

Team members: Bruno Rwabukumba & Xiang Di Su

FINAL PROJECT: ONLINE SNEAKER STORE (FOOTWEAR)

Footwear.ca is an online shoe store that caters to the average sneaker enthusiast. The website will focus on the online store part where users will be able to view available products (Shoes and accessories) and the quantity for each product. Users will be able to view upcoming sneaker release dates as well as ongoing promotions that footwear.ca is offering.

The goal of our system will be to keep track of inventory, payment method, track clients order, display price information, product description, etc. Although users will not be able to complete the orders as the payment option won't be implemented, users will still be able to do the following:

- Create an account and choose the option to receive promo email
- Select products they wish to add to the cart
- Add products to their favorite items
- View their cart and the total of their order based on the selected products
- Proceed to checkout
- Select both shipping and payment methods

We will model our application and functionalities similar to that of footlocker.ca. We estimate that we will spend a total of around 100 hours building this project. Therefore, 50 hours per team member.

User stories

1. Card: A user can create an account

Conversation: The information to create an account will be added to the user database

Conformity: When user presses the “create” button 1) Fetches the texts in textboxes 2) inserts the data in our user table

Implementation: User inputs will be saved to the database

INSERT INTO user VALUES(Texts from Textboxes);

[Footwear](#) [Products](#) [About us](#) [Create your Account](#) [Login](#)

Create your Account

Username	<input type="text"/>
Password	<input type="password"/>
Email	<input type="text" value="aime_bruno@yahoo.com"/>
Address	<input type="text"/>
Phone	<input type="text"/>

Sign Up

2. Card: A user can delete an account

Conversation: The information to delete an account will be obtained the user's login information

Conformity: When user press the “delete” button 1) Fetches the user's username 2) Fetches the customerID associated with the username 3) removes the data in our database table that's associated to this customerID

Implementation: Will place the CustomerID into a variable called userID.

DELETE FROM user WHERE CustomerID == userID;

3. Card: A user can log in

Conversation: The information to log in will be compared to the user table.

Conformity: When the user presses “login” button 1) Fetches the textboxes inputs 2) Compares if the information matches. Users will be logged in to their own personal account if information matches.

Implementation: username and password will be placed into variables userName and password, SELECT FROM user

WHERE username == userName and password == password;

FootwearProductsAbout us

Create your AccountLogin

Email

Username

aime_bruno@yahoo.com

Password

.....

Log In

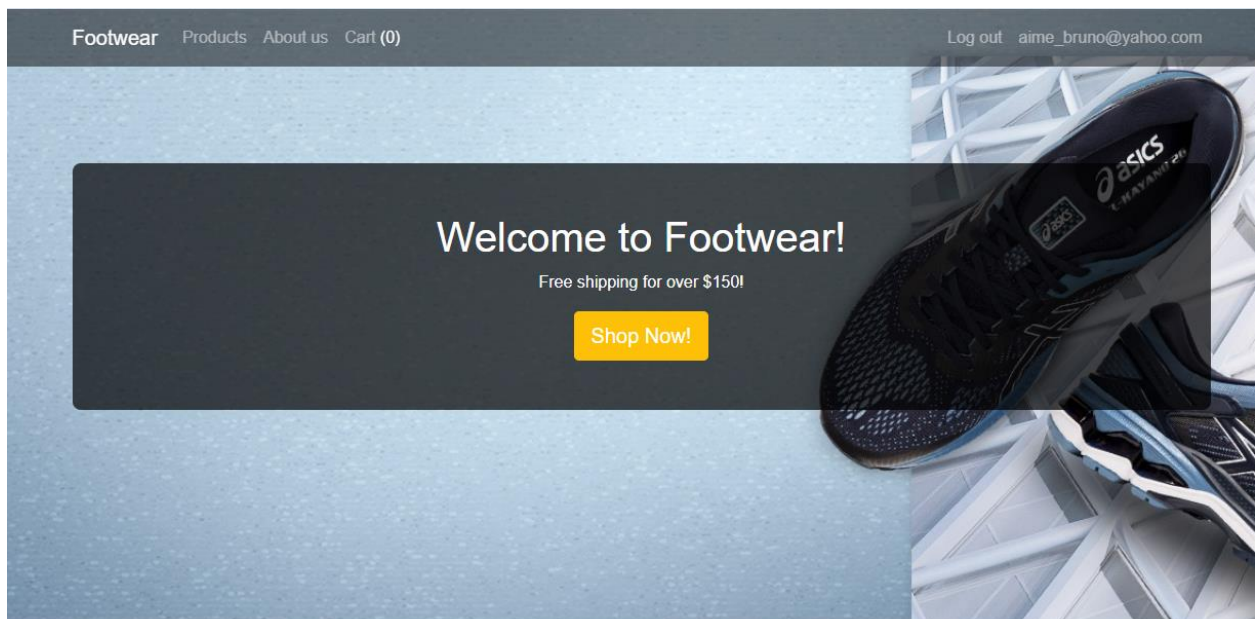
Copyright©Footwear | All Rights Reserved | Contact Us at: +514 123 4567

4. **Card: A user can log out**

Conversation: The use will be logged.

Conformity: There will be no further action required except that the user will not be able to view their profile, previous orders, etc.

Implementation: Once the user clicks the logout button, previous orders page and profile page will be blank. They will instead see a message telling them to log in.



5. **Card: A user can view their profile**

Conversation: The information will be displayed from the user table in our database

Conformity: We will display the data by 1) Fetching the information of the logged in user from the user table and 2) Display it in labels of the profile page.

Implementation:

```
var newUser = from x in db.users(x => x.customerID == userID) select
x.FirstOrDefault();
nameLabel.text = newUser.Name; addressLabel.text = newUser.Address; etc
```

6. Card: A user can edit their profile

Conversation: The details will be updated in the user table of our database.

Conformity: When the user presses the “edit” button 1) Fetches the textboxes inputs 2) updated the user table with the new inputs.

Implementation: New input will be place in various variables,

```
UPDATE TABLE user column = new values WHERE customerID == userID;
```

7. Card: A user can view all products

Conversation: The information will be displayed from the products table in our database

Conformity: We will display the data by 1) Fetching the product's name, description, image path, and price 2) Display it on a table

Implementation: The table will have labels for name, description, and price, and image element to hold the image. We will get a list of all products then assign each label to the corresponding product info.

```
Var products = (FROM x in products
SELECT x).ToList();
```

Footwear

Products


About us

Cart (0)

Log out

aime_bruno@yahoo.com


Go back to home / Products



Comme des garcon
\$75

Add To Cart


Edit



Van Strike-edited
\$4500

Add To Cart


Edit



Mickey Sneakers
\$65

Add To Cart

Edit



Air Nike Black
\$50

Add To Cart

Edit

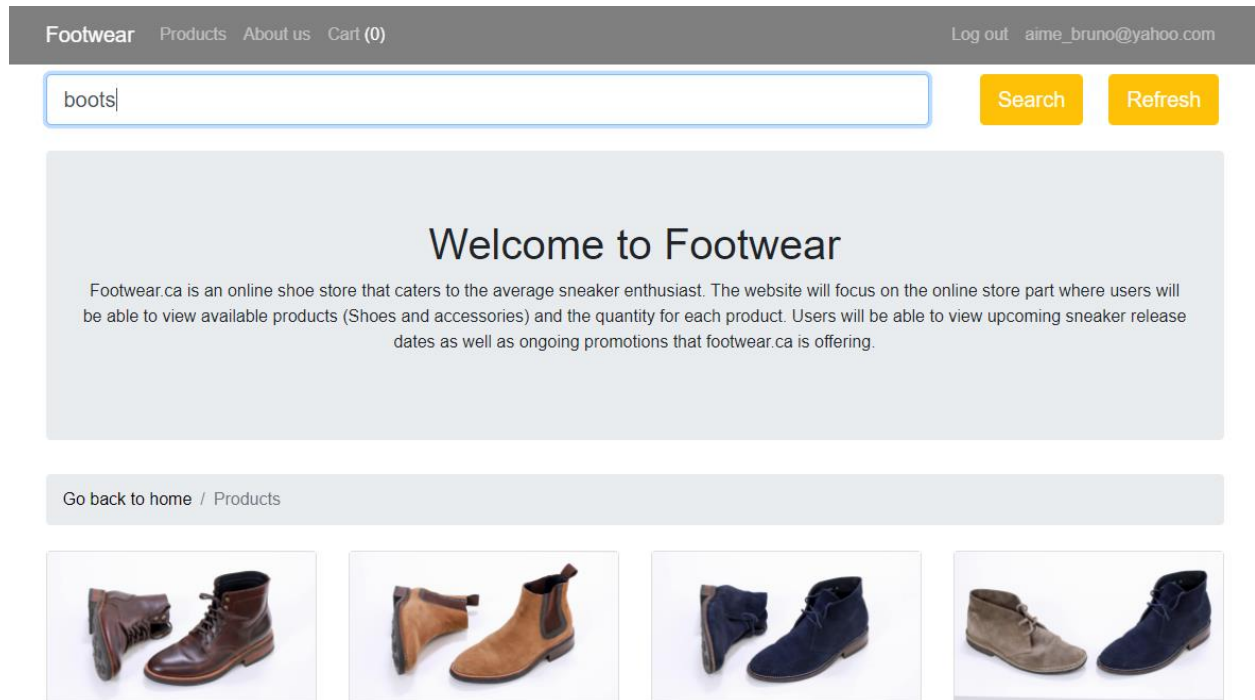
Previous Next Last >>

8. Card: A user can search for products by price

Conversation: The products will be derived from the products database

Conformity: We will be verifying that the user entered a valid integer and then comparing the input to the prices column in the database.

Implementation: User input is placed in a variable "Price" of type int,
`SELECT FROM products WHERE product_price == Price;`

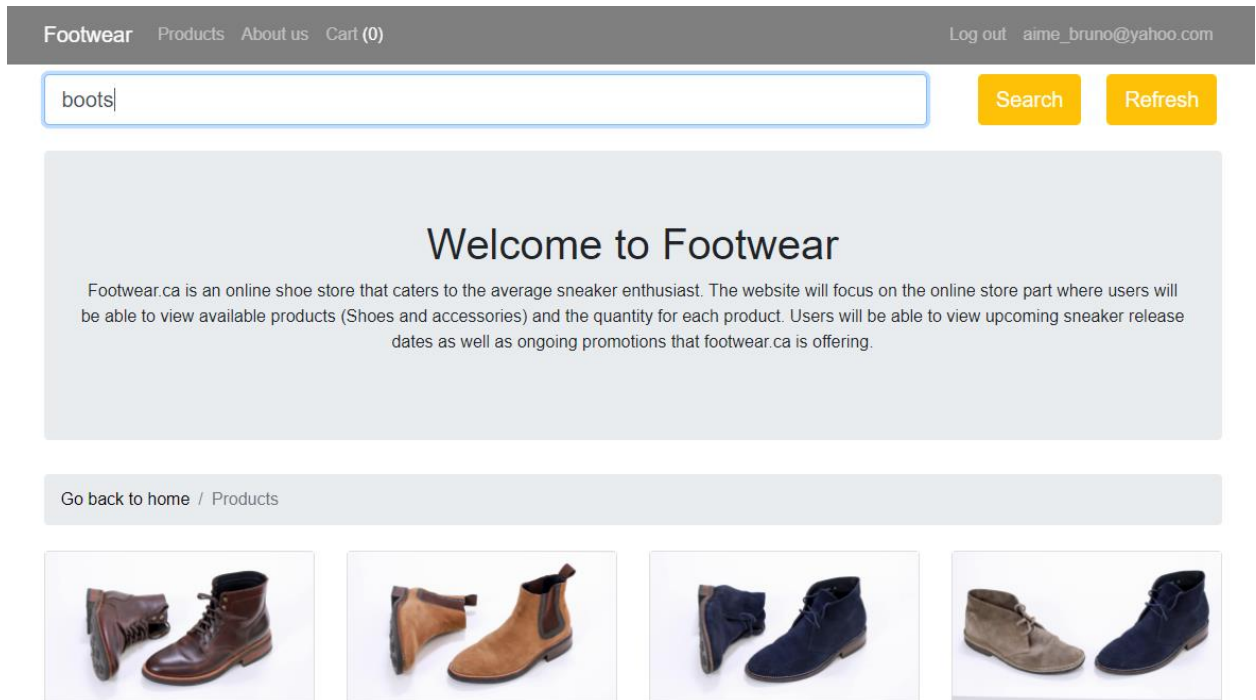


9. **Card: A user can select products by name**

Conversation: The products will be derived from the products database

Conformity: We will be comparing the string input from the user to the names column in the database.

Implementation: User input is placed in a variable \$Name of type string,
`SELECT FROM products WHERE product_name LIKE '%$Name%';`



10. Card: A user can search by description of product

Conversation: The products will be derived from the products database. This will come out handy for those who wants a specific type of product

Conformity: We will be comparing the string input from the user to the product_description column in the database.

Implementation: User input is placed in a variable \$Name of type string,
`SELECT FROM products WHERE product_description LIKE '%$Name%';`

11. Card: A user can select by brand

Conversation: The products will be derived from the products database

Conformity: We will be comparing the string input from the user to the brand column in the database.


Implementation: User input is placed in a variable \$Brand of type string,
`SELECT FROM products WHERE product_brand LIKE '%$Brand%';`

12. Card: A user can add selected products to the cart

Conversation: There will be a checkbox for each product in the table/list and the user will have to check the box to select the product.

Conformity: The name, description, price, and image of the selected product will be added to the order database.

Implementation: The name, description, price, and image of the product will be added into Variables, \$Name, \$Description, \$Price, and \$img respectively,
`INSERT INTO order VALUES(Name, Description, Price, img);`



Brown Boots
\$30


Add To Cart

Edit

[About us](#) [Cart \(1\)](#)

[Log out](#)

This is the page for cart information!



Brown Boots
\$30

Delete

Shipping address:

500 santa-claus ave

Total: \$30 + HST

Checkout

13. Card: A user can see the total in the cart as they add products

Conversation: The total will be obtained from the order database

Conformity: We will be adding prices of all products added to the cart by the user and obtaining the total.

Implementation: Get the list of orders associated with this user, then go through all orders while adding their prices and placing them in Variable \$Total,

```
Var temp = SELECT * FROM ORDER WHERE customerID == userID;
```

```
Int total = 0;
```

```
Foreach (var i in temp){
```

```
    Total += Convert.ToInt32(i.Price); }
```

About us Cart (3)

Log out

This is the page for cart information!



Brown Boots

\$30

Delete



Vans Strike

\$45

Delete



Air Nike Black

\$50

Delete

Shipping address:

500 santa-claus ave

Total: \$125 + HST

Checkout

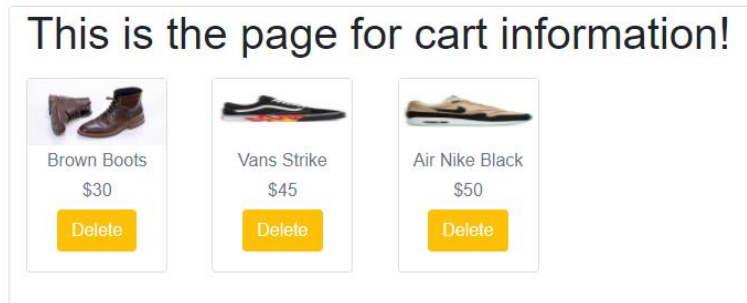
14. Card: A user can open the cart to view all products added

Conversation: The list will be obtained from the order database

Conformity: we will be going through the database and retrieving the orders the user just added

Implementation: we will obtain this user's ID and will place it in a variable called userID


```
SELECT * FROM order WHERE customerID == userID.
```



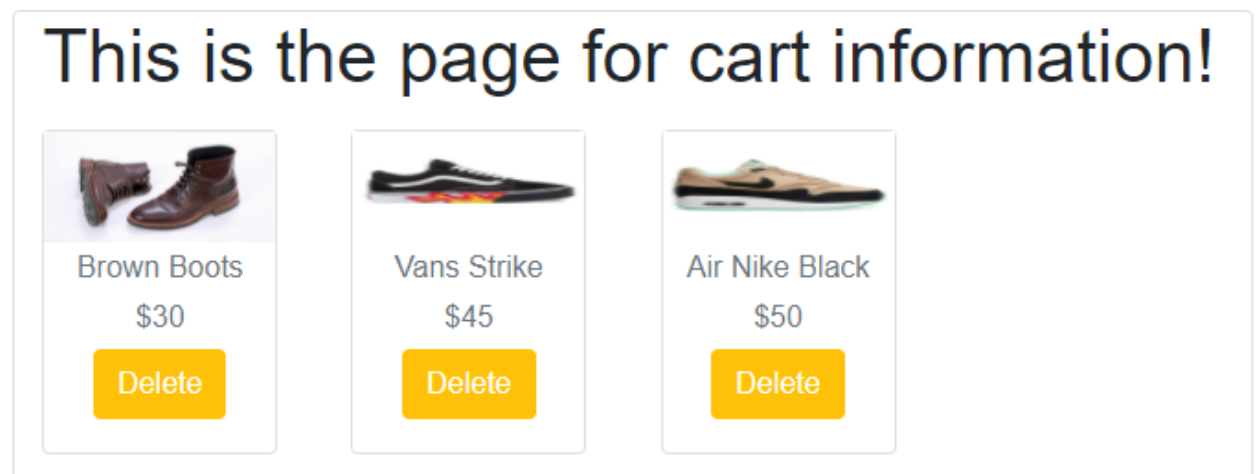
15. Card: A user can remove any product added to the cart

Conversation: The removed product will be removed from the cart table

Conformity: We will grab the productID of the product that the user just removed, then remove the product from the cart table that matches this ID. We will then refresh the cart's page with the updated info.

Implementation: we will place the removed productID in a variable called id.

```
DELETE FROM cart WHERE productID = id;
```



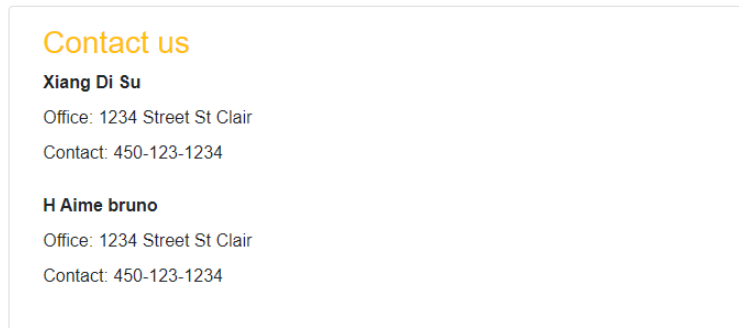
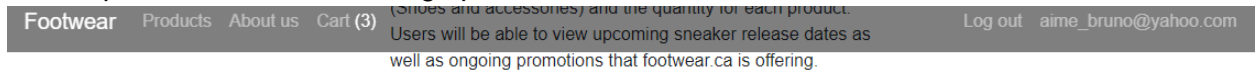
16. Card: A user can view contact us page

Conversation: The contact information will be derived from the contact table

Conformity: When the user clicks on the contact us page view, the page will load with information about the company. Such information will include: phone numbers, email, address, etc

Implementation: In the page load of the contact us page we will have labels for each info we want to display. We will then hold contact from the db in a "contactPage" variable and get each column value and insert into the label.

```
Var contactPage = from x in db.contact select x
phoneLabel = contactPage.phone, etc..
```



17. Card: A user can View promotions page

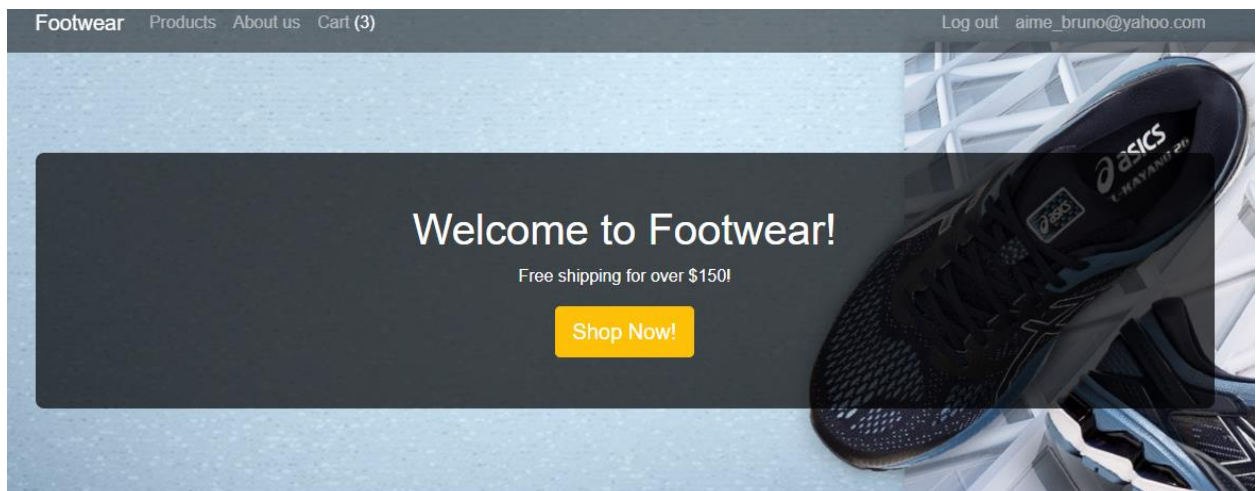
Conversation: The promotions will be displayed from the promotion table in the database

Conformity: We will display the data by 1) Fetching the product's name, promo description, image path, start date, and end date 2) Display it in a table or list

Implementation: Each table cell will have labels for name, promo description, start date, end date, and image element to hold the image. We will get a list of all promotions as a list then loop through the list and assign each label to the corresponding product info.

```
Var promos = (FROM x in db.promotion
              SELECT x).ToList();
```

```
foreach(item in promos)
    nameLabel.text = item.name, etc..
```



18. Card: A user can mark products as favorites

Conversation: The favorite products will be added to the favorite table

Conformity: Once the product is marked as favorite through the use of a checkbox or a button, we will grab all information about the product (name, description, image, price) and store them

Implementation: variables favName, favDesc, favImg, favPrice will hold the product's info. userID will hold the current user's customer id.

```
INSERT INTO favorite VALUES (userId, favName, favDesc, favImg, favPrice);
```

19. Card: A user can view their favorite products

Conversation: favorite products will be derived from the favorite table

Conformity: We will grab the current user's id and compare to the customer ids in the favorite table, if there is a match, we will fetch all rows that match this id and display them on the favorite page

Implementation: variable currUser will hold the user's id, var favList will hold a list of products with customerID that matches currUser. We will then bind the list to the gridview on the favorite page.

```
Gridview.Source = (from x in db.ITEMs select x).ToList<ITEM>();  
Gridview.DataBind();
```

20. Card: A user can unmark products as favorites

Conversation: The unmarked product will be deleted from the favorite table

Conformity: Once the product is unmarked as favorite through the use of a checkbox or a button, we will grab all information about the product (name, description, image, price) and remove them.

Implementation: variable favName will hold the product's info. userID will hold the current user's customer id.

```
DELETE FROM favorite WHERE name == favName AND userID == customerID;
```

21. Card: A user can sign up for the company's newsletter

Conversation: The user's First name, Email address, and sign up date will be stored in the subscription table.

Conformity: When the user submits their subscription to the newsletter, we will first confirm that the email is valid using regex, then store it into the database. If they are already logged in, we will automatically grab their name and email

Implementation: Have variables to hold the user's name, email, and current timestamp.

```
INSERT INTO subscription VALUES(name, email, current timestamp);
```

Term Project: Use Cases Entity Relationship Diagram

