INFSCI 2710: Database Management

Project: Database System for E-commerce

Group Member: Siwei Liu (sil29), Jun Zhao (juz21), Yiheng Hu (yih44)

1. System Overview

The cooperation is a distributor that sells Dell computers and accessories by physical stores in different locations. The system is an internal management system of the cooperation that helps employees to do their daily works with online real-time information, which mainly focus on the management of sales transactions, inventories, customers and employees. Customers may also see products and prices that the cooperation offers in the web pages, but cannot place orders with this system.

2. System Assumptions

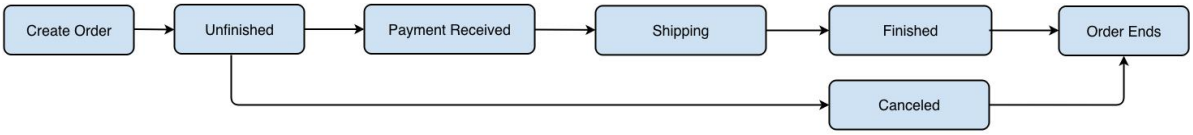
The cooperation has 100 stores in 10 regions, and has about 100000 customers. Customers are divided into two types: business customers and individual customers, and both their number are 50000. Customers can browse the website to see products and price, but they are not able to place orders online. Customers don't need to log in to the system.

Each store has 9 salesman and 1 store manager, and every region has a region manager. Therefore, there are totally 1010 employees. Employees must log in the system before any operations. Salesmen can create/manage customer information, check product and inventory, and create/manage orders, payments and delivery; store managers can manage inventory, employees, customer; region manager can manage stores, regions, and get information of sales statistics.

Every store has a warehouse to store products. Therefore, inventory records are identified by both stores and products. Employees cannot insert inventory records, but can only select or update.

Orders have different status: unfinished, payment received, shipping, finished and canceled. When an order is created, the original status is "unfinished". Its status can be changed to "payment received" or "canceled". The order only can be canceled before the payments are received, otherwise the

order must complete eventually; it means no returns are accepted. After "payment received", the next status is "shipping". After the products are received by customers, the order is "finished". The flowchart is as below.



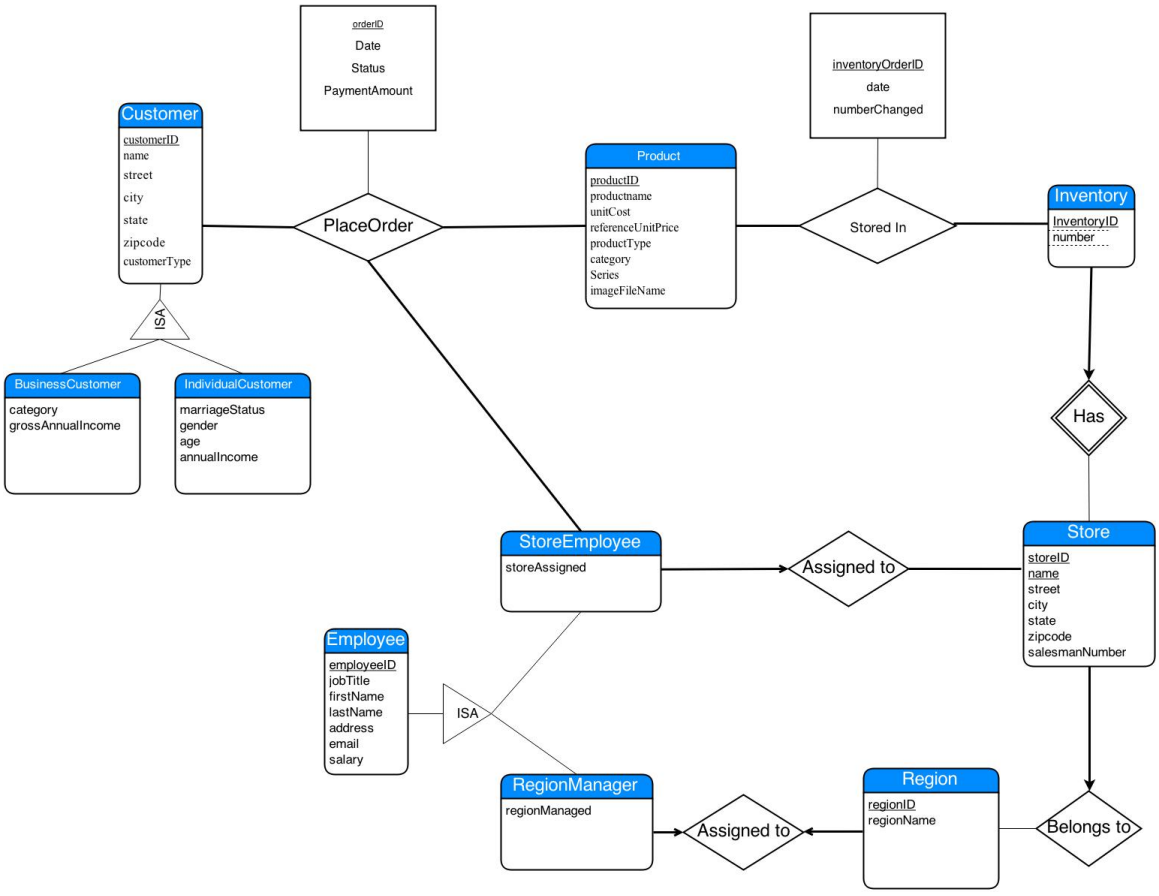
Customers can browse the web site to check products and prices, but they cannot place orders online; customers should come to the stores in person, or call salesman by phone to place orders. After customers' request, salesmen will check inventory first. If the number of customers' request is smaller than stock number, salesman will create an order.

If customers come to store in person, customers probably place an order and pay after having the product in their hands, so it is not possible to have a situation that created an order a few moments before but the product becomes out of stock later. If customer calls a salesman to place an order, the store will deliver product after receiving payment. In this case, salesman is able to create order with product that are out of stock or not enough, and will deliver product when it is available.

When order is created, an order record and one or several order details that show products and numbers will be inserted into database. When the status is "payment received", a payment record will be inserted. When the status is "shipping", an inventory order and details of the inventory order will be inserted.

3. Database Design

3.1. E-R Diagram



- **Entity Sets**

The figure above is the E-R diagram designed for this system. There are 10 entities sets: Customer, BusinessCustomer, IndividualCustomer, Employee, StoreEmployee, RegionManager, Product, Store, Inventory, and Region.

Customer is divided into two categories: individual customer, and business customer. We use ISA Hierarchy to represent the relationship between BusinessCustomer and IndividualCustomer, and customer. For all the customers, it has several attributes: unique customerID, name, address information and customerType differentiates whether a customer is an individual customer or business customer. BusinessCustomer Entity also has two exclusive attributes:

business category and gross annual income. IndividualCustomer Entity includes four exclusive attributes: gender, age, marriage status and annual income.

Employees have three types: regular salesman, store manager, and region manager. We put Regular salesman and store manager into storeEmployee entity, since they must be assigned into one store. We also use ISA Hierarchy to represent the relationship between employee entity and its children. All employees have the attributes: employeeID, jobTitle, firstName, lastName, address, email, and salary. For storeEmployee, it has one exclusive storeAssigned attribute. RegionManager has exclusive attribute regionAssigned.

The products we sell are Dell computers and accessories. Thus, the product entity has these attributes: productID, productName, unitCost, series, referenceUnitPrice, productType (for home, or for work), category (laptop, PC, or accessories) and imageFileName.

Each Store entity has a weak entity Inventory, which means each store has its own warehouse. Store Entity has attributes: storeID, name, address information and salesmanNumber. Weak entity inventory has two attributes: inventoryID and number which represents the number of product in stock.

Each store belongs to one region and this region is represented by Region entity. This entity has two attributes: regionID and regionName.

- **Relationship Set**

Here I want to demonstrate two relationship sets: placeOrder and Stored in. For placeOrder, it represents the action that salesman create an order for a customer. This relationship relates with three entities (customer, products, and StoreEmployee). It has four attributes: orderID, date, status (unfinished, payment received, shipping), and paymentAmount means the how much the customer has paid.

Stored in represents the relationship between product entity and inventory entity. We assume all products store in the store's own inventory. When one order is generated, salesman needs to get the product from the warehouse. As a

result, we need these attributes: inventoryOrderID, date, numberChanged means the number of one product for each order.

3.2. Relational Schema

3.2.1 General database implement

First of all, we create tables that based on every entity except region manager. Therefore we have these tables: Employee, Region, Store, StoreEmployee, Product, Inventory, Customer, IndividualCustomer, and BusinessCustomer. The information of RegionManager entity is duplicated from the information of Region, so we do not create the RegionManager table.

And then, we create tables for relations: Orders, OrderDetail, OrderPayment, OrderPaymentTransaction, InventoryOrder, InventoryOrderDetail. Since one order can have a few products, OrderDetails record every product of the order; OrderPayment, OrderPaymentTransaction, InventoryOrder, and InventoryOrderDetail are similar. Records are inserted into OrderPayment and OrderPayemntTransaction when the order's status is "payment received", and are inserted into InventoryOrder and InventoryOrderDetail when the order's status is "shipping".

We design the payment in two tables if customers pay in several times, so that OrderPaymentTransaction can record every single payment, and OrderPayment sum up the total payment of one order. However, system supports only one-time full amount payment now.

In addition, because of the system requirement, we created a Login table with username, password and usertype. The username is employee ID, and usertype is equal to their job title. Moreover, we create DBMS users for all employees, which have different grants to access the DBMS. Specific definitions of grants are below.

3.2.2 Users and Grants in DBMS:

Administrator: all approval.

Region manager:

Employee: select, insert, update

StoreEmployee: select, insert, update

RegionManager:select, insert, update

Region: select, insert, update

Store: select, insert, update

Customer: select, insert, update

BussniessCustomer: select, insert, update

IndividualCustomer: select, insert, update

Product: select, insert, update

Inventory: select, insert

Order: select

OrderDetail: select

InventoryOrder: select

InventoryOrderDetail: select

OrderPayment: select

OrderPaymentTransaction: select

Store manager:

Employee: select

StoreEmployee:select

Store: select

Region: select

Customer: select, insert, update

BussinessCustomer: select, insert, update

IndividualCustomer: select, insert, update

Product: select

Inventory: select, update

Order: select, insert, update

OrderDetail: select, insert, update

InventoryOrder: select, insert, update

InventoryOrderDetail: select, insert, update

OrderPayment: select, insert, update

OrderPaymentTransaction: select, insert, update

Salesman:

Store: select

Customer: select, insert, update

BussinessCustomer: select, insert, update

IndividualCustomer: select, insert, update

Product: select

Inventory: select, update

Order: select, insert, update

OrderDetail: select, insert, update

InventoryOrder: select, insert, update

InventoryOrderDetail: select, insert, update

OrderPayment: select, insert, update

OrderPaymentTransaction: select, insert, update

3.3. DDL Statements and Normal Form Identification

3.3.1 DDL Statements

```
create database infsci2710;
use infsci2710;

create table employee(
  employeeID int(5) primary key not null unique,
  firstName varchar(30) not null,
  lastName varchar(30) not null,
  jobTitle varchar(14) not null default 'salesman',
  street varchar(200),
  city varchar(200),
  state varchar(10),
```

```
  zipcode int(5),
  email varchar(200),
  salary decimal(7,2)
) ENGINE=InnoDB;

create table login(
  employeeID int(5) primary key not null unique,
  password varchar(15) not null,
  userType varchar(14) not null,
  foreign key (employeeID) references employee (employeeID)
) ENGINE=InnoDB;

create table region(
  regionID int primary key not null unique,
  name varchar(30) not null,
  regionManagerID int(5) not null,
  foreign key (regionManagerID) references employee (employeeID)
) ENGINE=InnoDB;

create table store(
  storeID int primary key not null unique,
  managerID int(5) unique not null,
  regionID int not null,
  name varchar(30) not null,
  street varchar(200),
  city varchar(200),
  state varchar(10),
  zipcode int(5),
      salesmanNumber int not null default 0,
  foreign key (managerID) references employee (employeeID),
```

```
  foreign key (regionID) references region (regionID)
) ENGINE=InnoDB;

create table storeEmployee(
  employeeID int(5) primary key not null unique,
  assignedStoreID int not null,
  foreign key (employeeID) references employee (employeeID),
  foreign key (assignedStoreID) references store (storeID)
) ENGINE=InnoDB;

create table customer(
  customerID int primary key not null unique,
  name varchar(100) not null,
  street varchar(200),
  city varchar(200),
  state varchar(10),
  zipcode int(5),
  customerType varchar(10) not null default 'individual'
) ENGINE=InnoDB;

create table businessCustomer(
  customerID int primary key not null unique,
  category varchar(30) not null,
  grossAnnualIncome decimal(10,2),
  foreign key (customerID) references customer (customerID)
) ENGINE=InnoDB;

create table individualCustomer(
  customerID int primary key not null unique,
  gender varchar(6),
```

```
  age int,
  annualIncome decimal(10,2),
  marriageStatus varchar(10),
  foreign key (customerID) references customer (customerID)
) ENGINE=InnoDB;

create table product(
  productID int primary key not null unique,
  name varchar(100) not null,
  unitCost decimal(6,2) not null,
  referenceUnitPrice decimal(6,2) not null,
  productType varchar(4),
  category   varchar(30),
  series varchar(30),
  imageFileName varchar(20)
) ENGINE=InnoDB;

create table inventory(
  storeID int not null,
  productID int not null,
  number int,
 primary key(storeID, productID),
  foreign key (storeID) references store (storeID),
  foreign key (productID) references product (productID)
) ENGINE=InnoDB;

create table orders(
  orderID int primary key not null unique auto_increment,
  customerID int not null,
  salesmanID int(5) not null,
```

```
  storeID int not null,
  date date not null,
  status varchar(20) not null default 'unfinished',
  foreign key (customerID) references customer (customerID),
  foreign key (salesmanID) references storeEmployee (employeeID),
  foreign key (storeID) references store (storeID)
) ENGINE=InnoDB;

create table orderDetail(
  orderDetailID int primary key not null unique auto_increment,
  orderID int not null,
  productID int not null,
  number int not null,
  unitPrice decimal(6,2) not null,
  foreign key (OrderID) references orders (OrderID),
  foreign key (productID) references product (productID)
) ENGINE=InnoDB;

create table inventoryOrder(
  inventoryOrderID int primary key unique not null auto_increment,
  orderID int not null,
  date date not null,
  status varchar(11) not null default 'unsent',
  foreign key (orderID) references orders (orderID)
) ENGINE=InnoDB;

create table inventoryOrderDetail(
  inventoryOrderDetailID int primary key unique not null auto_increment,
  inventoryOrderID int not null,
  productID int not null,
```

```
  storeID int not null,
  numberChanged int not null,
  foreign key (inventoryOrderID) references inventoryOrder
(inventoryOrderID),
  foreign key (productID) references product (productID),
  foreign key (storeID) references store (storeID)
) ENGINE=InnoDB;

create table orderPayment(
  orderID int primary key not null unique auto_increment,
  amountShouldPay decimal(10,2) not null default 0.00,
  amountPaid decimal(10,2) not null default 0.00,
  status varchar(10) not null default 'unfinished',
  foreign key (orderID) references orders (orderID)
) ENGINE=InnoDB;

create table orderPaymentTransaction(
  orderPaymentTransactionID int primary key not null unique auto_increment,
  orderID int not null,
  paymentAmount decimal(10,2) default 0.00 not null,
  foreign key (orderID) references orders (orderID)
) ENGINE=InnoDB;
```

3.3.2 Data Dictionary and Normal form identification

Because Mysql does not support check constraint, the check constraints below were not implemented in the DBMS.

Employee (BCNF, employeeID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check

employeeID	int	5		yes	yes	yes		
firstName	varchar	30		yes				
lastName	varchar	30		yes				
jobTitle	varchar	14	salesman	yes				=salesman =store manager =region manager
street	varchar	200						
City	varchar	200						
State	varchar	10						
zipcode	int	5						
email	varchar	200						
salary	decimal	(7,2)						

Login (BCNF, employeeID->all)

Although userType is the same as jobTitle in Employee table, it is better to store in login table again to avoid join operation, since login may be a frequent operation.

column name	data type	length	default	not null	unique	primary key	foreign key	check
employeeID	int	5		yes	yes	yes	Employee	
password	varchar	15		yes				
userType	varchar	14	salesman	yes				=salesman =store

								manager =region manager

Region (BCNF, regionID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
regionID	int			yes	yes	yes		
name	varchar	30		yes				
regionManagerID	int			yes			Employee	

Store (BCNF, storeID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
storeID	int			yes	yes	yes		
managerID	int	5		yes	yes		Employee	
regionID	int			yes			Region	
name	varchar	30		yes				
street	varchar	200						
City	varchar	200						
State	varchar	10						
zipcode	int	5						

salesmanNumber	int		0	yes				>=0

StoreEmployee (BCNF, employeeID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
employeeID	int	5		yes	yes	yes	Employee	
assignedStoreID	int			yes			Store	

Customer (BCNF, customerID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
customerID	int			yes	yes	yes		
name	varchar	100		yes				
street	varchar	200						
city	varchar	200						
state	varchar	10						
zipcode	int	5						
customerType	varchar	10	individual	yes				=individual =business

BusinessCustomer (BCNF, customerID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check

customerID	int			yes	yes	yes	Custom er	
category	varchar	30		yes				
grossAnnualInco me	decimal(10, 2)							

IndividualCustomer (BCNF, customerID->all)

column name	data type	lengt h	defau lt	not nul l	uniqu e	primar y key	foreign key	check
customerID	int			ye s	yes	yes	Custom er	
gender	varchar	6						=male =female
Age	int							
annualInco me	decimal(10, 2)							
marriageStat us	varchar	10						=single =married =divorced =common-l aw

Product (BCNF, productID->all)

column name	data type	lengt h	defaul t	not nul l	uniqu e	primar y key	foreig n key	check
productID	int			yes	yes	yes		
name	varchar	100		yes				
unitCost	decima l	(6,2)		yes				

referenceUnitPrice	decimal	(6,2)		yes				
productType	varchar	4						=home =work
category	varchar	30						
series	varchar	30						
imageFileName	varchar	20						

Inventory (BCNF, (storeID, productID)->number)

column name	data type	length	default	not null	unique	primary key	foreign key	check
storeID	int			yes		yes	Store	
productID	int			yes		yes	Product	
number	int							>=0

Order: (BCNF, orderID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
orderID	int			yes	yes	yes		
customerID	int			yes			Customer	
salesmanID	int	5		yes			Employee	
storeID	int			yes			Store	
Date	date			yes				

status	varchar	20	unfinished	yes				=unfinished =payment received =shipping =finished =canceled

OrderDetail (BCNF, orderDetailID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
orderDetailID	int			yes	yes	yes		
orderID	int			yes			Order	
productID	int			yes			Product	
number	int			yes				>0
unitPrice	decimal	(6, 2)		yes				

InventoryOrder: (BCNF, inventoryOrderID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
inventoryOrderID	int			yes	yes	yes		
orderID	int			yes			Order	
Date	date			yes				
status	varchar	11	unsent	yes				=unsent =partly sent

								=finish

InventoryOrderDetail: (BCNF, inventoryOrderDetailID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
inventoryOrderDetailID	int			yes	yes	yes		
inventoryOrderID	int			yes			inventoryOrder	
productID	int			yes			Product	
storeID	int			yes			Store	
numberChanged	int			yes				>0

OrderPayment: (BCNF, orderID->all)

check constraint: amountShouldpay >= amountPaid

column name	data type	length	default	not null	unique	primary key	foreign key	check
orderID	int			yes	yes	yes	Order	
amountShouldPay	decimal	(10,2)	0.00	yes				>0
amountPaid	decimal	(10,2)	0.00	yes				
status	varchar	10	unfinished	yes				=unfinished =finished =canceled

OrderPaymentTransaction: (BCNF, orderPaymentTransactionID->all)

column name	data type	length	default	not null	unique	primary key	foreign key	check
orderPaymentTransactionID	int			yes	yes	yes		
orderID	int			yes			OrderPayment	
paymentAmount	decimal	(10, 2)	0.00	yes				>0

4. UI Design & Database Connection

4.1. UI Design

Two main roles (customers or employees) may use this system. For all the customers, they can only browse the products and sort the products by product type and category. Also customer is able to search the product by its name. Employees may have different job titles, so there are three types of employees: salesman, store manager, region manager. Salesman and store manager can create or check orders for customers. Store manager can view customer information, add new customer and view products and inventory. In addition, he can also see the top category of the product and its sales, and the top five products that sales most. Region manger is able to see the store information in this region and view the customer and salesman list, add salesman, and view products and inventory. Region manager can also see the sales comparison among all the regions.

Interfaces	Description
Index Page	Customers browse all the products on this page and can sort the product by category and product type. In addition, customer can search the product by its name. Employees can login on this page and it will redirect to "salesman page", "store manager page", or "region manager page" according to its job title.
Login Page	Employee can log in via this page.
Salesman Page	This page is the homepage of salesman. It also lists all the products and can be sorted by category and product type. In the navigation bar, it displays the login information. Salesman can also create order, check order, logout by clicking the link.
Create Order Page	This page displays an order to be filled. It displays order date and salesman name. Then customer name need to be filled. A modal will popup, salesman can select the customer from the list. Then products also need to be chosen from the product list. After clicking add button, the product will be added to the order. The quantity can be modified. The product information and total price of the order will be shown.
Check Order Page	This page will list all the orders generated by this employee ordered by order date. In each order, it will display the product name, unit price, quantity, and total price. The store employee can change the order status (unfinished, payment received, shipping).
Store Manager Page	This page displays the store information that the manager in charge of. The store manager can click the two buttons: "see customers", and "see product and inventory".
Customer List Page	This page displays the basic information of all users, and

	store manager can choose to see individual customers or business customers. There is a button to view the order history of each customer and another button to add a new customer
Add Customer Page	This page enables store manager to add new customer into the database.
Customer Order Page	This page displays all the orders of one customer. For each order, it shows order date, order ID, customer ID, order status and the information of each product.
Product List page	This page displays all the products information including inventory information. Store manager can see some statistics of each product: the business buying most, the sales and profits of this product. These statistics are displayed in modals.
Region Manager Page	This page displays region manager information and the information of all stores in this region. There are three buttons in this page, view customers, view product and inventory, and sale comparison. Sale comparison will be shown in a modal.
Salesman List Page	This page shows all the salesman information in this region. The region manager can modify the information. And there is a button using to add a new salesman.
Add Salesman Page	This page enables region manager to assign new salesman to a store.

4.2. Front-end to Back-end Connection

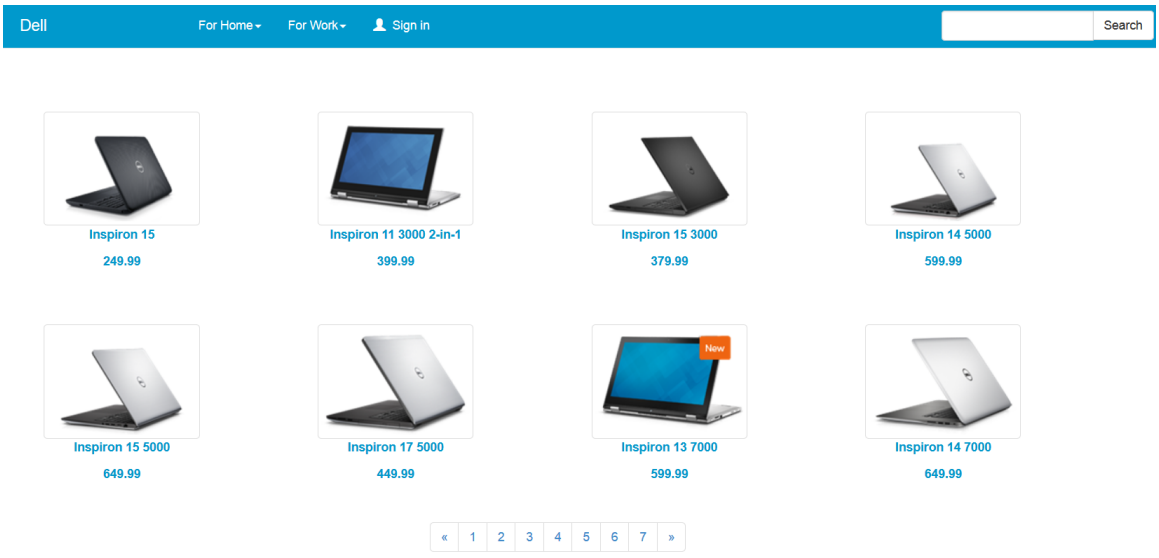
This application mainly uses php to connect to the back-end. Php has a large number of extensions to help it connect to database. By using the MySQLi extension, it offers an effective way to transfer front-end data to back-end database. To use MySQLi, we first need to create a new mysqli

connection object

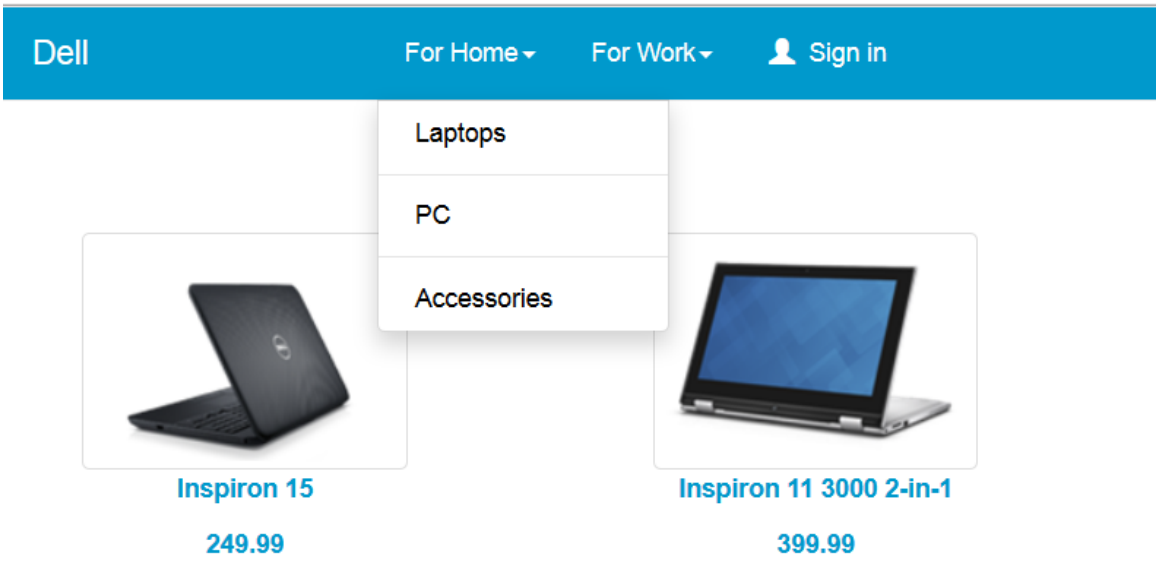
5. Implementation

5.1. Customer Browsing

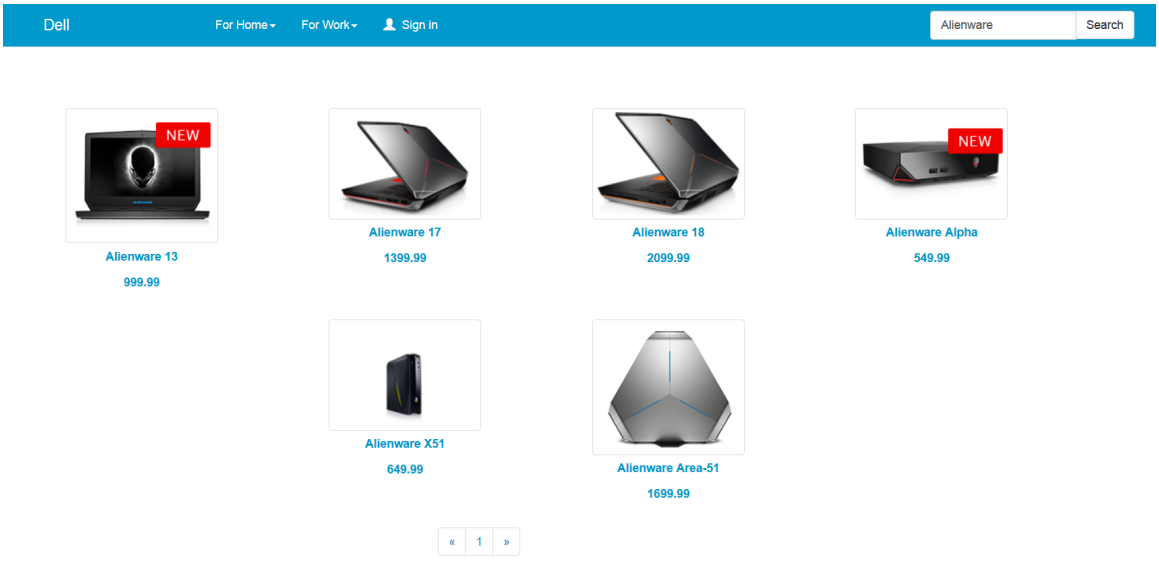
At index page, customers and other employees can browse products that our store have and search products by key words or by types. Also, if there are more than 8 products, we will help you paging the results and you can browse the remaining results in next pages.



By clicking “For Home” and “For work” buttons at the top bar, customers can further choose the product type they are looking for. For example, the “For Home” has a list of types like laptops, PC, and accessories.

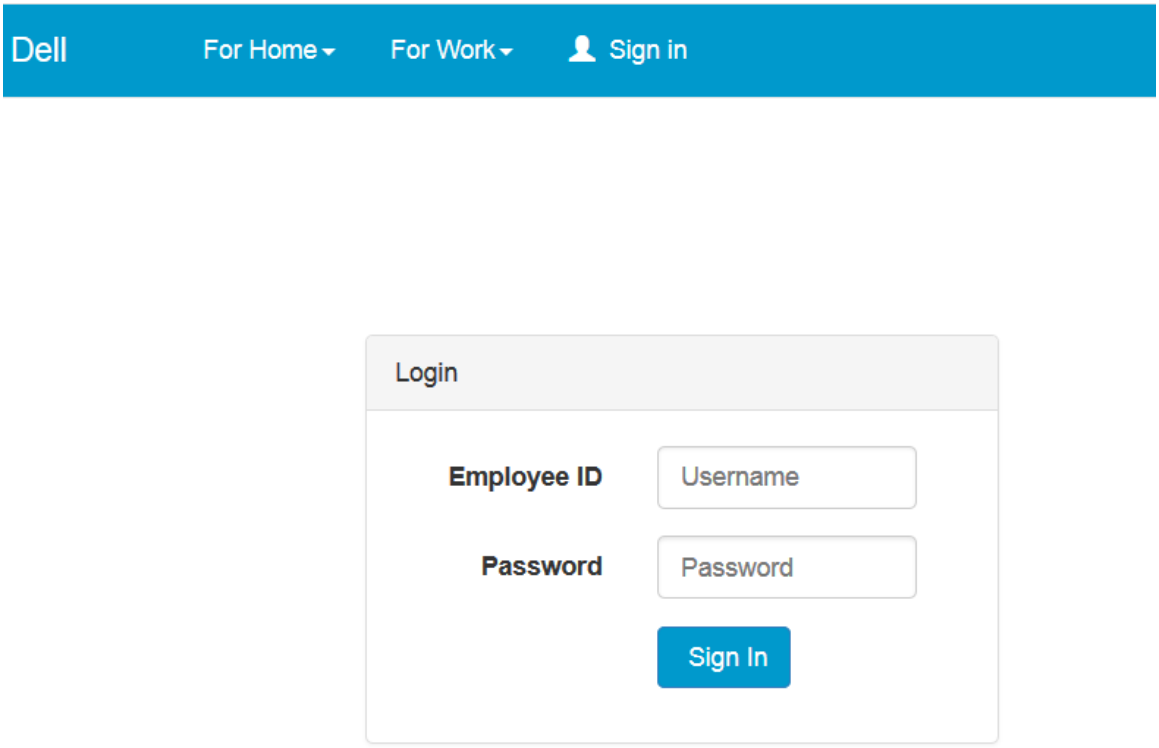


Furthermore, if customers want to search product by themselves, they can use the search function at the right top corner. For example, if a customer wants to search product whose name has “Alienware”, they can type Alienware and click search button, and the result will be like this.

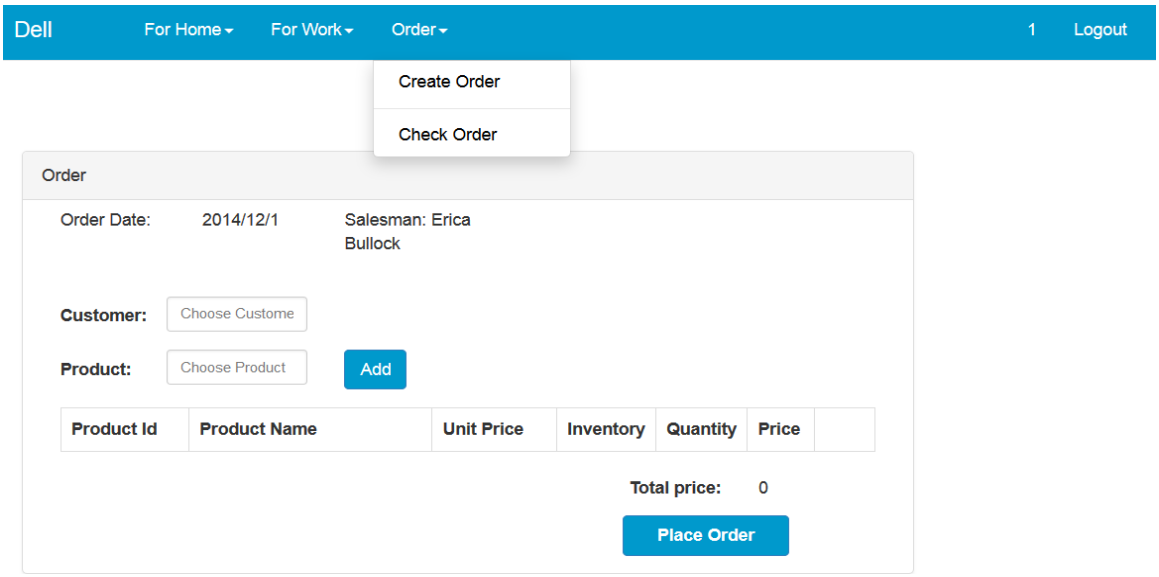


5.2. Employee Sign in System

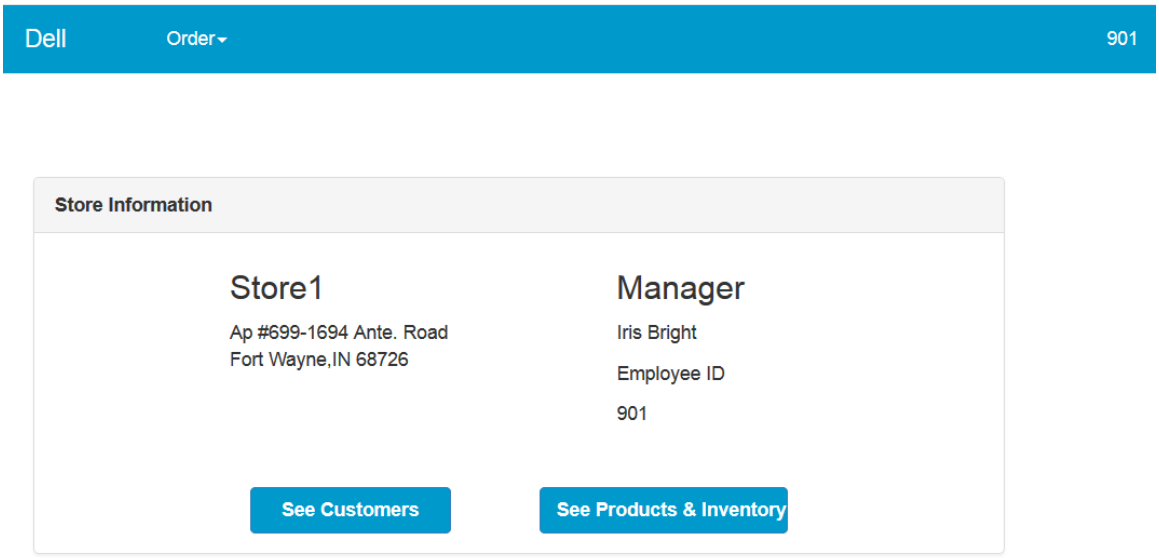
By clicking the Sign in button at the top bar, employees can use their employee ID and password to sign in.

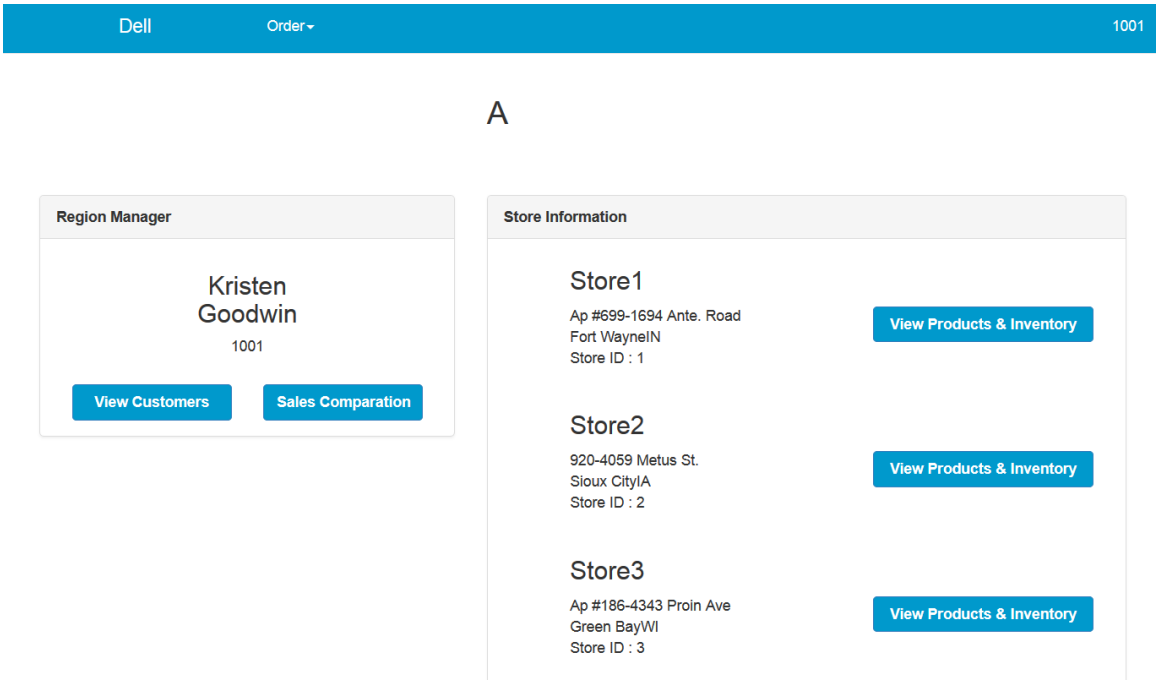


Depend on employees' type, login page will direct to different pages and employee can make different operations in their pages. For example, if a salesman login, he will be direct to salesman page. And at this page, he can create orders and check orders.



On the other hand, if the employee is a store manager, his page will pay more attention about customer management and inventory management; and if the employee is a region manager, his page will focus on stores management and profit management. Here are the examples of these two type pages.

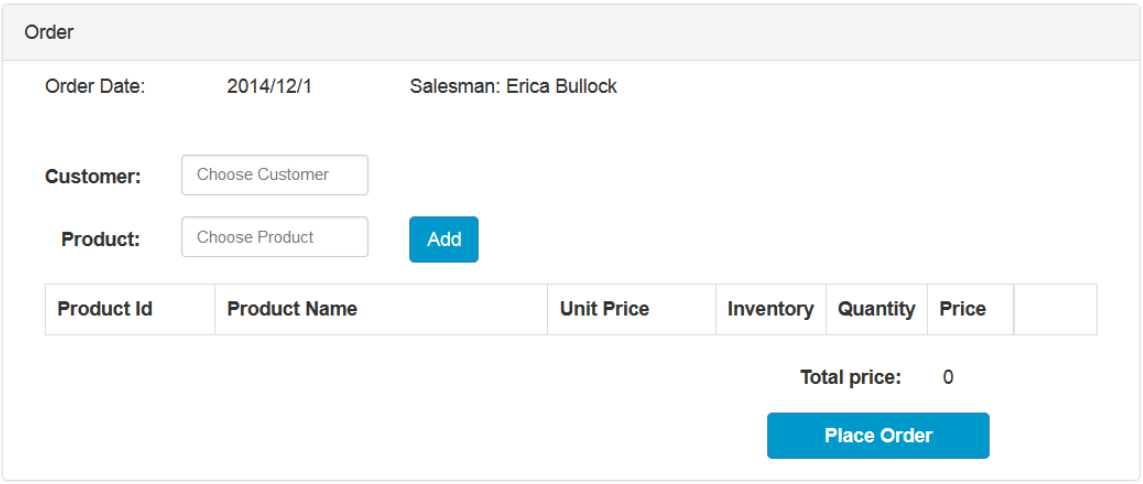




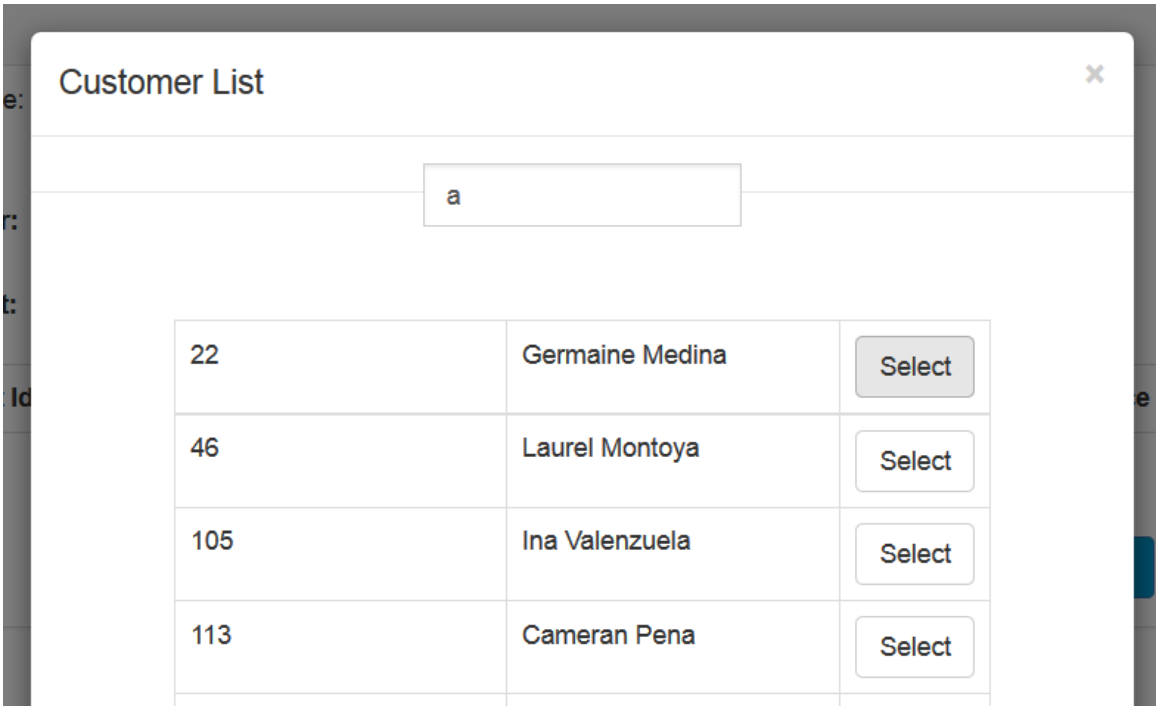
A

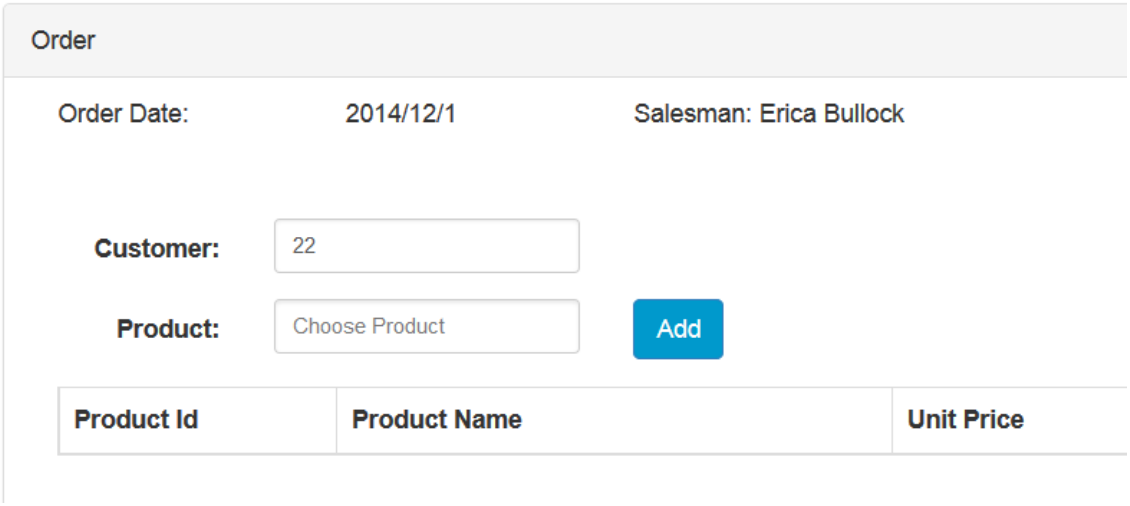
5.3. Create Order

Employee can create orders and check orders at their page. Here is an example for create order.

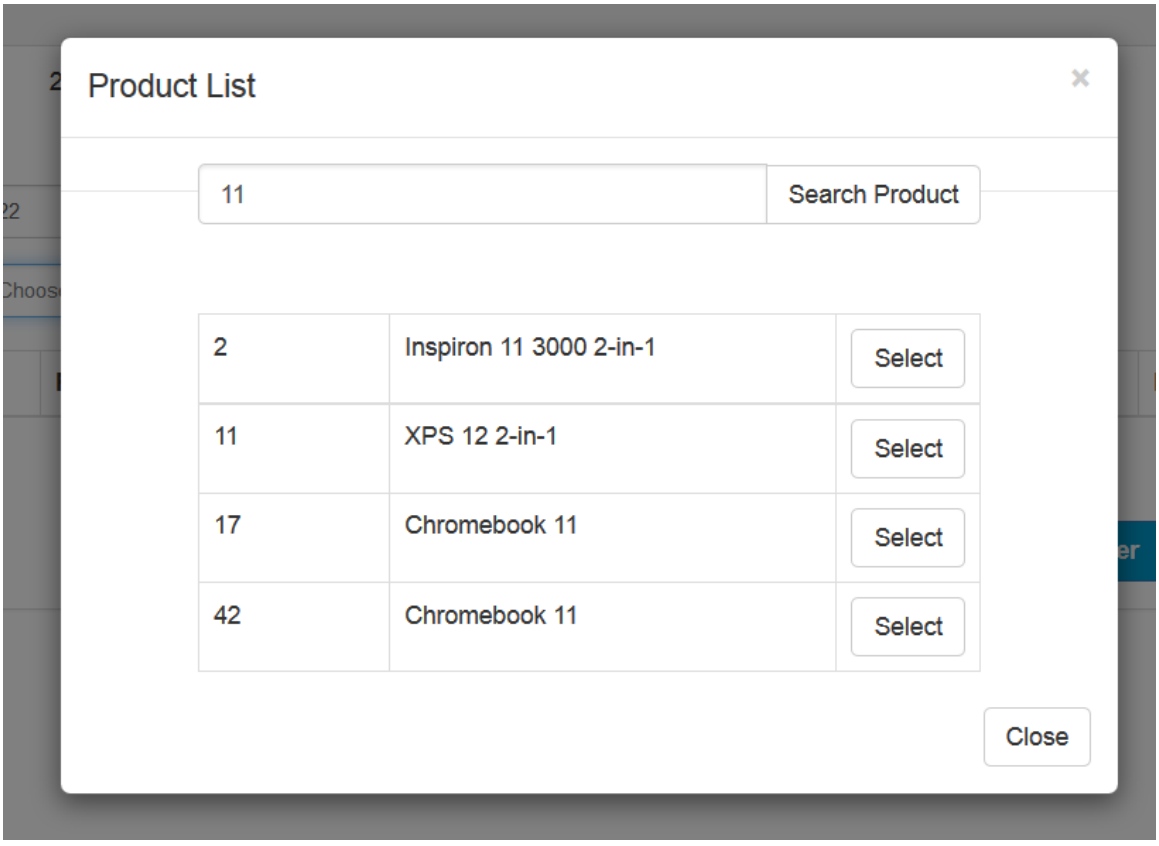


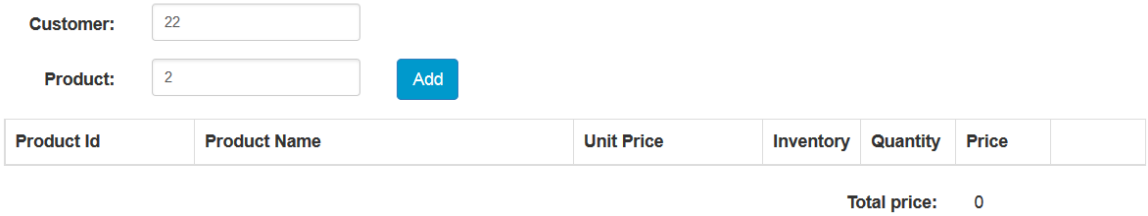
The employee first needs to click input box at the customer line. After clicking it will pop up an input box. In the box, employees can search the customer by their name. When they find the right customer, they need to click on the "select" button after the right customer, and it will return the customer's ID to the original input box after customer line.



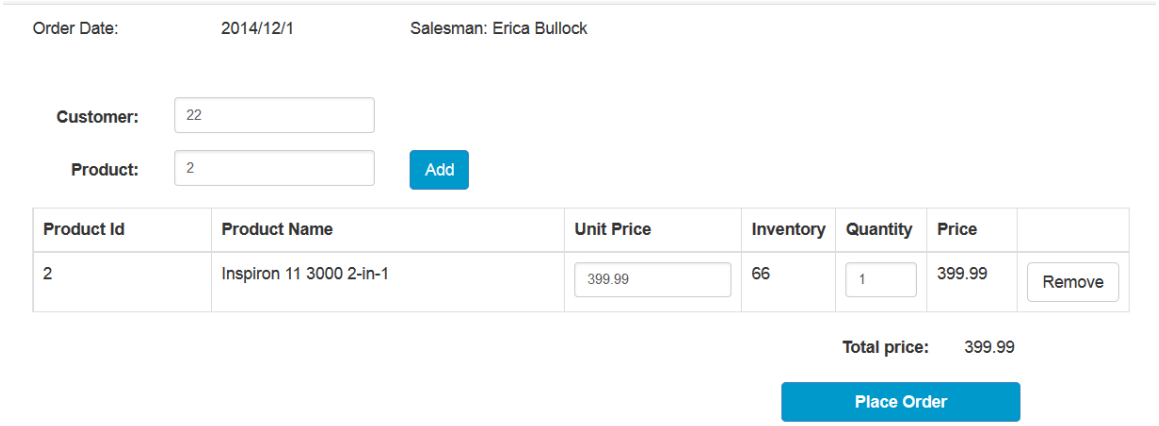


After selecting the customer, the employee needs to select the product which customer wants to buy. Almost the same way as selecting customer: click the input box at product line and select ate the new window. And it will return the product ID to the original input box.





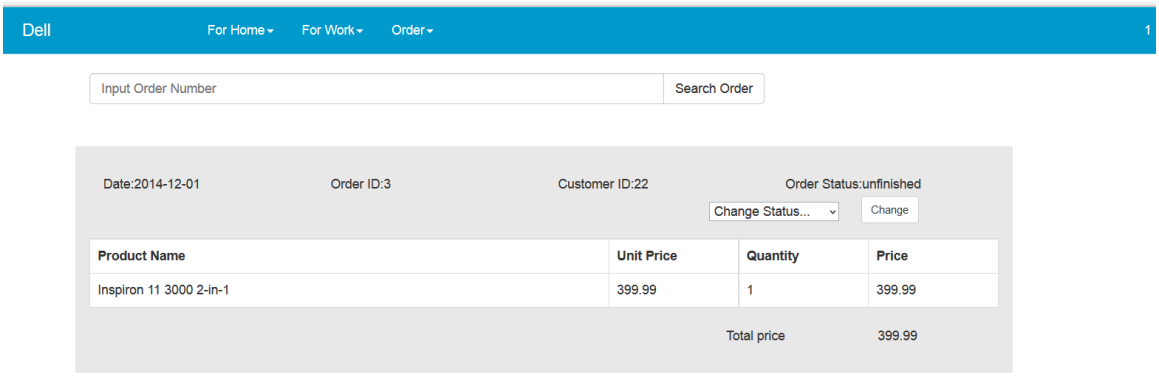
Then employees just need to click the add button and product will add to the product list below.



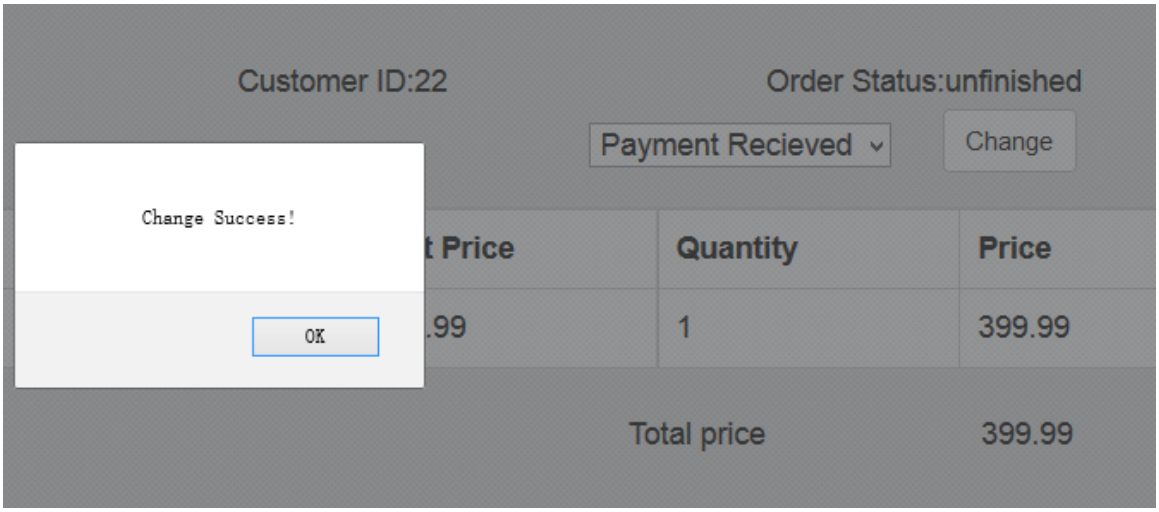
And if there is no other products need to be added, the employees can click the place order button. And the order will be created and it can be checked at the check order page.

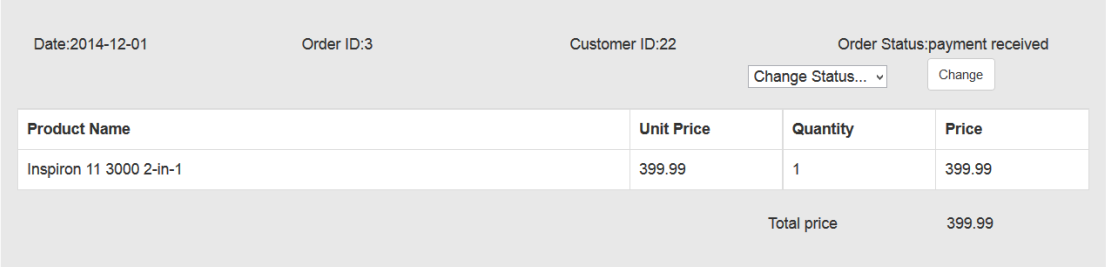
5.4. Check Order

Right after creating order, employees can check their orders at check order page.



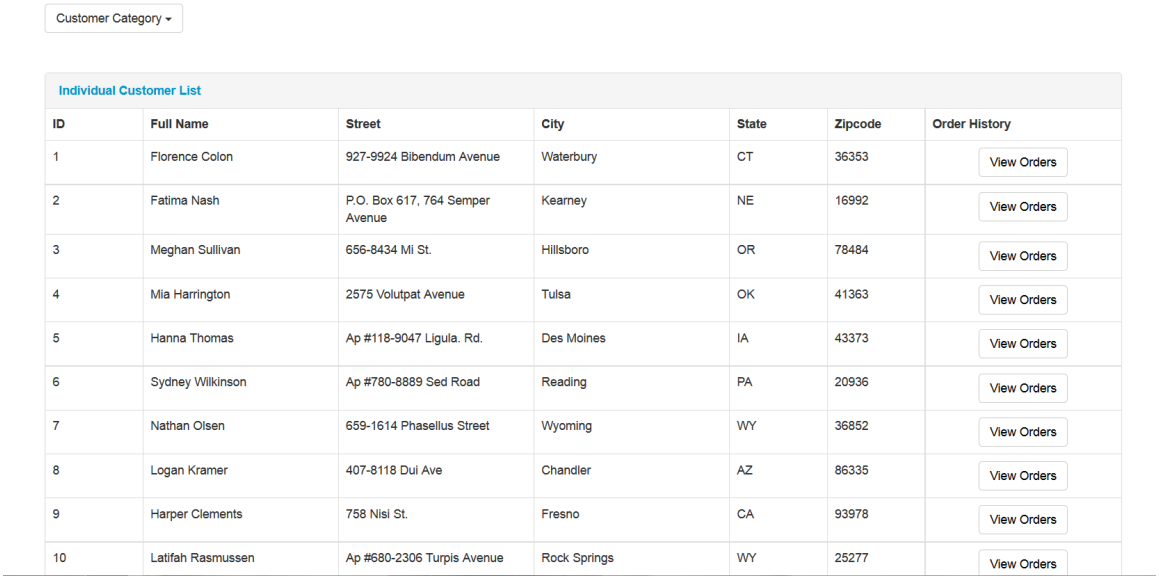
Every order has 4 different statuses. The original status is “unfinished”, and it can be changed to “payment received”, “shipping”, “cancel”. Employee can change the status by choosing the “Change Status” and then click the change button. And if success, it will pop up a success message. Then order status will change to payment received.



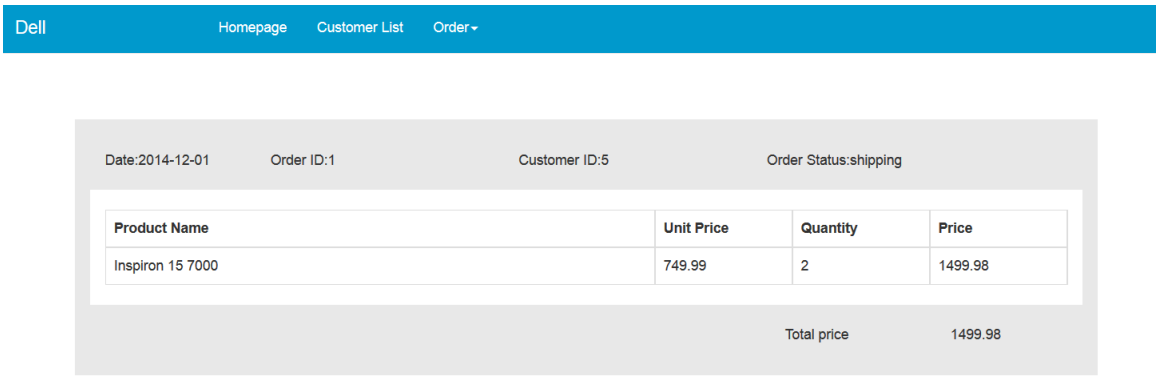


5.5. Check Customer

Employees like store manager and region manager can check customer at their page. At the customer list page, it will list all the customers we have at the database. It shows customers’ information like name, address, annual income (only available in business customers) and order history.

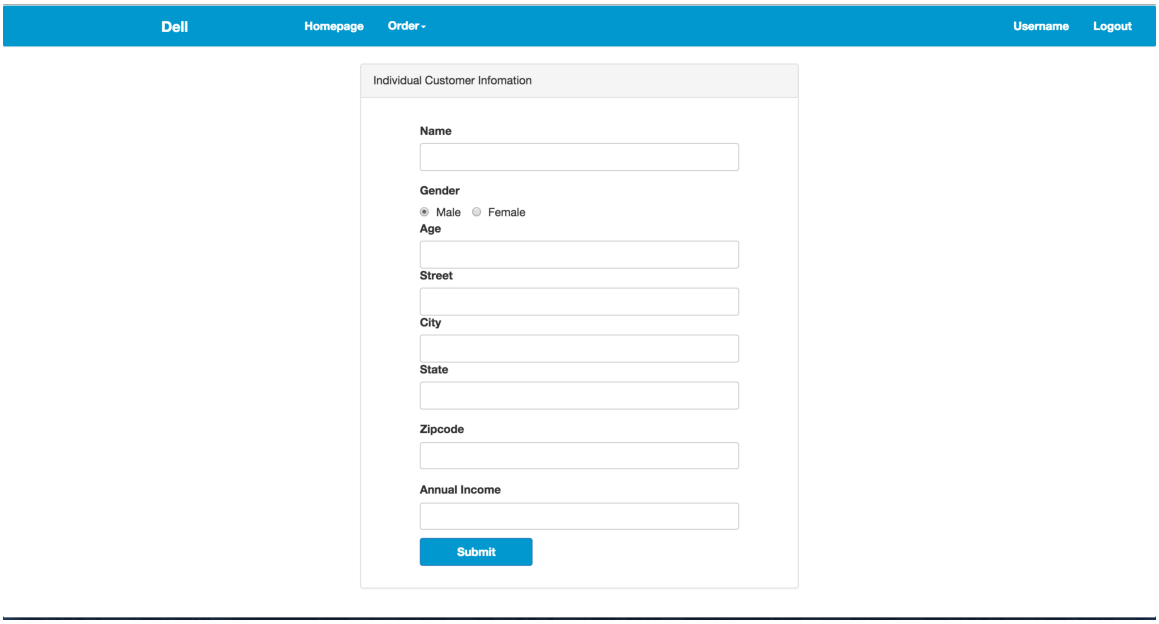


If employee wants to check a customer's order history, just click the "view orders" button. And it will direct to a new page lists all the orders this customer made. Here's the order which related to customer ID = 5;

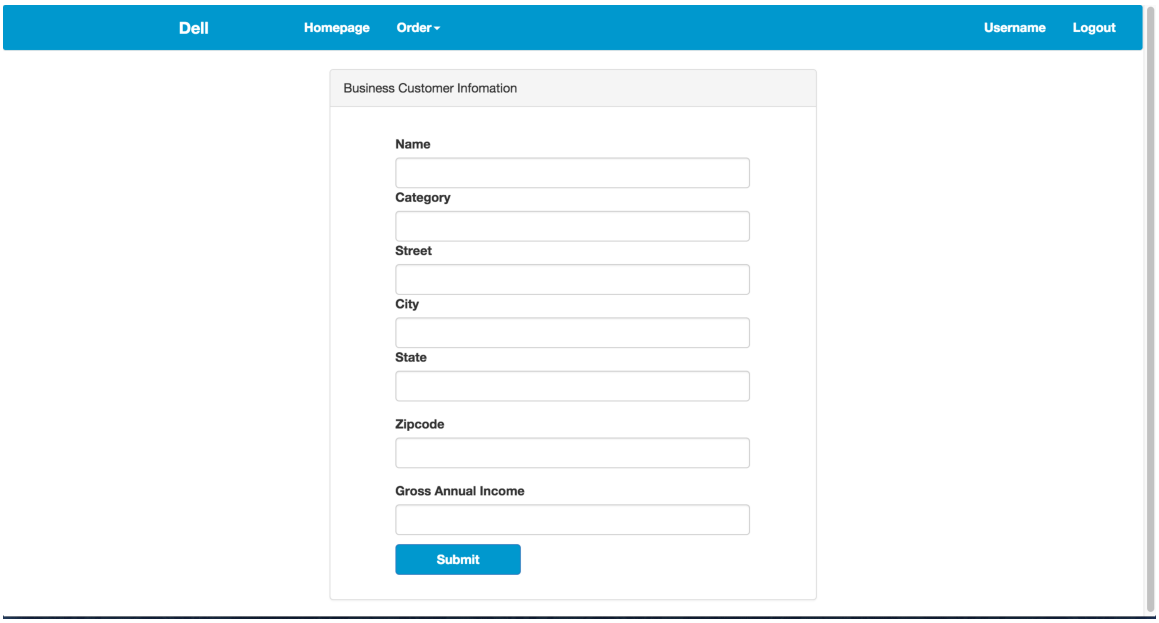


5.6. Add New Customer

Store manager can add a new individual customer into the database. He needs to fill in all the information and click submit button.

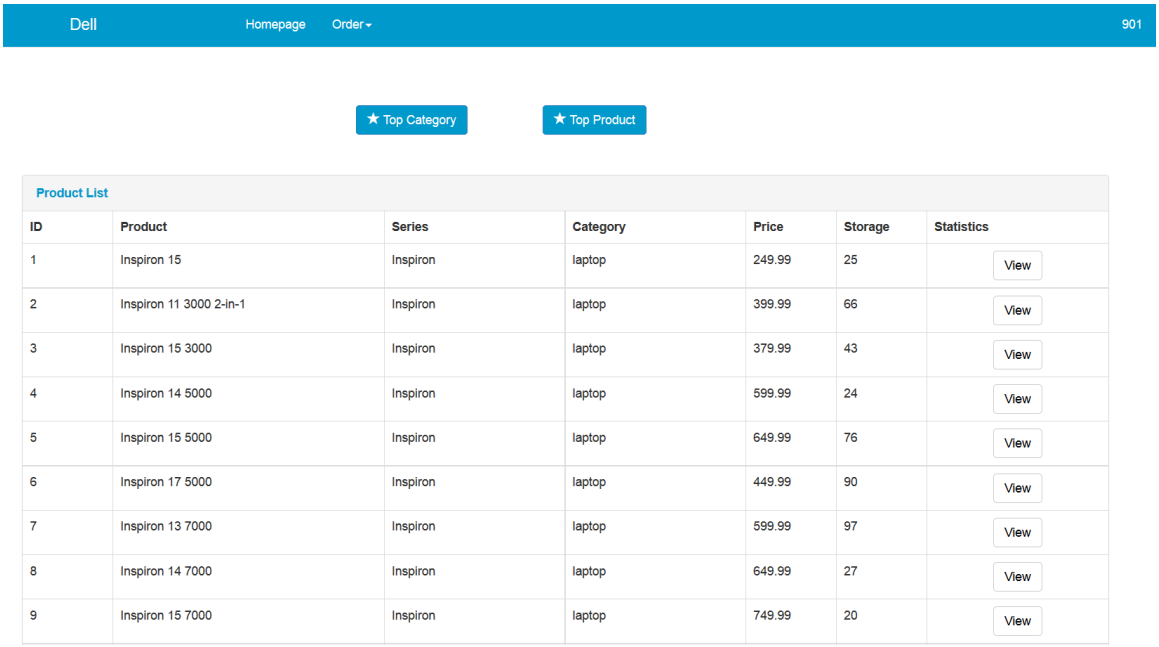


Store manager can also add a new business customer into the database. He needs to fill in all the information and click submit button.

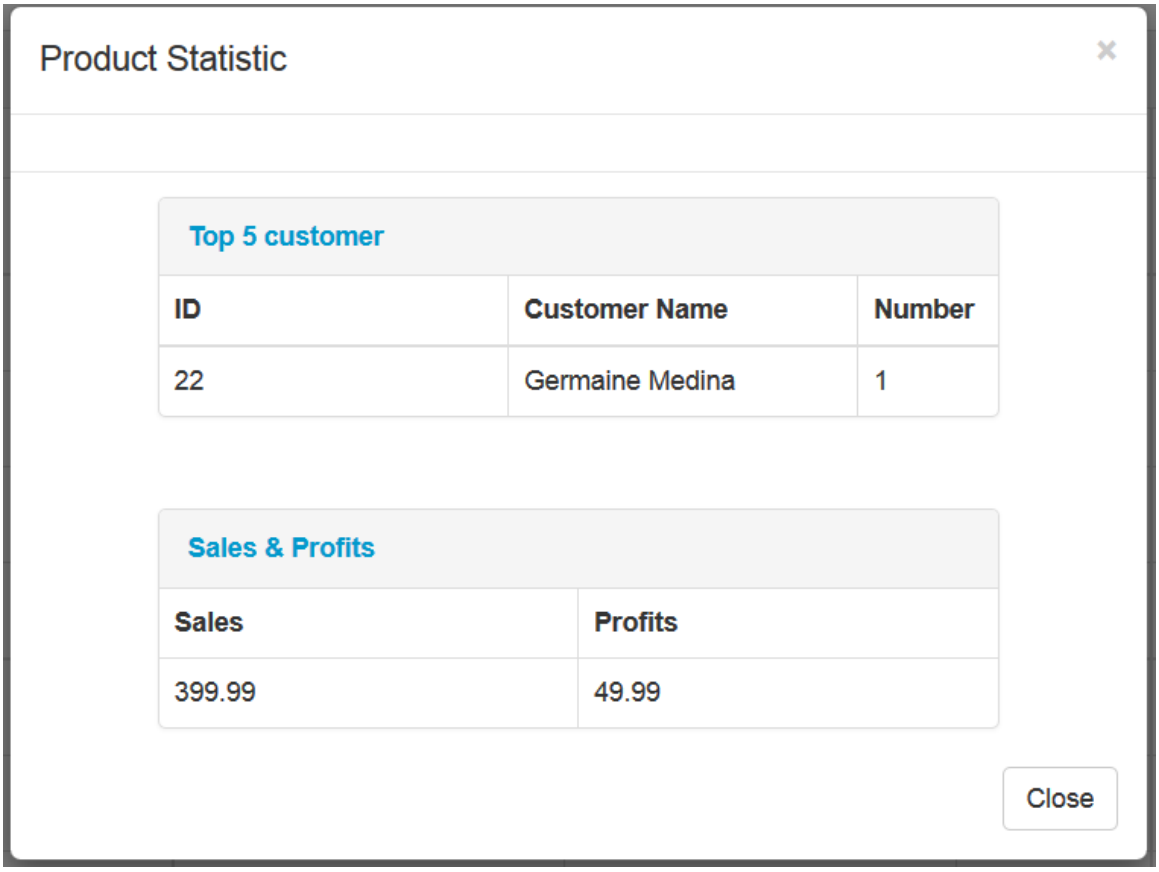


5.7. Check Inventory

The store manager can check the inventory in his store at the store product list page. Picture below is the product list page from store manager ID 901.

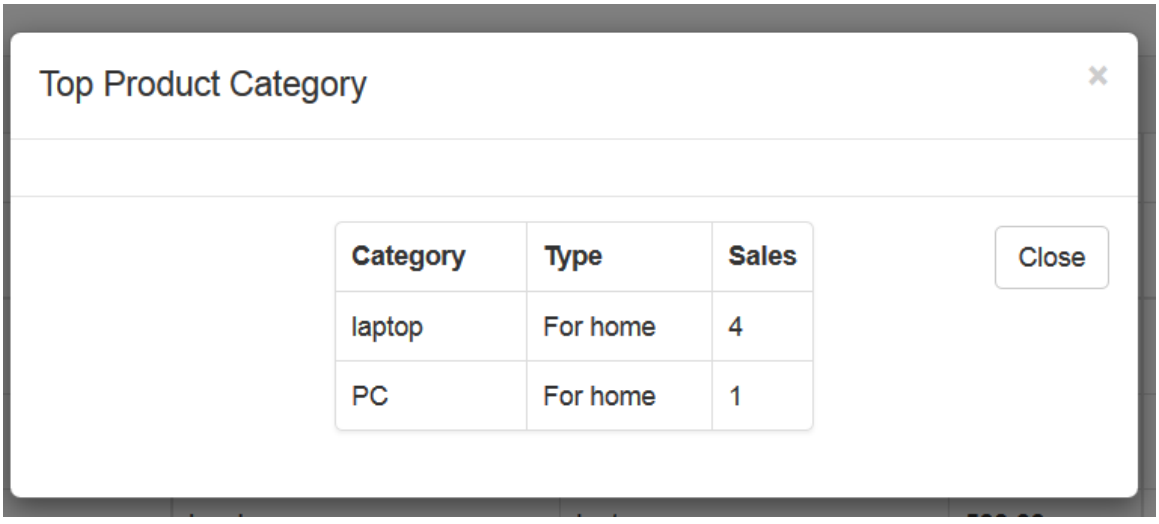


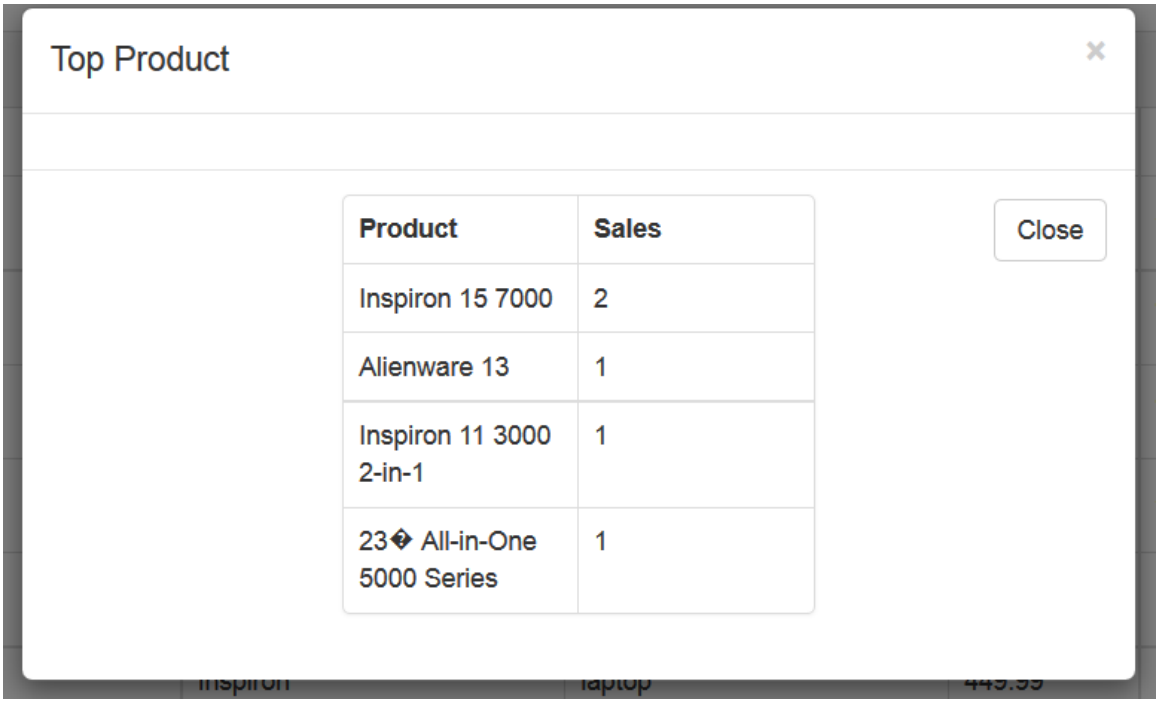
Also, store managers can view the statistics of every product. Click the View button at the end of every product line. It will count the sales and profits and the customers who buy most.



5.8. Top Product Statistic

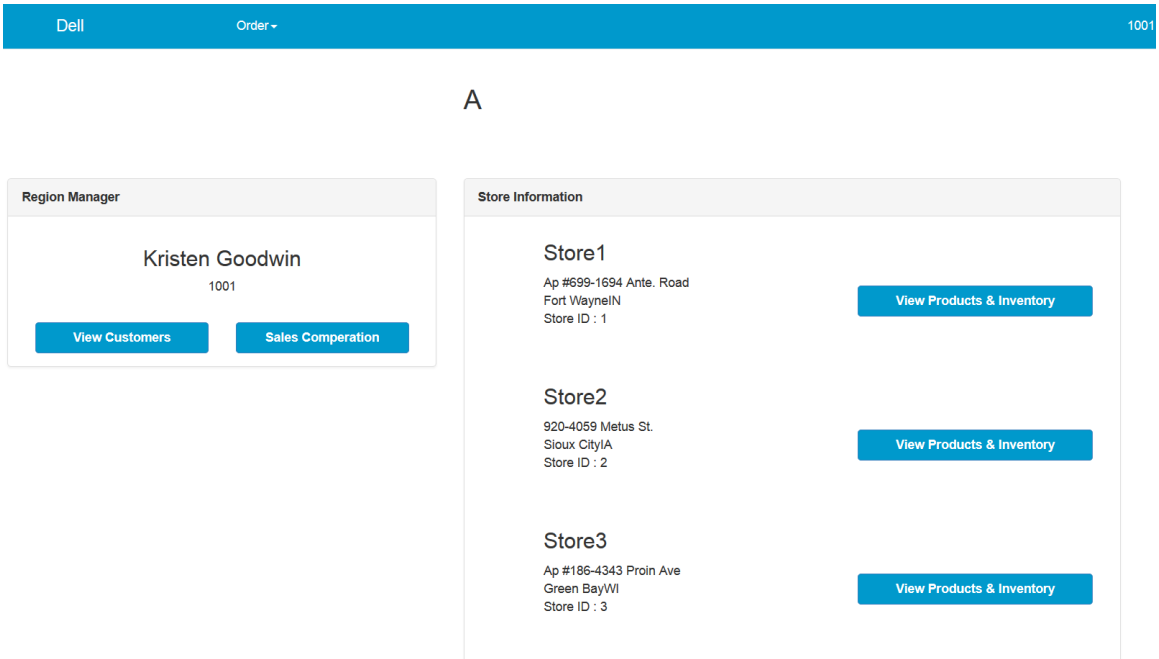
By clicking the “Top Category” and “Top Product” buttons at the top of product list page, it will show you which category and product sells best.





5.9. Region management

The region manager can log in and manage stores in his region. Here's an example from region manger 1001:

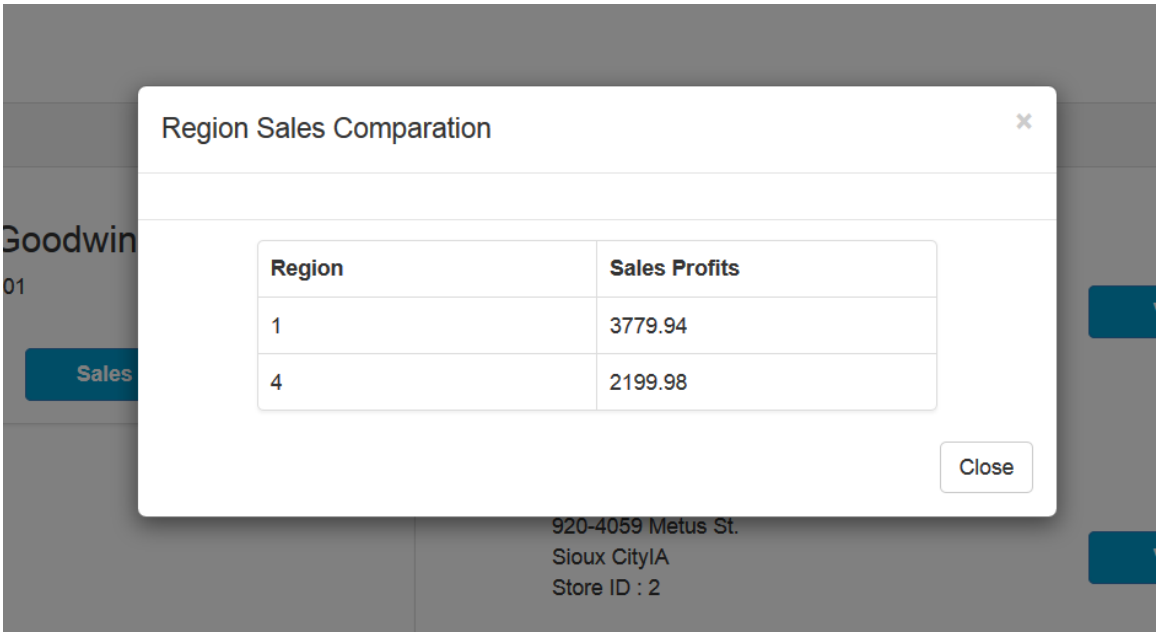


He can check every store's sales and inventory listed on the right. It will direct

to store product list page and it looks like pages in part 5.6.

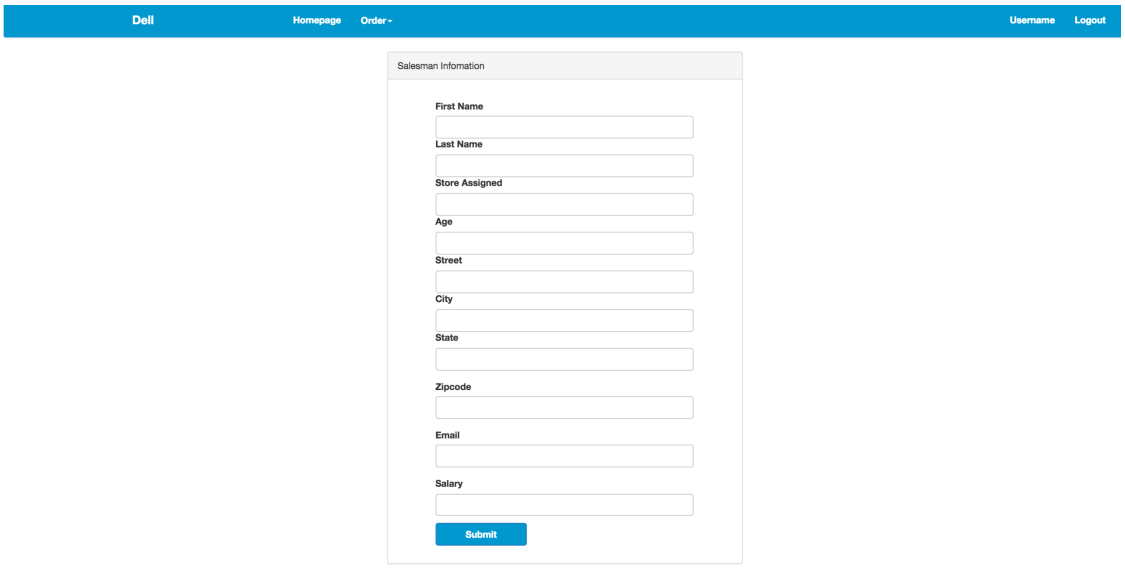
5.10. Region Sales Compare

By clicking the Sales comparison at left, it shows the total sales number group by different regions.



5.11. Add New Salesman

Region manager can assign a new salesman to one store. He needs to fill in all the information and click submit button.



6. Exception Detection & Handling

6.1. Inventory Check

When create a new order, every time employee add a product to the table, it will show the inventory of this product in the employee's store. If the employee change the amount to a number bigger than the inventory number, it will pop up an alert warning the employee that product amount should less than or equal to inventory amount.

6.2. Order Status Change Check

As mentioned before, every order has four status, which are unfinished, payment received, shipping and cancel. These four statuses have logical relations between each other. If the status is unfinished, it can only be changed to payment received or cancel. After the status being changed to payment received, it can only be changed to shipping then. Finally, if the status is shipping or cancel, it cannot be changed to any other status.

7. Limitations & Improvements

In reality, distributors may transfer products, between stores when a customer want to buy a product, which is out of stock in the store that the customer presents, but a nearby store has that product. We may create two tables to record the tranferOrder and tranferOrderDetail to manage this transaction.

As mentioned in 3.2, customers may pay for the order in several times, but the system only supports one-time full amount payment. However, the database design is compatible for the future system update.

8. Experience of inserting large amount of data

Phpmyadmin is a good tool to manage mysql, but it is very slow when inserting large amount of data. I learned how to use Mysql console to manage Mysql DBMS when I was doing this project, and it can be efficient to insert large amount of data when set autocommit=0 at the beginning, and manually commit

after import the sql file that insert a large amount of data. It is efficient because I asked the DBMS not to commit after every “insert” statement, but commit all the changes of inserting the large amount of data at once. It will cost much less time.

Statement: All images in the system are from www.dell.com