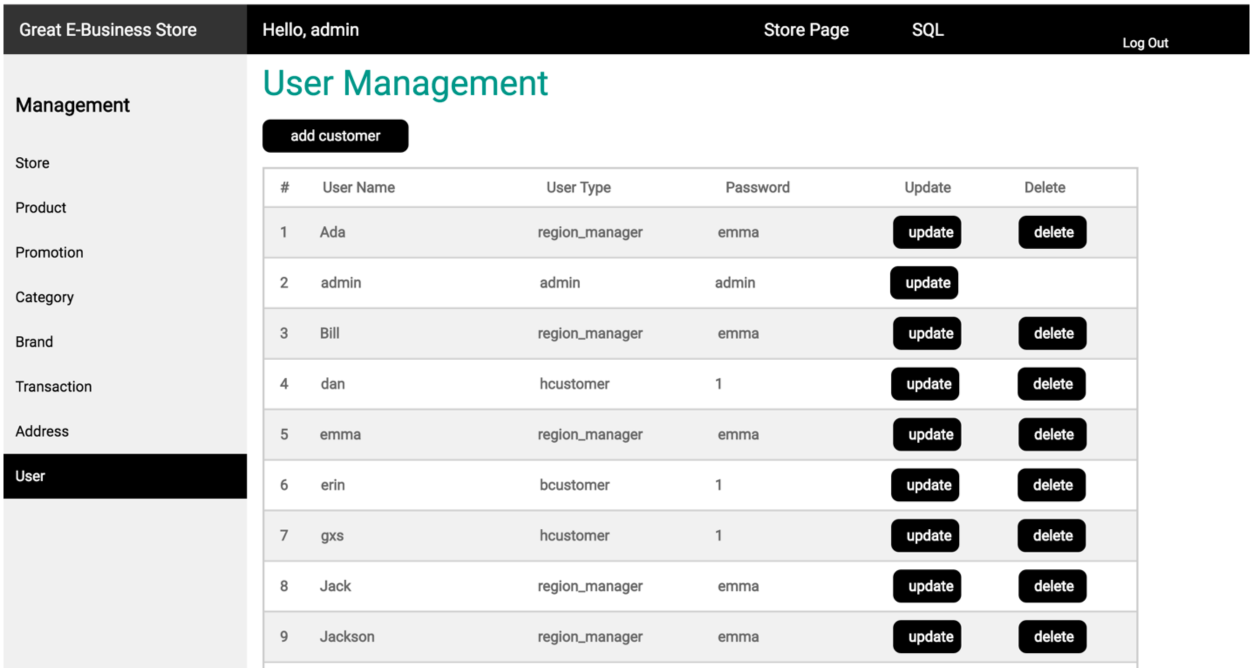
Admin can view, update and delete stores and users.
View the list of stores.



Store Management

#	Store ID	Store Name	Region	State	City	Zip Code	Store Manager		
1	1	Happy New Store	New York Region	NY	NewYork City	16303	xiaoshi	update	delete
2	2	a	New York Region	NY	NewYork City	16303	Jack	update	delete
3	3	a	Alabama	PA	Pittsburgh	15213	Jackson	update	delete
4	4	b	Alaska	NY	NewYork City	16303	Johnson	update	delete
5	5	c	Colorado	CA	San Diego	15333	Bill	update	delete
6	6	d	Florida	CA	Washington	17001	Mary	update	delete
7	7	e	Georgia	NY	NewYork City	17001	Lisa	update	delete
8	8	f	Hawaii	CA	Washington	15333	James	update	delete

View the list of users.



User Management

#	User Name	User Type	Password	Update	Delete
1	Ada	region_manager	emma	update	delete
2	admin	admin	admin	update	
3	Bill	region_manager	emma	update	delete
4	dan	hcustomer	1	update	delete
5	emma	region_manager	emma	update	delete
6	erin	bcustomer	1	update	delete
7	gxs	hcustomer	1	update	delete
8	Jack	region_manager	emma	update	delete
9	Jackson	region_manager	emma	update	delete

Admin can see the list of product, promotion, category, brand, transaction and address. An example page of product list.

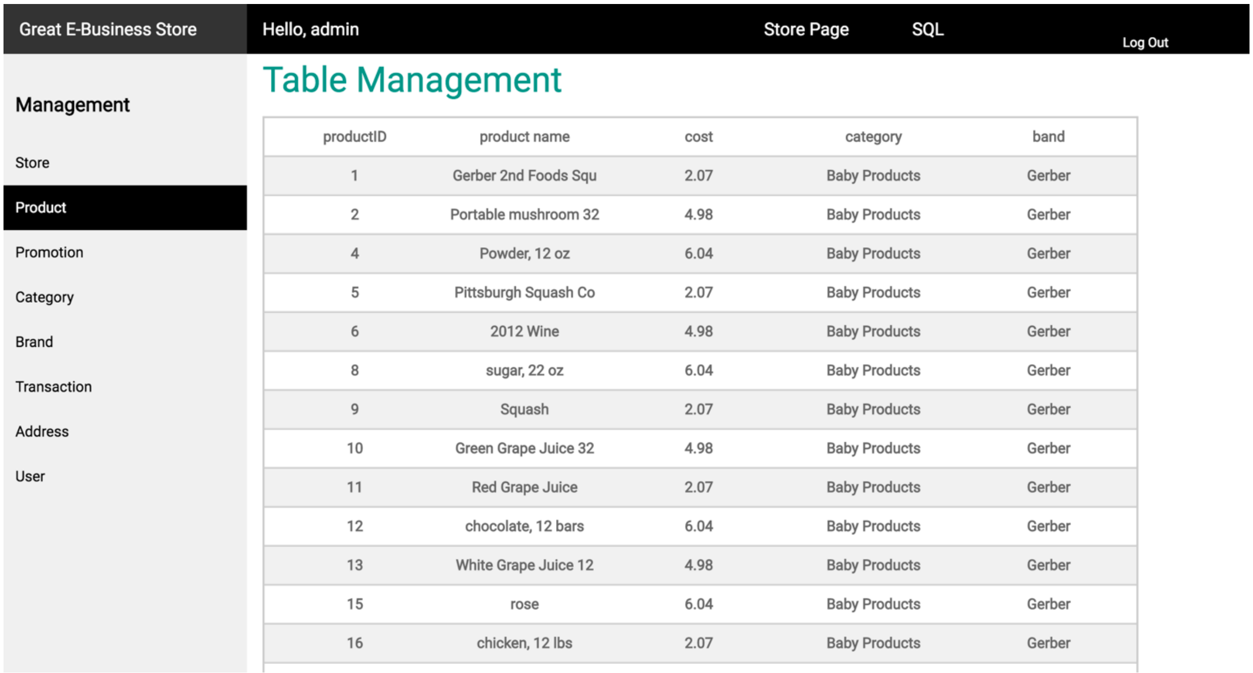


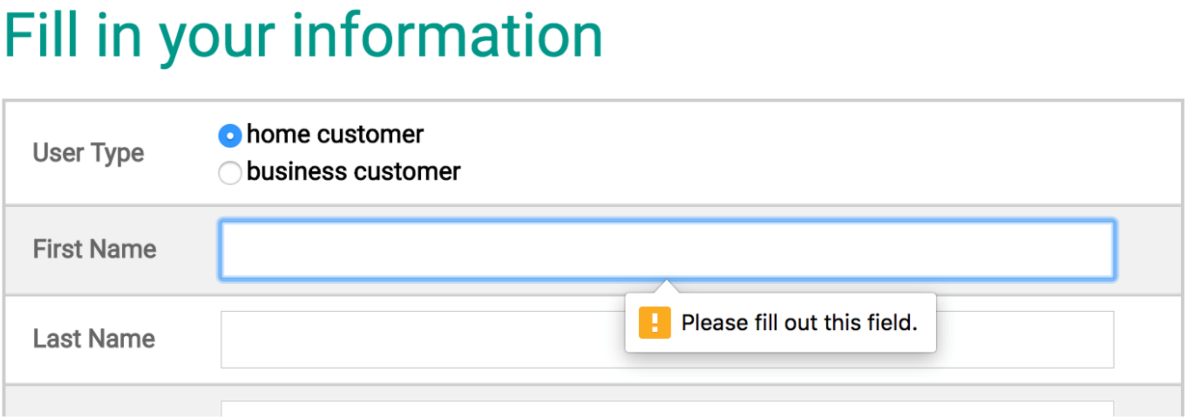
Table Management

productID	product name	cost	category	band
1	Gerber 2nd Foods Squ	2.07	Baby Products	Gerber
2	Portable mushroom 32	4.98	Baby Products	Gerber
4	Powder, 12 oz	6.04	Baby Products	Gerber
5	Pittsburgh Squash Co	2.07	Baby Products	Gerber
6	2012 Wine	4.98	Baby Products	Gerber
8	sugar, 22 oz	6.04	Baby Products	Gerber
9	Squash	2.07	Baby Products	Gerber
10	Green Grape Juice 32	4.98	Baby Products	Gerber
11	Red Grape Juice	2.07	Baby Products	Gerber
12	chocolate, 12 bars	6.04	Baby Products	Gerber
13	White Grape Juice 12	4.98	Baby Products	Gerber
15	rose	6.04	Baby Products	Gerber
16	chicken, 12 lbs	2.07	Baby Products	Gerber

Testing and Errors

A description of your testing efforts and erroneous cases that your system can detect and handle.

Create and update account
Our system can find the empty field when users submit the form. So the page will show the notice of the empty field and refuse to submit.

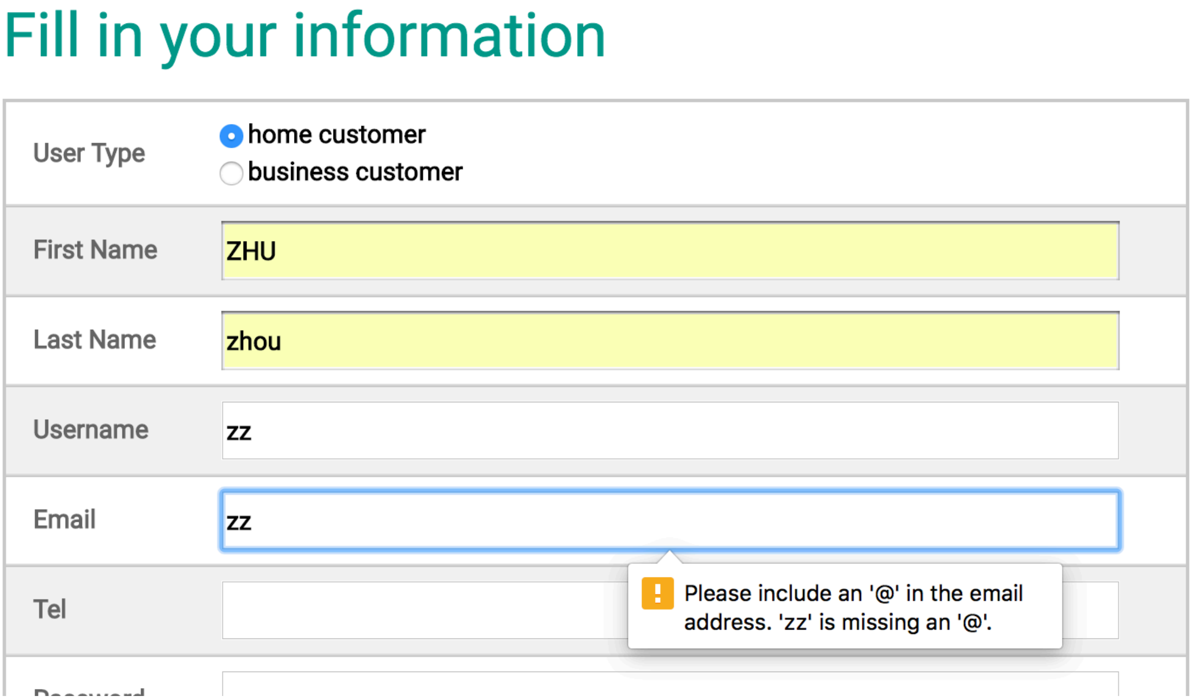


Fill in your information

User Type	● home customer ○ business customer
First Name	
Last Name	

Please fill out this field.

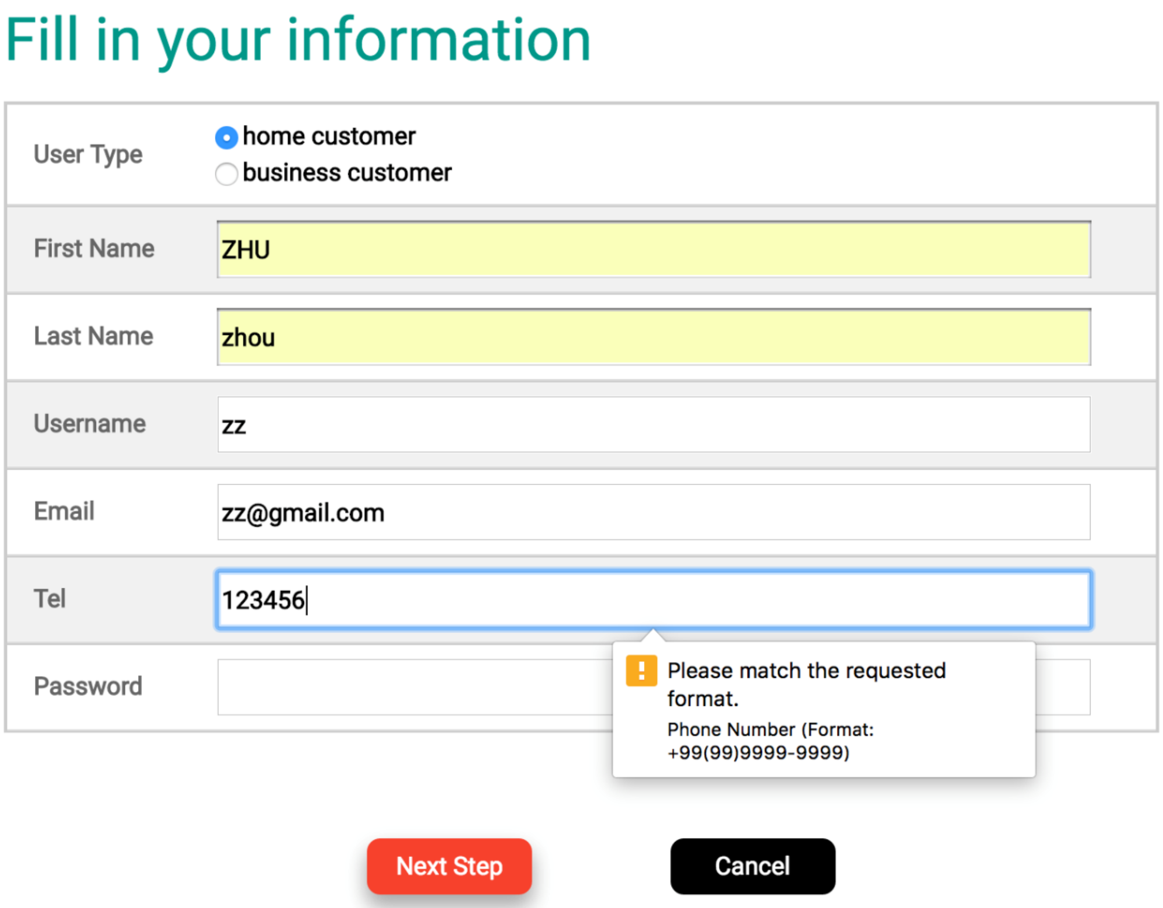
In terms of email and telephone number fields, our system will check the pattern and show the notice if the user does not input a valid value.



Fill in your information

User Type	● home customer ○ business customer
First Name	ZHU
Last Name	zhou
Username	zz
Email	zz
Tel	

Please include an '@' in the email address. 'zz' is missing an '@'.

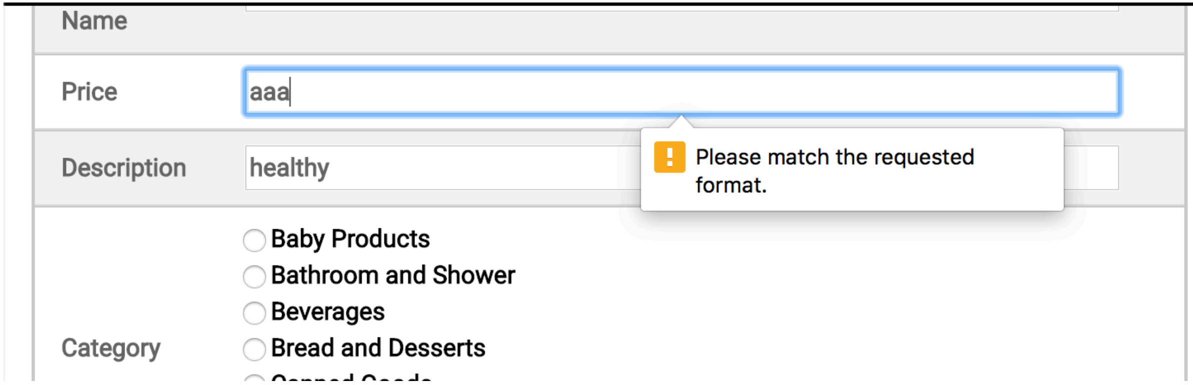


Fill in your information

User Type	● home customer ○ business customer
First Name	ZHU
Last Name	zhou
Username	zz
Email	zz@gmail.com
Tel	123456
Password	

Please match the requested format.
Phone Number (Format: +99(99)9999-9999)

Add or update the price of product
When user enter price of the product, the price should be numbers with the format “xx.xx”, if it is not, the system will show the notice.



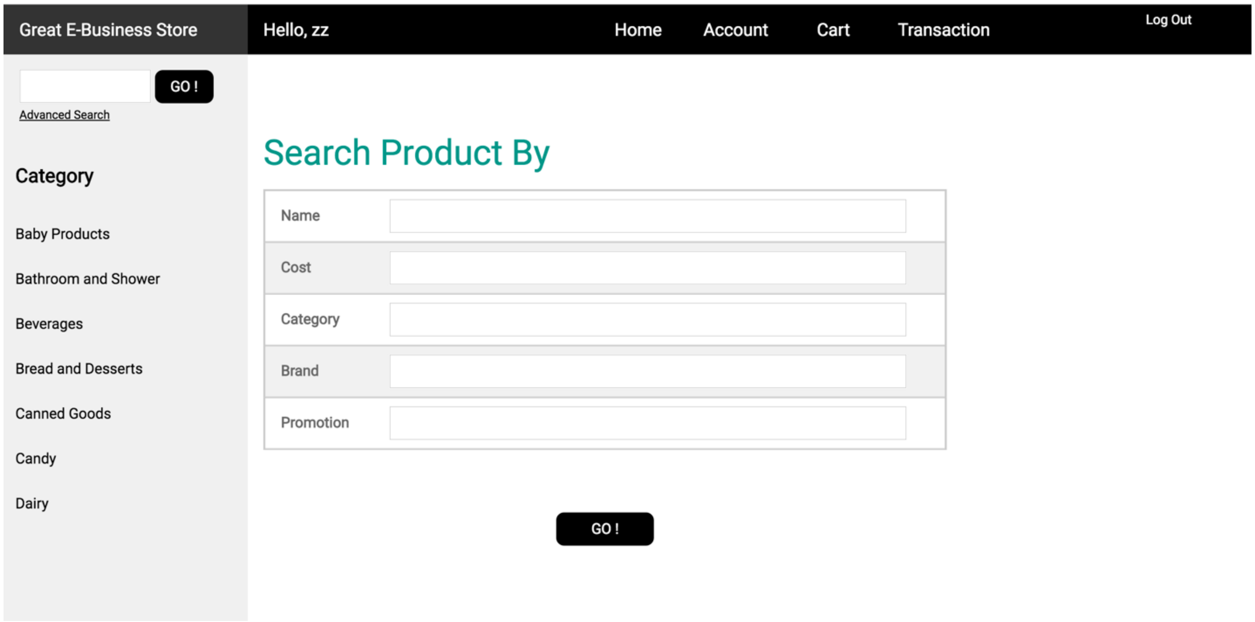
Name	
Price	aaa
Description	healthy
Category	○ Baby Products ○ Bathroom and Shower ○ Beverages ○ Bread and Desserts

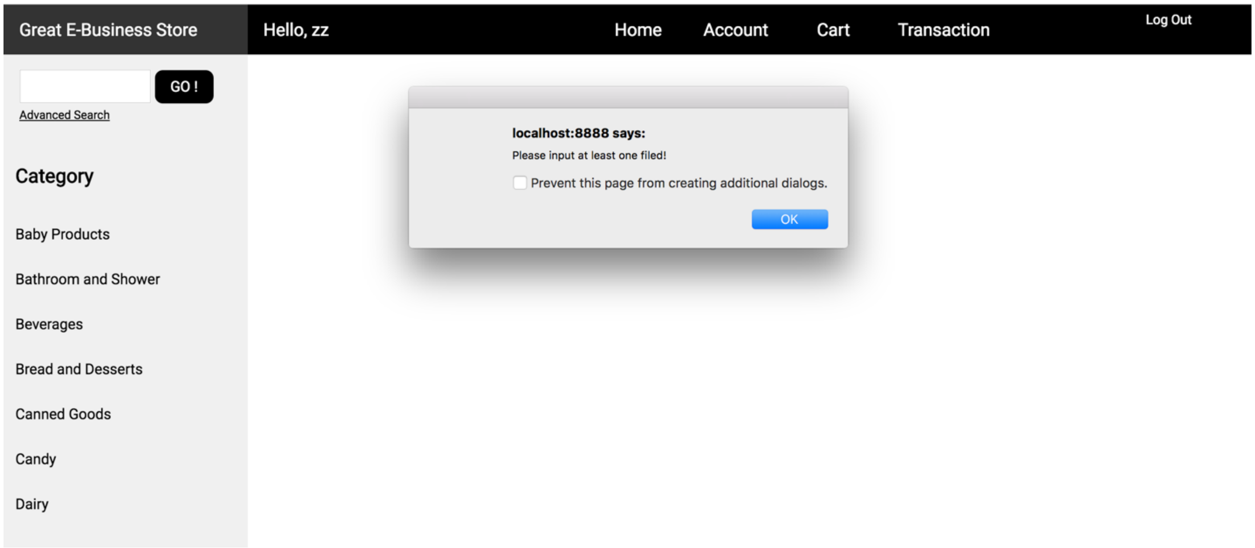
Please match the requested format.

Add product to cart
When add product to cart, customers can input the quantity they want to buy, if they input negative number or decimal, our system will show the notice of the invalid input.

Multiple search

If users are using multiple search without inputting any fields in the form, the system will pop up a notice to remind the user.





localhost:8888 says:

Please input at least one filed!

□ Prevent this page from creating additional dialogs.

Limitations and Possibilities for Improvements

A description of the system's limitations and the possibilities for improvements.

For the back-end part:
In reality, the store manager could get the weekly, monthly and yearly data. But if we do in that way, we need a much larger database which is too complicated. But if we have more time we could enlarge our database to do a report related to time period.

For products in different store:
In reality, the different stores could can have same products, but these products may have different information, such as price, description. But if we take this situation into account, the database should be much complicated.

For zip code:
In reality, every area has its own unique zip code, but the street name can be the same in different areas. In our system we simplify this condition and assume that several areas can have streets with same names, and each street name combines with a unique zip code.

For delivery:
In reality, there would be an interface to select a delivery date and time for the customer's transaction. Then there would be a job assignment in the back-end for a specific delivery person to be assigned to this transaction. Then there would be a process to track the delivery of the groceries to the customer's door.

For payment:
In reality, there would be a checkout page to collect credit card and billing information from the customer in order for them to pay online. For now, the system assumes the customer pays the delivery person instead.

For transaction and lock in database:
In reality, if the system crashes with a transaction does not finish, when system recover after that, the un-done transaction should be undo and cannot stay at the unfinished status. But if we take this situation into account, it should be much more complicated to realize that. And in terms of necessity with lock, when comes into several customers buy the same product in inventory, the resource should be locked for only one customer.