CPSC 471 Project Report

Group 72 – Stevan Beljic (30061812), Max Kepler (30093950)

Bros in a Bun

Abstract

Inventory management systems are very important to fast food restaurants, especially smaller businesses. Seeing that an overwhelming 87% of businesses are overstocking their inventory by more than 20% (Lightspeed, 2021), which is even more costly in the restaurant industry in which overstocked goods lead to waste and therefore monetary loss, adequate inventory management is vital to the profitable operation of local restaurants. The solution to this problem is not quite so simple for many businesses as offerings from companies such as LightSpeed or TripleSeat offer overly complicated applications that are difficult to use and inflate prices by including services that are often not needed. Our solution is a website with two primary users: managers and customers. Managers can directly view and order stock, while customers can view available items and place orders ahead of time.

In this report, we will justify the economic need for our proposed solution by defining the magnitude of the problem, why it is important for it to be resolved, and how our solution differs from that of competitors. We will explain our proposed solution through entity-relation and relational-model diagrams, and the reasoning behind certain design decision and any modifications from previous iterations of diagrams. Furthermore, we will explain the different end-users of the application and demonstrate how they would each navigate the application, as well as how the database was implemented in the backend.

Contents

Problem Definition

Inventory management systems are vital to the operation of businesses, big or small. Inventory management systems ensure the smooth operation of businesses through maintaining adequate levels of stock for sale, while avoiding over-purchasing stock so that it does not sit idle or go to waste. This is especially prevalent in the restaurant industry, where stock consists of food items which are either quickly expended or quickly expired. Small, local businesses can not afford to incur these losses on inventory on their already narrow profit margins, and most inventory management systems, such as LightSpeed, offer simple services for a much higher and unaffordable price. These inventory systems also merely allow the business to monitor stock and flag items which are in low stock, but do not offer the ability to directly order said items through the application. Furthermore, similar inventory management systems such as TripleSeat add on additional features such as point of sales (POS) systems which are not needed by the business and inflate the price of the service offering. Seeing the difficulties faced by small business throughout the pandemic, where 56.6% of food service faced profitability issues as compared to 41.8% of other businesses (Government of Canada, Statistics Canada, 2021), a more streamlined and affordable inventory management system is necessary for small and medium sized restaurants to run more lean inventories and adequately manage stock.

Proposed Solution

Seeing the difficulties faced by small restaurants in managing inventory, particularly in turbulent economic times such as the COVID-19 pandemic, we sought to provide a simpler inventory management system which can be applied to a wide variety of fast-food restaurants, from pizza to burgers to sandwiches: our example will be a fictitious sandwich shop named “Bros in a Bun.” Through our offering, we give restaurants a cheaper and easy to use way to directly view and manage stock by providing direct ordering through the application, in which each ingredients’ supplier information is saved

within the database. What's more, we offer a customer view wherein live item availability can be viewed and even ordered from so that customers know their desired product is available and purchasable.

Project Design

End Users

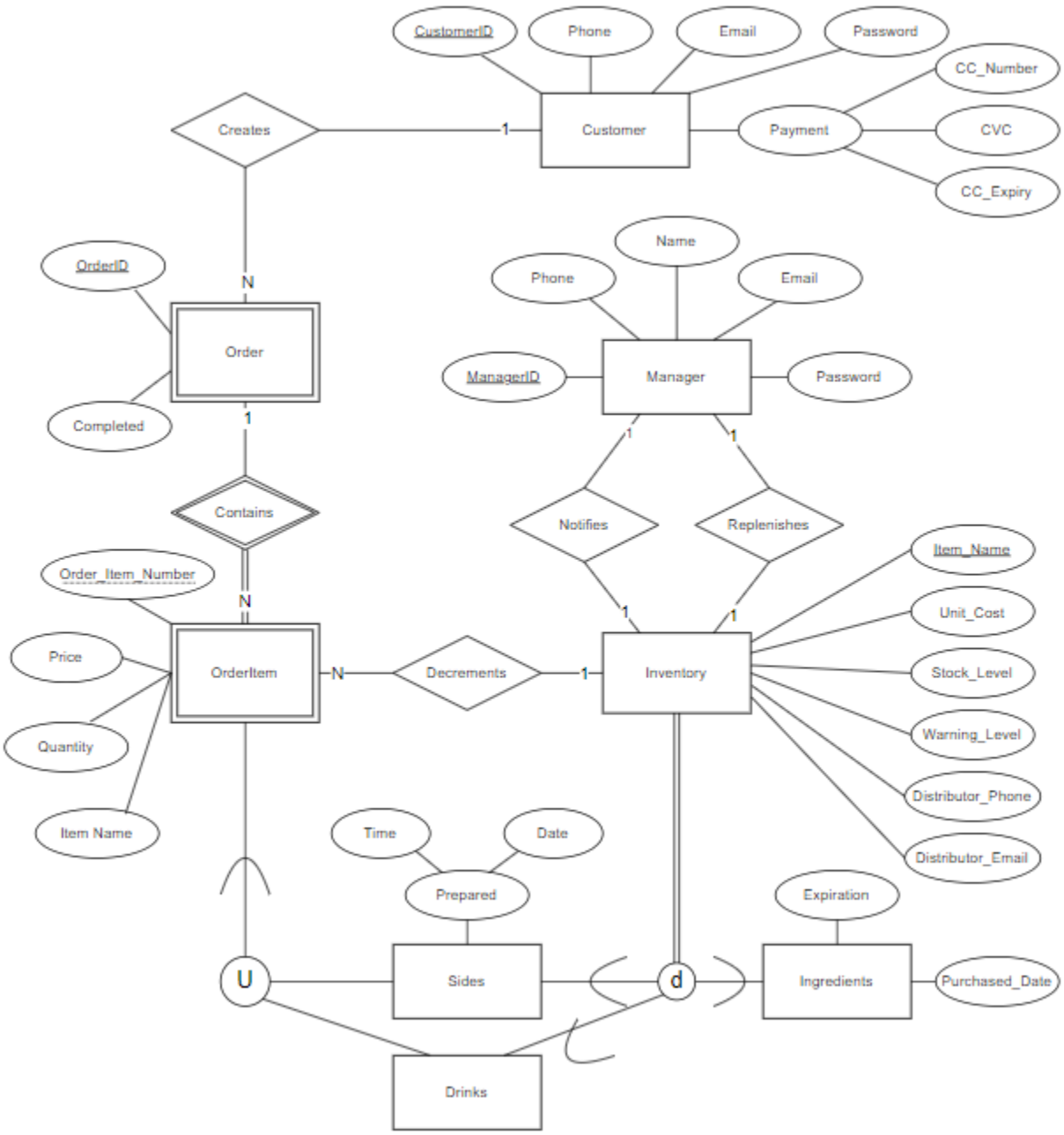
The application has two target end-users: managers who will be viewing and updating inventory levels, and customers who can view available inventory and place orders ahead of time. Both customers and managers sign in through the same login page. A manager must already be in the user database, while a customer can create an account if they do not already have an account saved in the database.

Upon successful login, managers are redirected to a landing page informing them they have logged in as a manager, in which managers are then given the options different from those which are shown to customers. Managers are given the option to view their restaurants inventory information, which display current stock levels for all inventory items in their product offering (which are ingredients, drinks, and sides). Each displayed item will have a corresponding drop-down from which managers can select how many of each individual item they wish to order, and then click a button to process all the orders at once.

Upon a successful customer login, the landing page is different from what is shown to the manager. Customers are met with a landing page from which they can either select to edit personal information or place an order. Creating an order allows customers to create customized sandwiches from which only available ingredients may be selected, and an image is updated which will display the customers' selected item. Should an item not be available for purchase, the customer will not have the ability to select the item. Once the customer places and confirms the order, the restaurant's inventory database is automatically decremented according to the items included in the order.

EERD

The Extended Entity Relationship Diagram (EERD) is shown below.



The EERD is largely the same as the one previously presented, however there have been some modifications which occurred during development, described as follows:

- The “Sandwich Recipe” entity has been removed. This is because we have made it so customers create their sandwiches by themselves, with the ingredients of their choosing, removing the need for recipes. Some pre-set options are available within the form, but these pre-set options are more

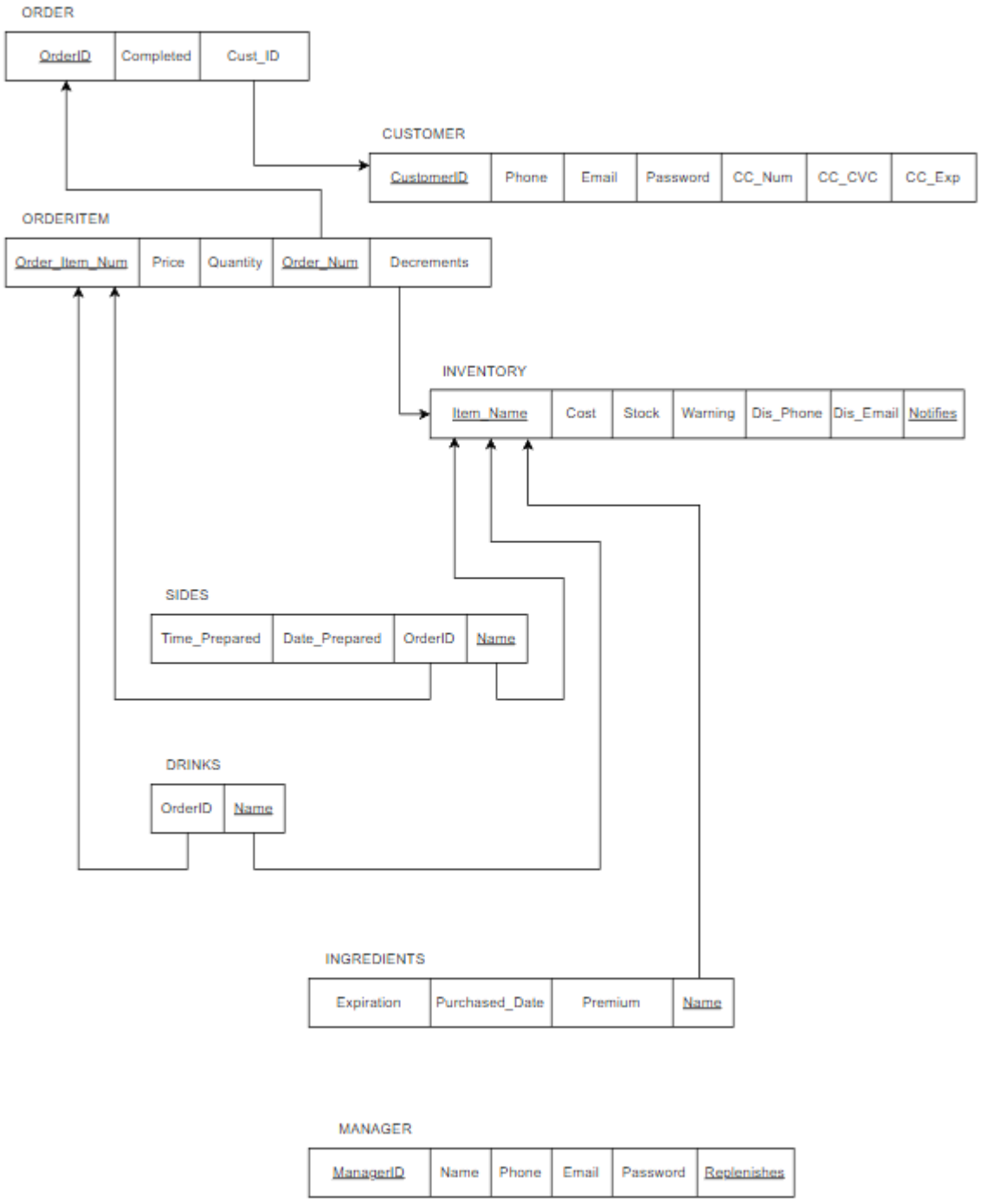
so starter guidelines than they are complete sandwich recipes. Thus, we found it redundant to have an entire table for sandwich recipes when the primary functionality is to create custom sandwiches.

- The “Order” entity is now a weak entity. This is because each order is identified first by the customer, and then by the individual order. A customer must be identified first in order for an order to be identified.
- The “Image Link” attribute was removed from the “Ingredients” entity since these images are hard-coded into the website itself and are not pulled from the database.
- A “Name” attribute has been added to the “Manager” entity.

Implementation

Relational Model

The Relational Model (RM) is shown below.



Akin to the EERD in the previous section, the RM has had minor changes along the development process to more accurately depict the application that was developed. The changes to the RM are as follows:

- The “Sandwich Recipe” and “Recipe Ingredients” relations have been removed as customers now custom select the ingredients in their sandwiches.
- “Cust_ID” has been added as a foreign key to the “Order” relation, referencing the “Customer” relation. This is because orders are identified first by the individual customer, and then by the order placed.
- “Image Link” has been removed from the “Ingredients” relation.
- “Name” has been added to the “Manager” relation.

Database Implementation

Our “Bros in a Bun” application is locally run on our desktops through XAMPP, using MariaDB as the database management system, PHP for the back-end development, and HTML and CSS for the front-end.

Listed below are queries used in the application, as well as their use.

File: *sign-in.php*

Query to retrieve all customers’ emails, passwords, and UserIDs from customer table

```
$sql = "SELECT Email, Password, UserID FROM customer";
```

Query to retrieve all manager’s emails, passwords, and ManagerIDs from manager table

```
$sql = "SELECT Email, Password, ManagerID FROM manager";
```

Query to retrieve customer tuple with matching userID to set session variable

```
$sql = "SELECT * FROM customer WHERE UserID = '$userID'";
```

Query to retrieve manager tuple with matching ManagerID to set session variable

```
$sql = "SELECT * FROM manager WHERE ManagerID = '$ManagerID'";
```

File: *sign-up.php*

Query to retrieve UserID if customer with the email already exists

```
$sql = "SELECT UserID FROM customer WHERE Email= '$Email'";
```

Insertion of new customer tuple into customer table

```
$sql = "INSERT INTO customer (Email, Password, Name, Phone, CC_Num, CC_CVC, CC_Exp) VALUES ('$Email', '$Password', '$Name', '$Phone', '$CC_Num', '$CC_CVC', '$CC_Exp')";
```

File: *edit-c.php*

Update Email in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET Email = '$email' WHERE UserID = '$id'";
```

Update Password in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET Password = '$Password' WHERE UserID = '$id'";
```

Update Name in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET Name = '$Name' WHERE UserID = '$id'";
```

Update Phone in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET Phone = '$Phone' WHERE UserID = '$id'";
```

Update credit card number in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET CC_Num = '$CC_Num' WHERE UserID = '$id'";
```

Update credit card cvc in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET CC_CVC = '$CC_CVC' WHERE UserID = '$id'";
```

Update credit card expiry in tuple with UserID (retrieved from session variable) in customer table

```
$sql = "UPDATE customer SET CC_Exp = '$CC_Exp' WHERE UserID = '$id'";
```

File: *confirm.php*

Query to retrieve ingredient stock level and cost from ingredient table joined with inventory

```
$sql = "SELECT Stock_level, Unit_cost FROM ingredients NATURAL JOIN inventory WHERE Item_name = '$sandTopping'";
```

Query to retrieve side stock level and cost from side table joined with inventory

```
$sql = "SELECT Stock_level, Unit_cost FROM sides NATURAL JOIN inventory WHERE Item_name = '$sideItem'";
```

Query to retrieve drink stock level and cost from drink table joined with inventory

```
$sql = "SELECT Stock_level, Unit_cost FROM drinks NATURAL JOIN inventory WHERE Item_name = '$drinkItem'";
```

Query to retrieve side stock level from sides table joined with inventory if more of item selected than available

```
$sql = "SELECT Stock_level FROM sides NATURAL JOIN inventory WHERE Item_name = '$sideItem'";
```

Query to retrieve drink stock level from drinks table joined with inventory if more of item selected than available

```
$sql = "SELECT Stock_level FROM drinks NATURAL JOIN inventory WHERE Item_name = '$drinkItem'";
```

Query to retrieve ingredient stock level from ingredients table joined with inventory if more of item selected than available

```
$sql = "SELECT Stock_level FROM ingredients NATURAL JOIN inventory WHERE Item_name = '$sandTopping'";
```

Insertion of new order into order table with empty attribute values, apart from UserID

```
$sql = "INSERT INTO `order` (`OrderID`, `Completed`, `CustomerID`) VALUES (NULL, '0', '$id')";
```

Query to retrieve OrderIDs of all uncompleted orders in order table with matching customerID

```
$sql = "SELECT `OrderID` FROM `order` WHERE `CustomerID`='$id' AND `Completed`='0'";
```

Query to retrieve cost of ingredient from ingredient table joined with inventory table

```
$sql = "SELECT Unit_cost FROM ingredients NATURAL JOIN inventory WHERE Item_name = '$sandTopping'";
```

Insertion of an ingredient into orderitem table upon successful placement of order

```
$sql = "INSERT INTO `orderitem` (`Order_Item_Num`, `Item_name`, `Price`, `Quantity`, `Order_Num`) VALUES (NULL, '$sandTopping', '$price', '1', '$orderID')";
```

Query to retrieve cost of side from side table joined with inventory table

```
$sql = "SELECT Unit_cost FROM sides NATURAL JOIN inventory WHERE Item_name = '$sideItem'";
```

Insertion of a side into orderitem table upon successful placement of order

```
$sql = "INSERT INTO `orderitem` (`Order_Item_Num`, `Item_name`, `Price`, `Quantity`, `Order_Num`) VALUES (NULL, '$sideItem', '$price', '$sideQuantity', '$orderID')";
```

Query to retrieve cost of drink from drink table joined with inventory table

```
$sql = "SELECT Unit_cost FROM drinks NATURAL JOIN inventory WHERE Item_name = '$drinkItem'";
```

Insertion of a drink into orderitem table upon successful placement of order

```
$sql = "INSERT INTO `orderitem` (`Order_Item_Num`, `Item_name`, `Price`, `Quantity`, `Order_Num`) VALUES (NULL, '$drinkItem', '$price', '$drinkQuantity', '$orderID')";
```

Query to retrieve item name and quantity purchased from orderitem table where orderID is that of the previously placed order

```
$sql = "SELECT Item_name, Quantity FROM `orderitem` WHERE Order_Num = '$orderID'";
```

Query to retrieve a corresponding item's stock level from inventory table

```
$sql = "SELECT Stock_level FROM inventory WHERE Item_name = '$item_name'";
```

Update the current item's stock level to the new amount following decrement upon successful order

```
$sql = "UPDATE inventory SET Stock_level = '$new_stock' WHERE Item_name = '$item_name'";
```

Update current order status to completed in order table

```
$sql = "UPDATE `order` SET `Completed`='1' WHERE `OrderID`='$orderID'";
```

File: *receipt.php*

Query to retrieve OrderID of all completed orders of current customer from order table

```
$sql = "SELECT `OrderID` FROM `order` WHERE `CustomerID`='$id' AND `Completed`='1'";
```

Query to retrieve name of ingredient and quantity purchased from most recent order from orderitem table joined with ingredients

```
$sql = "SELECT Item_name, Quantity FROM `orderitem` NATURAL JOIN ingredients WHERE Order_Num = '$orderID'";
```

Query to retrieve name of side and quantity purchased from most recent order from orderitem table joined with sides

```
$sql = "SELECT Item_name, Quantity FROM `orderitem` NATURAL JOIN sides WHERE Order_Num = '$orderID'";
```

Query to retrieve name of drink and quantity purchased from most recent order from orderitem table joined with drinks

```
$sql = "SELECT Item_name, Quantity FROM `orderitem` NATURAL JOIN drinks WHERE Order_Num = '$orderID'";
```

File: *View-inventory.php*

Query to retrieve all tuples in ingredients table joined with inventory

```
$sql  = 'SELECT * FROM  ingredients NATURAL JOIN inventory';
```

Query to retrieve all tuples in sides table joined with inventory

```
$sql  = 'SELECT * FROM  sides NATURAL JOIN inventory';
```

Query to retieve all tuples in drinks table joined with inventory

```
$sql  = 'SELECT * FROM  drinks NATURAL JOIN inventory';
```

Update the selected items stock level to the new level based on the quantity of the item ordered

```
$sql = "UPDATE inventory SET Stock_level = '$newLevel' WHERE Item_name = '$iName'";
```

Visual Interface Design

The visual interface of “Bros in a Bun” was designed to be sleek, easy to use, and require minimal keyboard input from the user. Unless explicitly required, everything on the website can be completed using solely mouse clicks. The login screen requires textual input for the email and password, as does the registration screen, since users need to provide textual data to the website. Similarly on the edit customer information screen, users will also need to enter textual information if they wish to update the textual data associated with their account. On all other pages of the website, all tasks can be accomplished using solely the mouse, in order to make the website as user-friendly and simple as possible. On the customer’s

create order page, we apply switches to select sandwich ingredients, as well as number fields for sides and drinks so that users have the choice of either textually entering quantities to order or employing the arrows to modify their selections. Similarly, managers are also given the number fields on the view inventory page where they can also use the arrows to avoid using keyboard input.

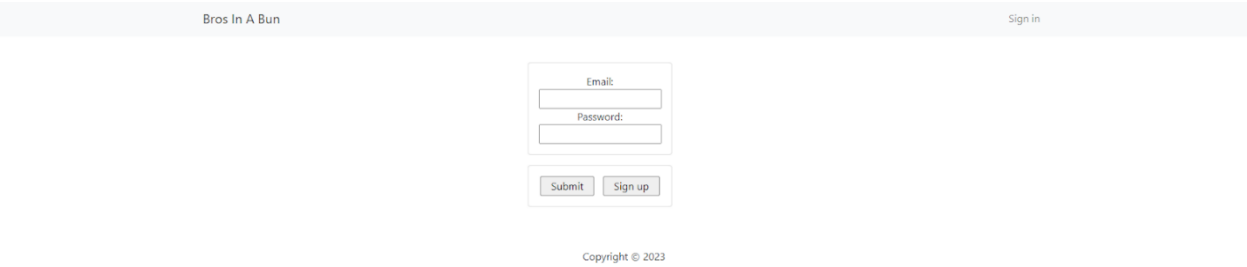
In the design of the website itself, we aimed to make everything look sleek and simplistic. We wanted to avoid having too many visual distractions on the screen so that users' eyes were drawn to the important information on the screen, such as the buttons or data being displayed. We used light gray backgrounds with dark text to maximize contrast so that information being displayed stood out to users.

User Guide

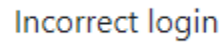
1. General

This is the user guide for general operation. These steps will be the same regardless of whether a user is a manager, customer, or unregistered.

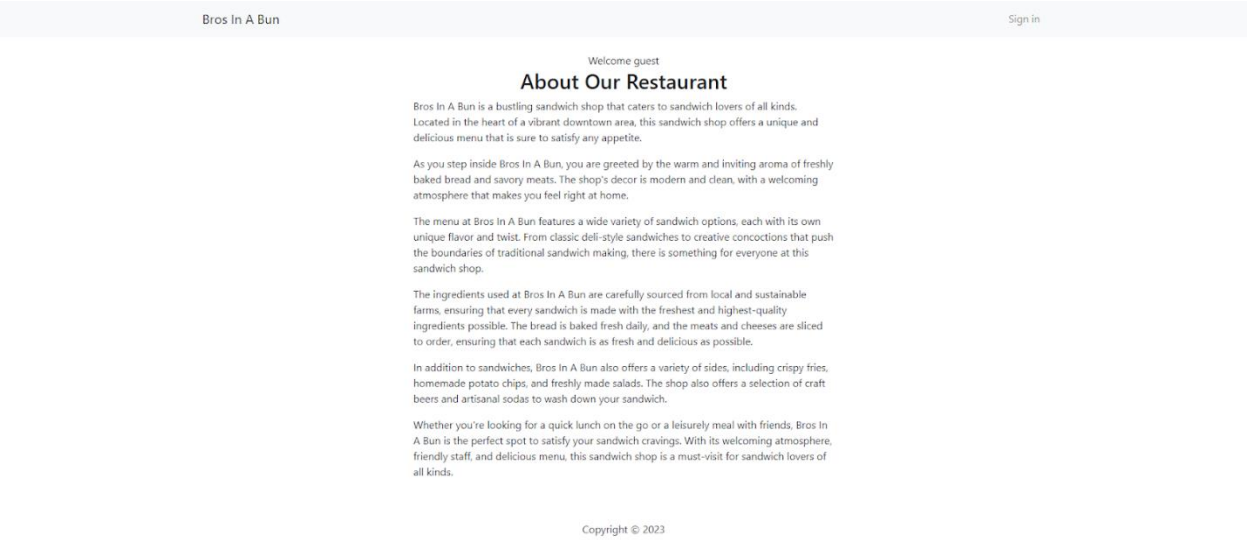
i) The landing screen a user will be shown is the following sign-in page. If the user is a manager or customer, they will enter their credentials. Otherwise, they may click Sign up to register as a customer.



ii) Upon an unsuccessful login, the following message will be displayed above the login section. Users must double check to ensure they are providing the correct credentials.



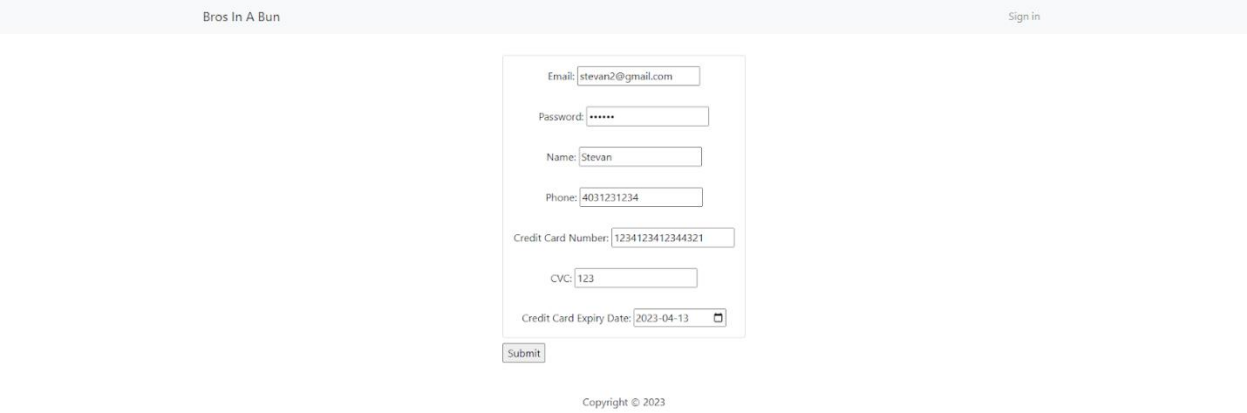
Upon signing out (as either a manager or customer), the user will be shown the default home page and their name atop the homepage will be presented as “guest.” From here, the only option the user has is to click Sign In.



2. Customer

i) New Customer

Upon clicking the Sign-Up button, customers will be presented with this page. Here, the customers must fill in their personal information, of which Email must be unique and not already used by another account.

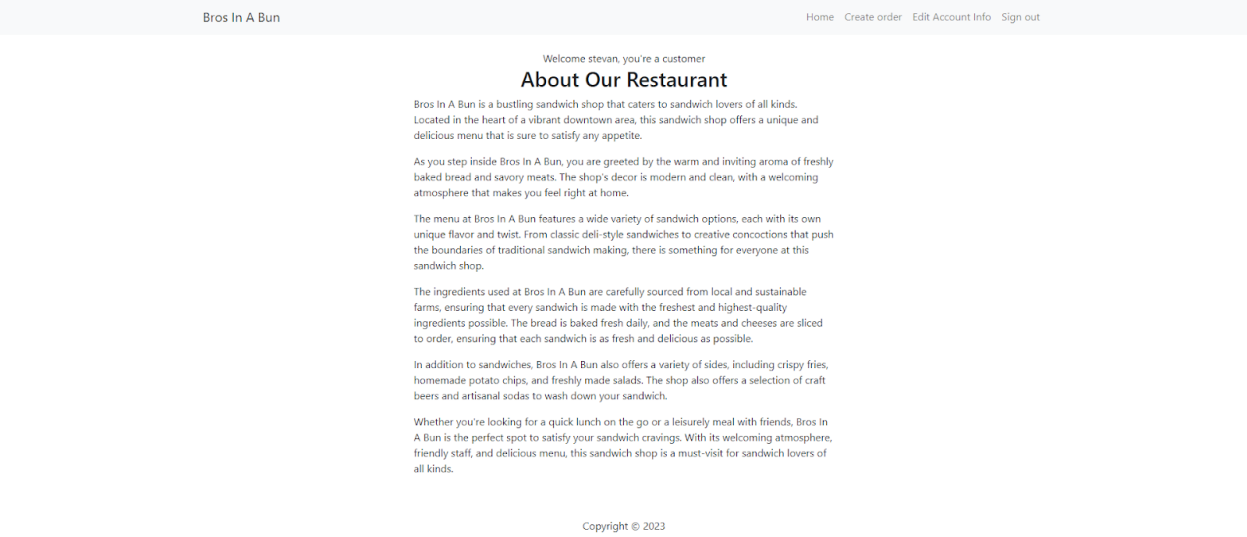


Upon clicking submit, the customer will be logged in and redirected to the website home page (Skip to section Customer.ii). Otherwise, they will be presented with the following error.

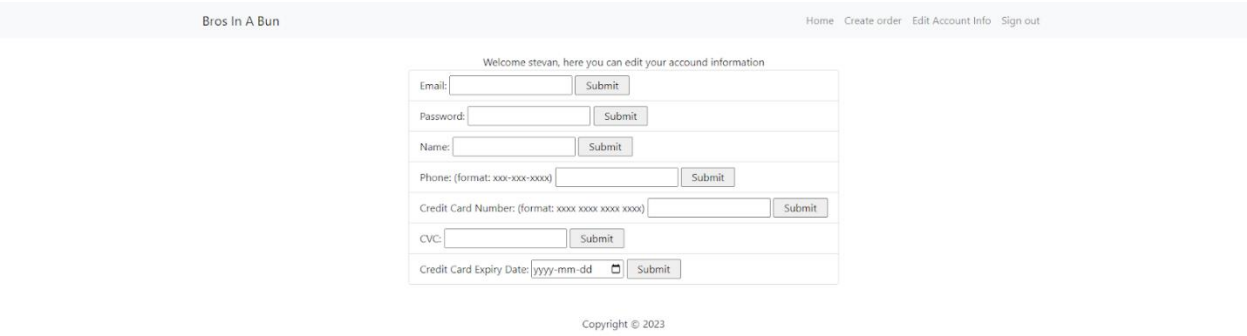
Please try a different email, there is already an account associated with that email.

ii) Existing Customer

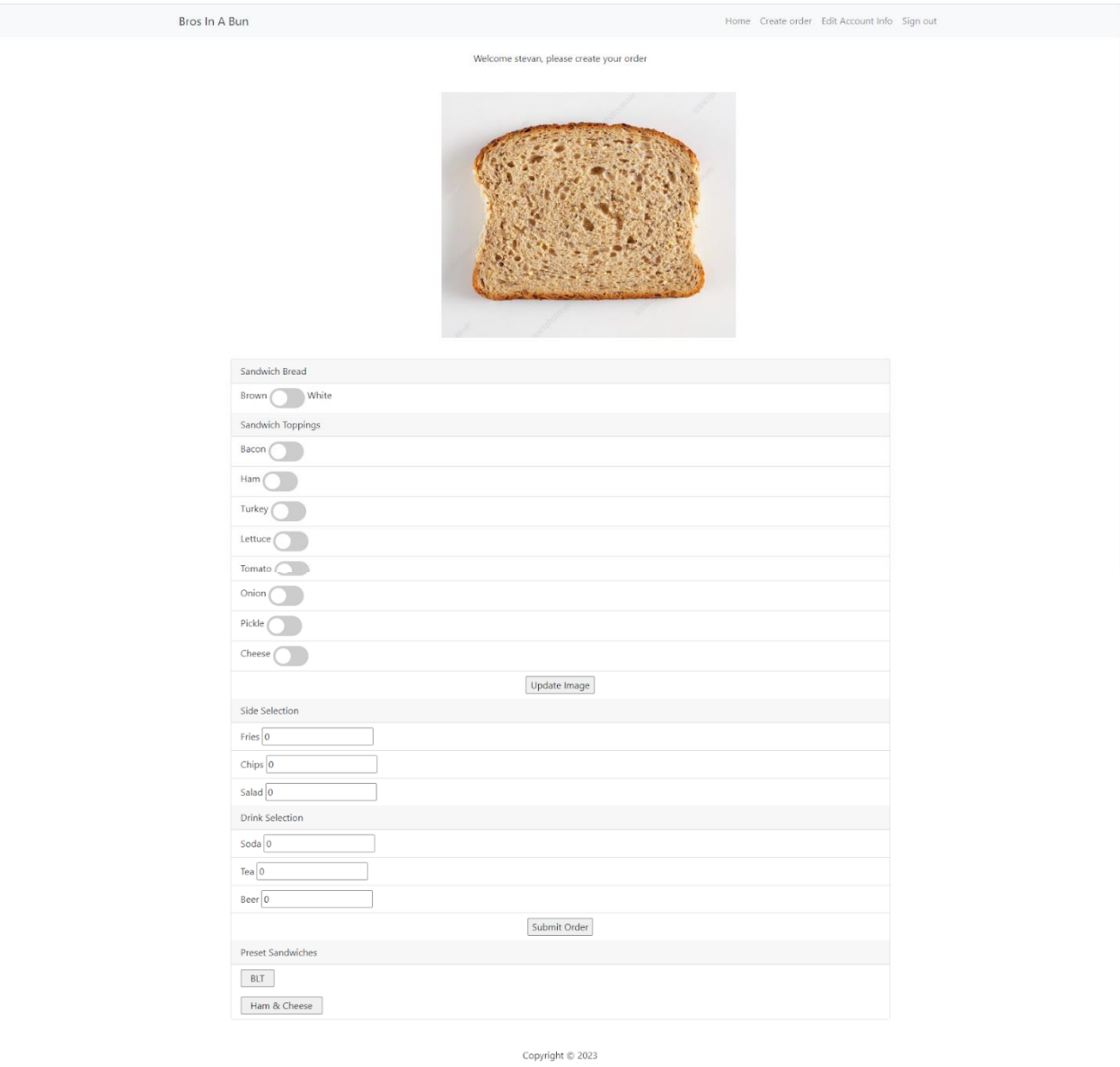
Customers are now on the homepage. From here, they are shown a short description of the restaurant, and are presented with the buttons of: Home, Create order, Edit Account Info, and Sign out. Clicking on Home or the “Bros in a Bun” text in the top left simply return the user to this same screen.



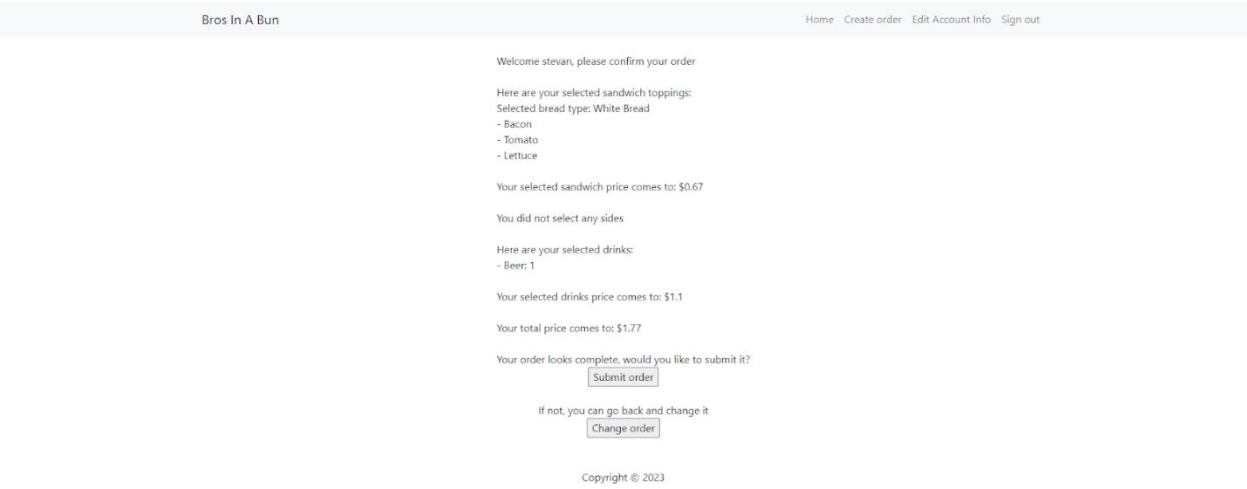
Upon clicking Edit Account Info, customers will be shown the following screen from which they can edit the details they entered when signing up. Submit must be clicked on each individual line for changes to take effect for that line. (For example, entering a new name then clicking Submit on the CVC line will not update the name).



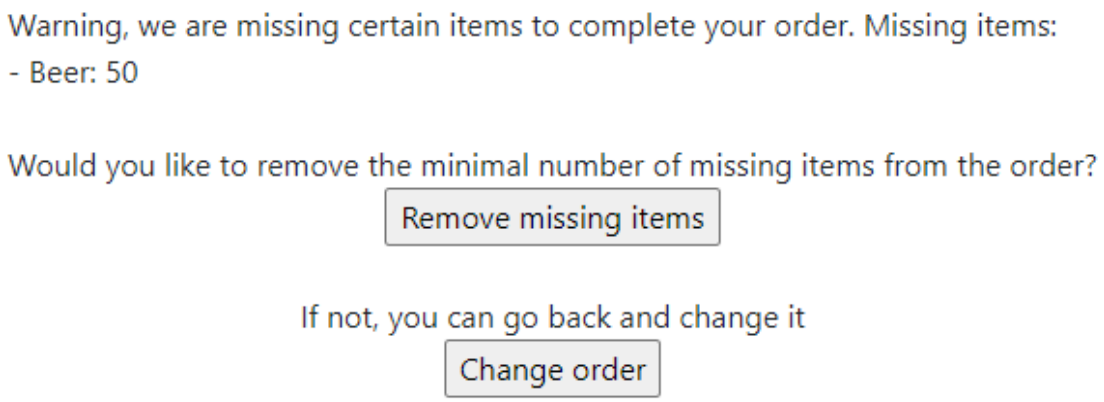
In the Create order page, customers are presented with the following screen. From here, they can select which ingredients they would like in their sandwich, and then click “Update Image” to refresh the image to get an idea of what their sandwich will look like. Pressing one of the pre-set sandwich options will automatically update the ingredients and image, at which point the customer is free to further customize. They can also select the quantity of specific sides and drinks they would like to purchase. Once the customer is content with their order, they can select Submit Order to continue with the purchase.



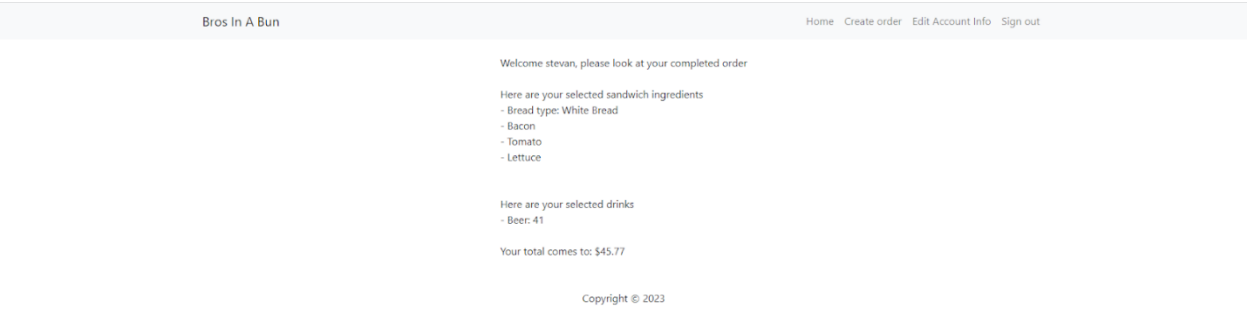
Upon selecting Submit Order, customer will be met the following confirmation page to confirm the order they have created. They will be shown the price for the order, as well as the orders individual components.



In the event of an item not being available in the quantity the customer desired, they will be shown the following warning telling them which item is not available, and they will be prompted to go back and modify their order.



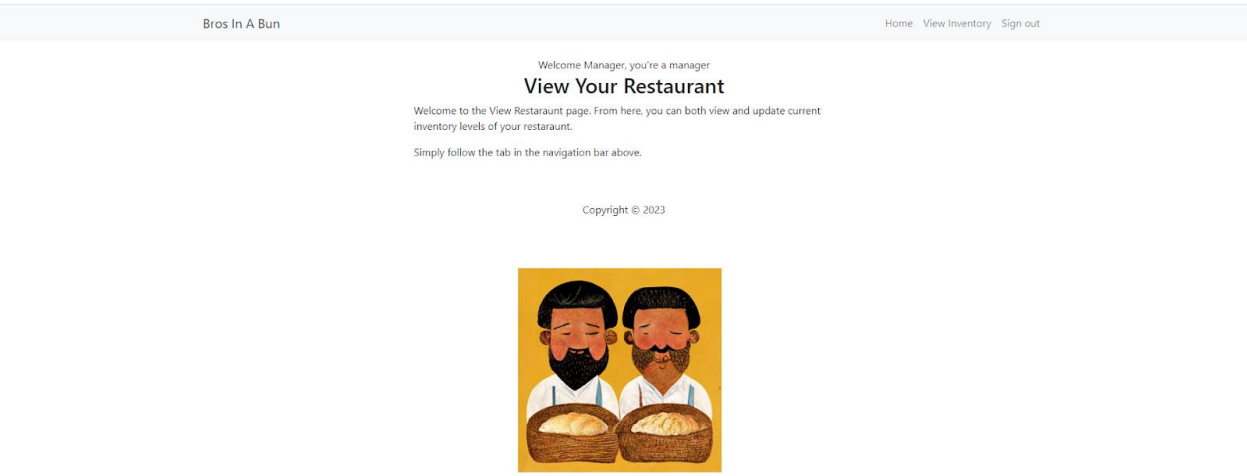
Upon successful completion of their order, the customer will be shown the following receipt screen.



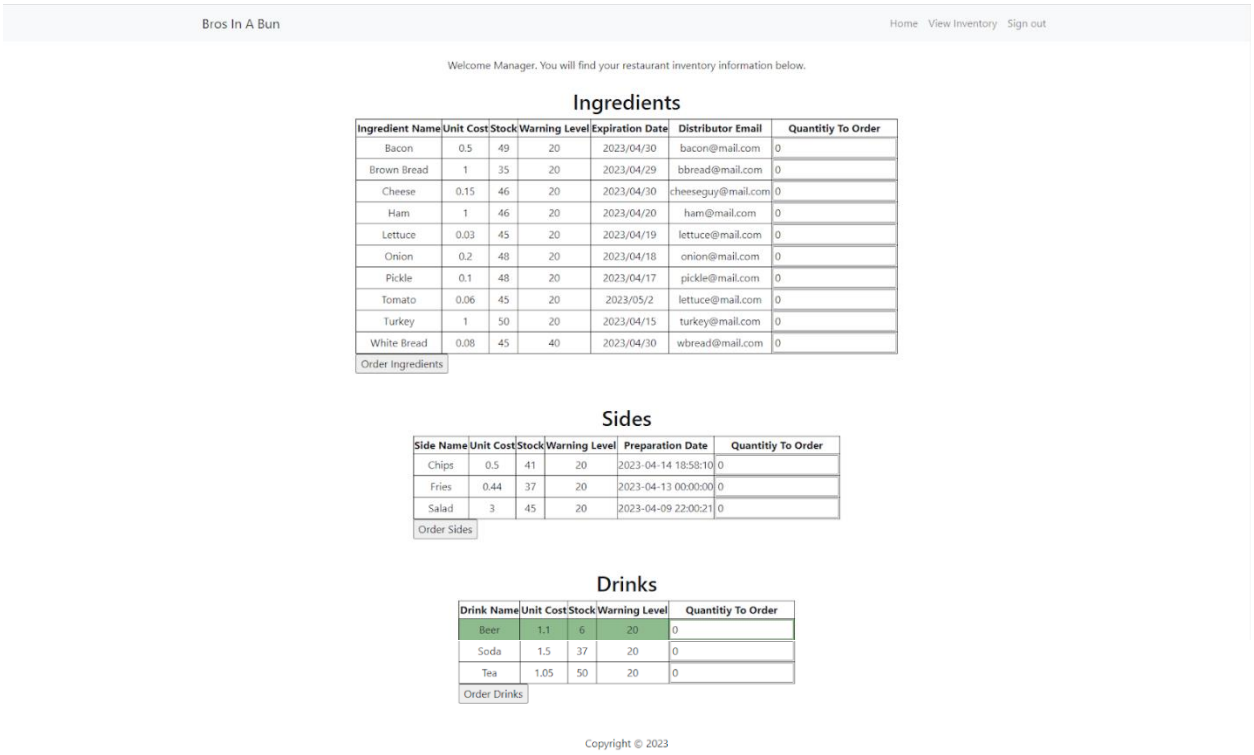
This concludes the customer section of the user manual.

3. Manager

Managers will be shown the following home page, which is different from that of the customers. A manager account can not be created from the website and must manually be inserted into the database for security reasons. From this home page, managers can, similar to customers, press on the Home or “Bros in a Bun” texts to return to this homepage. They can also click View Inventory to manage their restaurants inventory or Sign Out.



Upon clicking View Inventory, managers are shown a summary of the entire inventory of their restaurant, subdivided into smaller tables of ingredients, sides, and drinks. Each inventory item displays its stock level and warning level. The stock level represents the stock of the item, while the warning level represents a guideline as to when the manager should consider ordering more of that item. If the stock level falls below the warning level, the background of the row of that item is turned green to draw attention to it. Each row has a number field from which the quantity to order can be entered. Once a manager is finished overlooking inventory and is ready to order more, they can click the corresponding Order button for each table to place the orders. Upon orderings, the page will simply refresh, and the new stock level is shown.



This concludes the manager section of the user manual.

Sources

Government of Canada, Statistics Canada. (2021, March 18). *Impact of COVID-19 on food services and drinking places, first quarter of 2021*. https://www150.statcan.gc.ca/n1/pub/45-28-0001/2021001/article/00010-eng.htm

Lightspeed 2021-06-30. (2022, January 26). *The new restaurateur's Guide to Restaurant Supply Chain Management*. Lightspeed. Retrieved January 29, 2023, from https://www.lightspeedhq.com/blog/restaurant-supply-chain-management/

Traversy Media. (2022, March 30). *PHP For Beginners | 3+ Hour Crash Course* [Video]. YouTube. https://www.youtube.com/watch?v=BUCiSSyIGGU