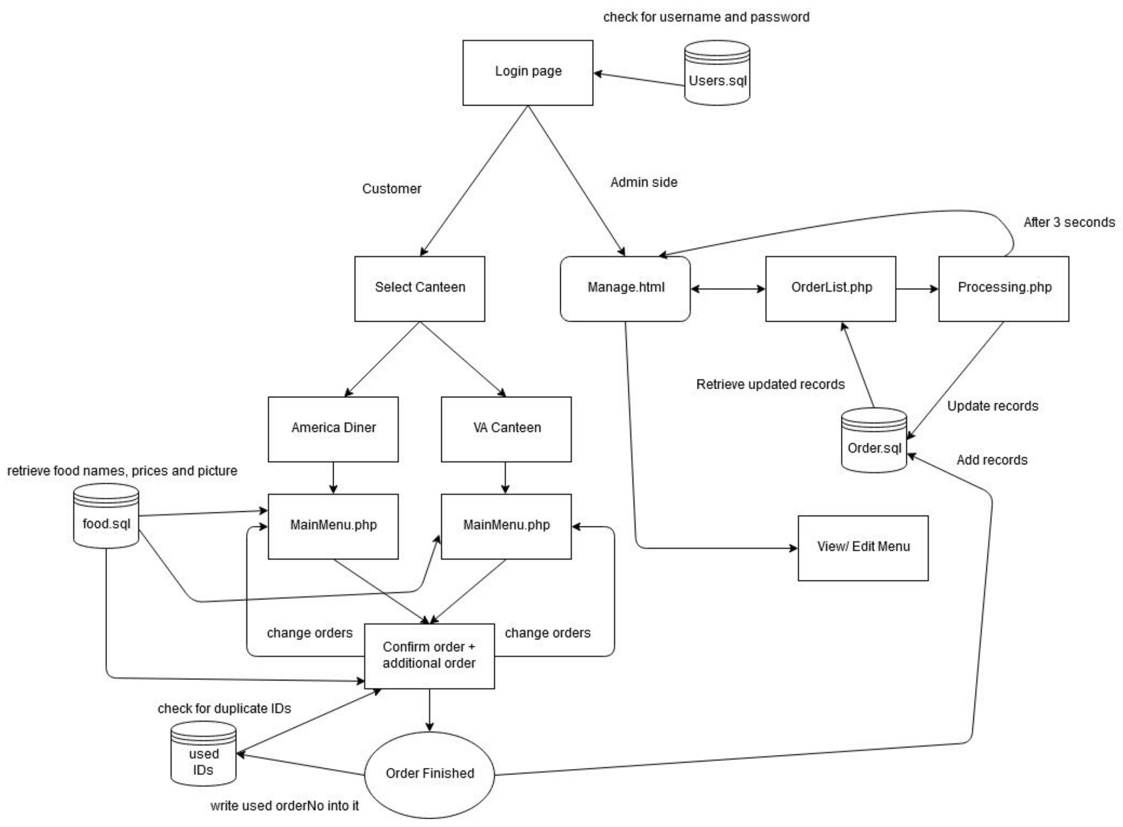
1. Introduction

Imagine that you're waiting in a long line in the canteen during the noon, just to wait for your turn to get to the counter and order your lunch as usual. But today you're sort of in a rush since you've got less than an hour for lunch before your afternoon lecture. Looking at the slow-moving line, you became more and more nervous while time runs out.
Introducing PolyU-Eat, a pre-order platform for our students and staff that speeds up the ordering process and leave you more time to enjoy your meal. Inspired by existing food-ordering platforms like Uber Eats or Deliveroo, we aim to create a system that allows users in our university to pre-order food from the canteens in our campus via any working web browser. Within just a few clicks, one can complete order without waiting in a long line. For the staff in the canteens, they can better manage time and workflow based on the users' ordering time and specified pick up time.
We hope that through this system, the flow of ordering food in our campus can be more efficient, and users can save more time to enjoy his or her meal.

2. Project Structure

Site Map (Rough Sketch, Please Edit in the drawio file by visiting draw.io)



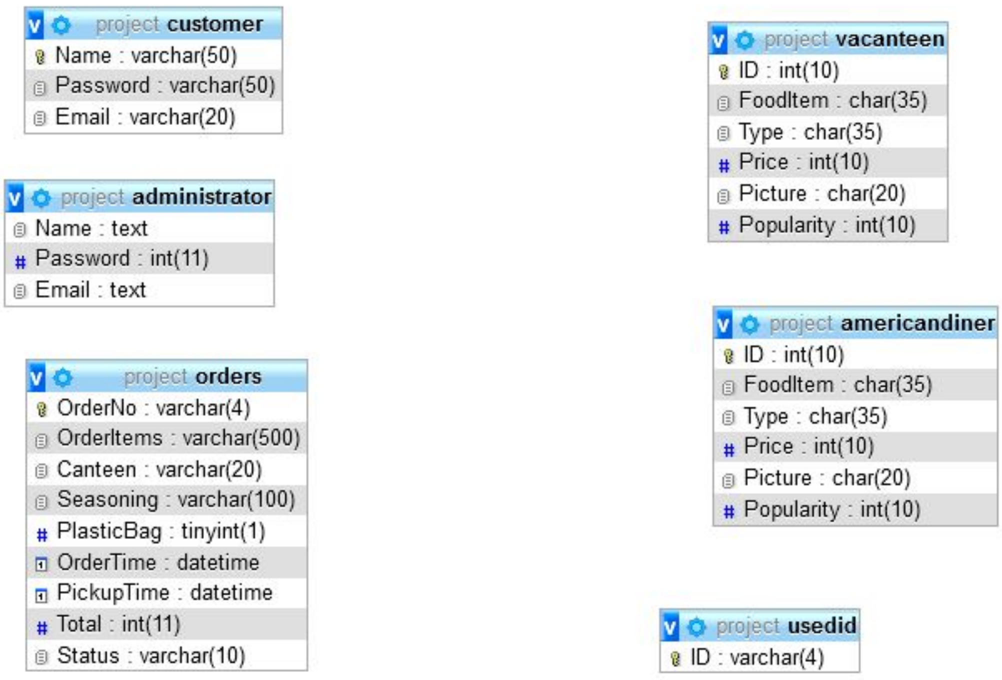
<u>Description</u>

The PolyU-Eat can be used for both customers and administrators. In this case, after the user is logged in, data will be retrieved from Users.sql to identify whether the user is a customer or an administrator. Two different pages will be displayed correspondingly.

For customers, they can first select which canteen they want to view. We currently have two canteens: American Diner and VA Canteen. After the canteen is selected, customers will be directed to the menu page, where they can view all the food in the selected canteen and add the food they want into the shopping cart. The information of the food is retrieved from food.sql. Customers can click to view their shopping cart at any time. This will lead them to the order confirmation page. Customers can go back to shopping or confirm their orders. Some additional order information will be asked to put in during the confirmation.
After the order is confirmed, customers will get a receipt with the food they ordered, the information of the order and the order status. Each order will have a specific order ID. After generating a new ID, the system will check whether it is used before in the usedID.sql. The order information will be put into the Order.sql.

For administrators, their job information which indicates which canteen they belonged to are stored in the database. They will see the management page once they logged in. Administrators can view the orders they have and the menu of their food on this page. The data are retrieved from Orders.sql and food.sql.
Administrators can manipulate the menu by clicking the button under the menu. Then, they can choose to change the price of food items, add food items or delete food items.
Administrators can manipulate the orders by clicking the button under orders. Then, they can view the detailed information of an order. There are four statuses in an order, and it can be updated by administrators. The information will be updated in Order.sql. The page will be automatically redirected to the management page after the operation is finished.

ER Diagrams



project **customer**
- Name : varchar(50)
- Password : varchar(50)
- Email : varchar(20)

project **administrator**
- Name : text
- Password : int(11)
- Email : text

project **orders**
- OrderNo : varchar(4)
- OrderItems : varchar(500)
- Canteen : varchar(20)
- Seasoning : varchar(100)
- PlasticBag : tinyint(1)
- OrderTime : datetime
- PickupTime : datetime
- Total : int(11)
- Status : varchar(10)

project **vacanteen**
- ID : int(10)
- FoodItem : char(35)
- Type : char(35)
- Price : int(10)
- Picture : char(20)
- Popularity : int(10)

project **americandiner**
- ID : int(10)
- FoodItem : char(35)
- Type : char(35)
- Price : int(10)
- Picture : char(20)
- Popularity : int(10)

project **usedid**
- ID : varchar(4)

3. Functionalities

Project.sql

Here is our project database. There are a total of six tables. Administrator and customer are for the user or admin to log in. AmericanDiner and VACanteen are for the menu list. Orders and usedid are for the orderlist and order status.

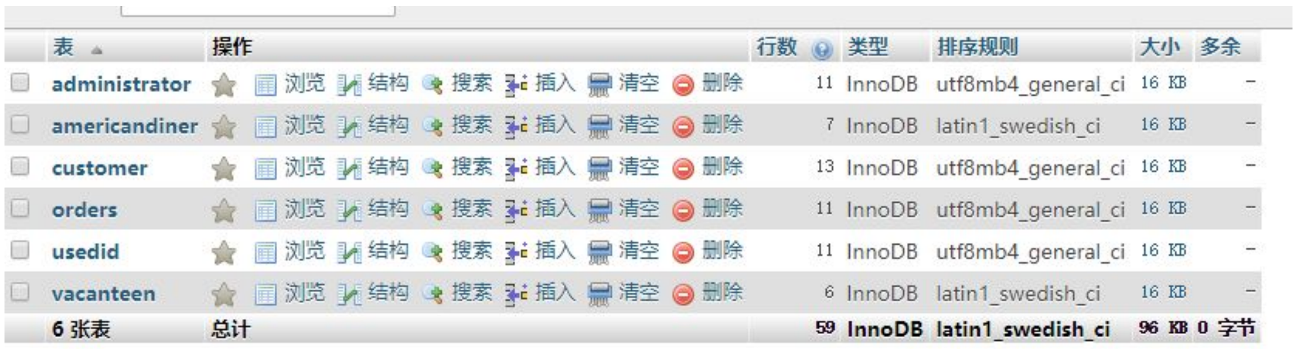


表	操作	行数	类型	排序规则	大小	多余
administrator	浏览 结构 搜索 插入 清空 删除	11	InnoDB	utf8mb4_general_ci	16 KB	-
americandiner	浏览 结构 搜索 插入 清空 删除	7	InnoDB	latin1_swedish_ci	16 KB	-
customer	浏览 结构 搜索 插入 清空 删除	13	InnoDB	utf8mb4_general_ci	16 KB	-
orders	浏览 结构 搜索 插入 清空 删除	11	InnoDB	utf8mb4_general_ci	16 KB	-
usedid	浏览 结构 搜索 插入 清空 删除	11	InnoDB	utf8mb4_general_ci	16 KB	-
vacanteen	浏览 结构 搜索 插入 清空 删除	6	InnoDB	latin1_swedish_ci	16 KB	-
6 张表	**总计**	**59**	**InnoDB**	**latin1_swedish_ci**	**96 KB**	**0 字节**

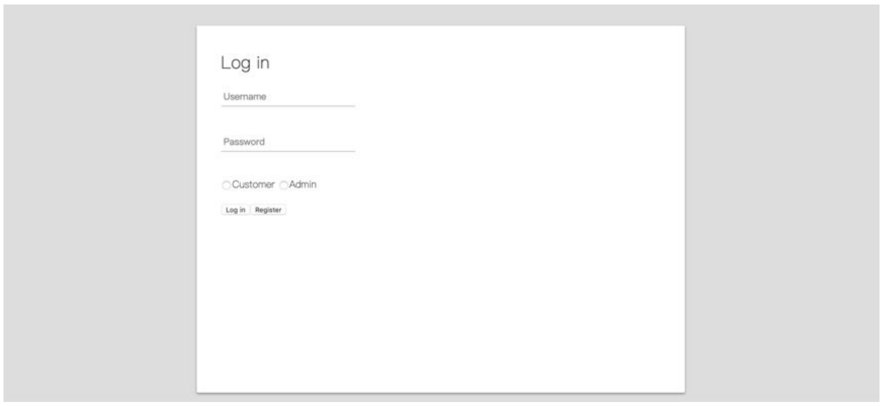
View Menu and Edit the Menu by Admin side (Manage.php+Admin.php+delete.php+changeprice.php)

Technology used: php, AJAX, html, css(local and from internet source), mysql, javascript.
These four interfaces are for managers. These four interfaces are equivalent to the background management of a restaurant. Managers can view the current order status and menu content through Manage.php. By clicking the update menu, the manager can adjust the price of the dishes, delete some dishes or add some dishes on Admin.php. Here is to query and update the database directly.

LogIn System

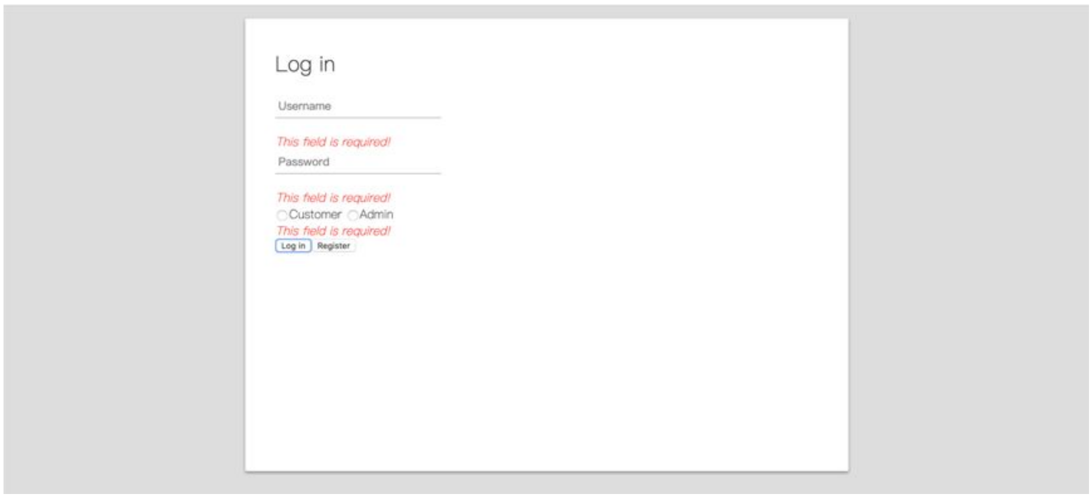
<u>LogIn.php</u>

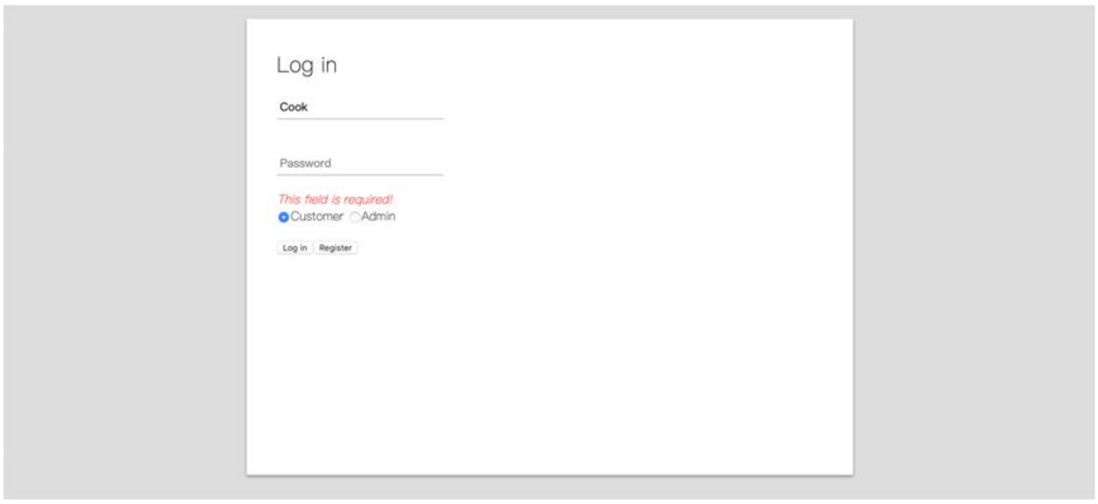
Users are required to input the information of “Username” and “Password”. If users already own accounts, they can directly log in. Or they have to register an account at first. Besides, they are required to choose the identity. As customers, they can order food online. As administrators, they can manage the whole ordering management.



The information will be checked and display errors.

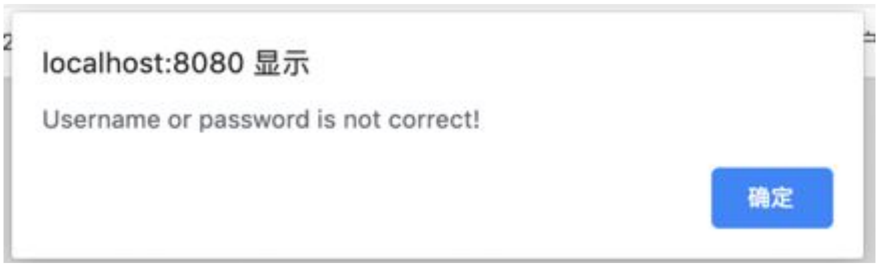
(1) None of “Username”, “Password” and “identity” can be empty.
In the page, it contains a JavaScript function. It is used to check whether the inputs are empty or not. Or it returns false so that the information of the form can not be sent out.





(2) "Username" and "Password" must be right.

When users input all the necessary information, the JavaScript function will return true and the form will be submit. Then, the PHP part will build connection with the corresponding database. If the "Username" and "Password" can not match with ones in the database, there will be an alert displaying "Username or password is not correct !" If both are right, the page will directly jump to the next page.

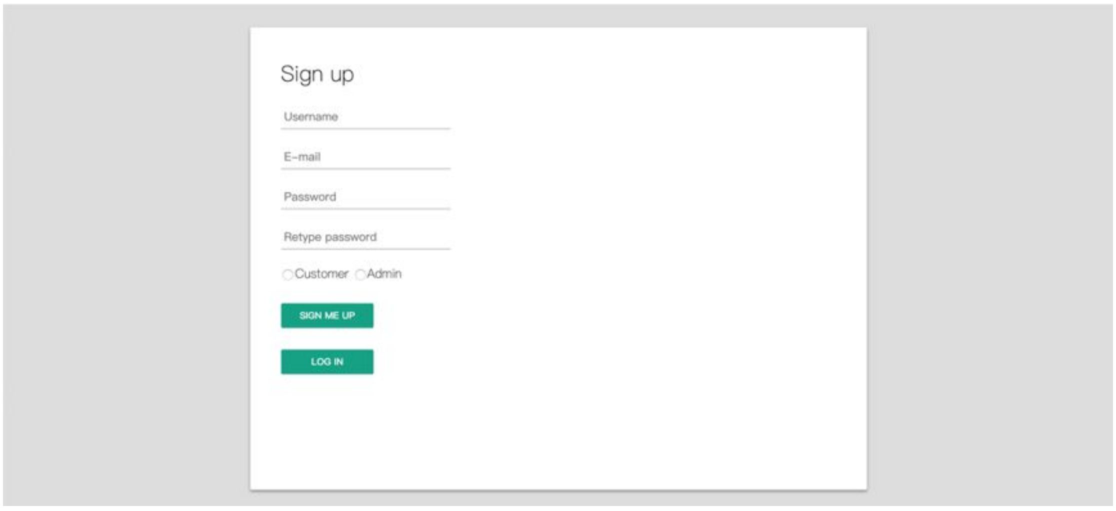


localhost:8080 显示

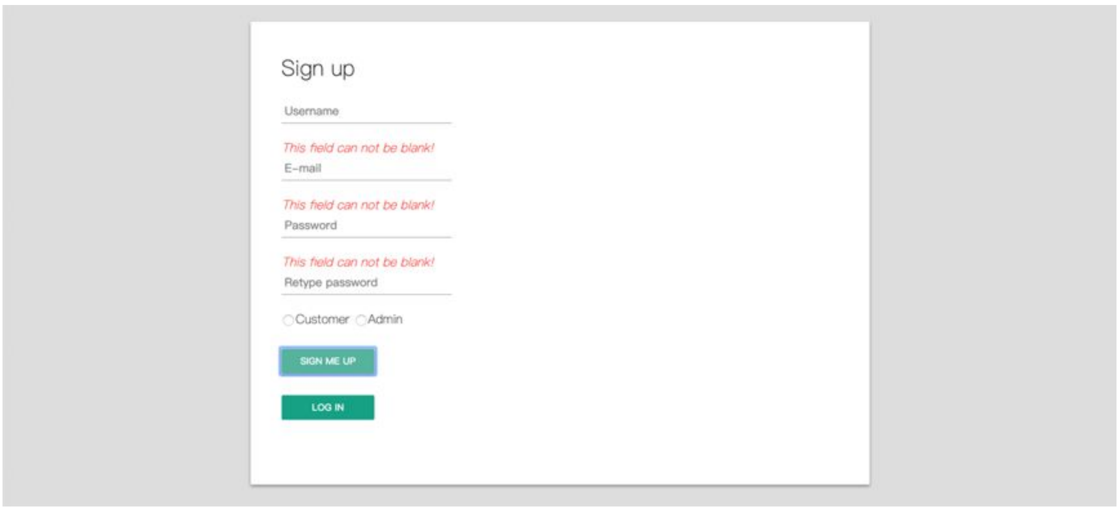
Username or password is not correct!

Register.php

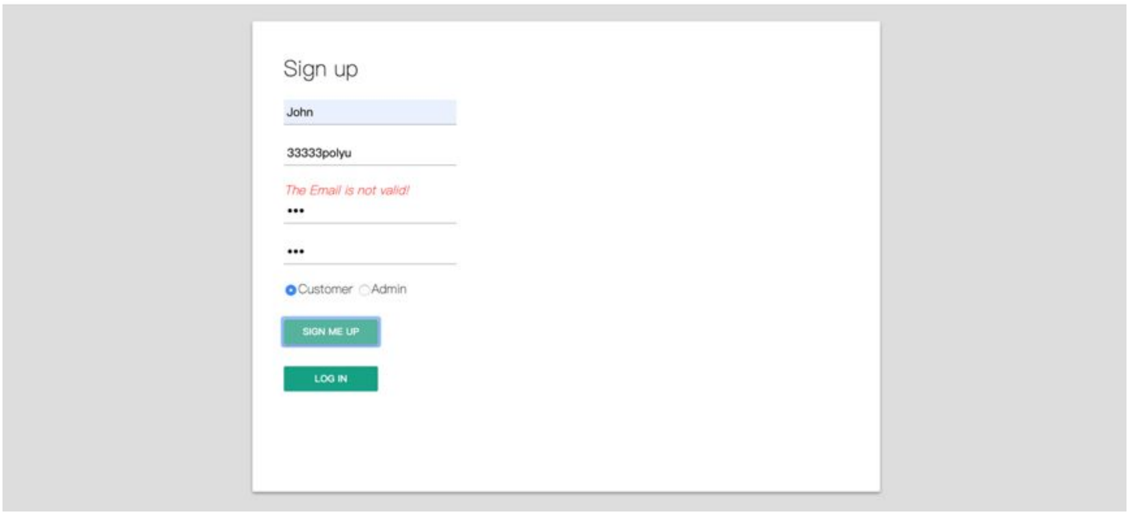
If users have no accounts, they can register a new one. All the information are necessary. "Email" must be a valid form. "Password" and "Retype password" must be the same. If all above requirements are met, the account can be created successfully.



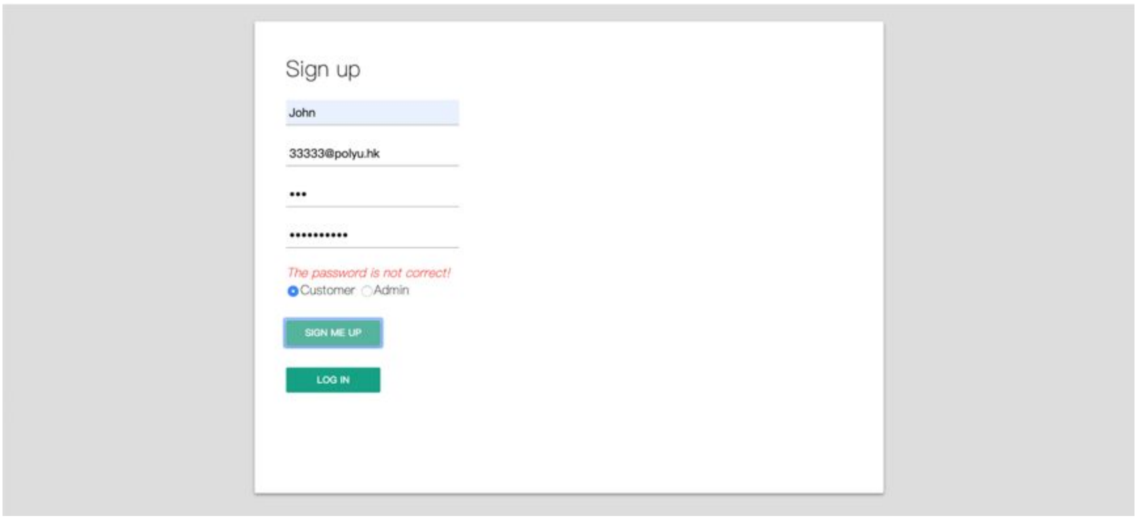
(1) None of inputs can be empty.


Sign up
Username
This field can not be blank!
E-mail
This field can not be blank!
Password
This field can not be blank!
Retype password
Customer
Admin
SIGN ME UP
LOG IN

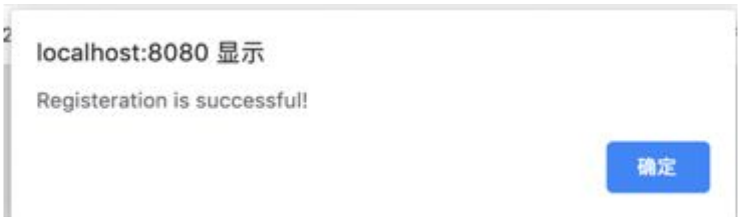
(2) Email must be valid.



(3) “Password” and “Retype password” must be the same.

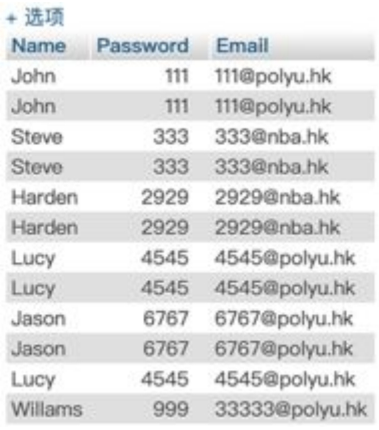


Only if all above requirements are met, the account can be created successfully. And a new row of “Customer” information in the database.


localhost:8080 显示

Registeration is successful!

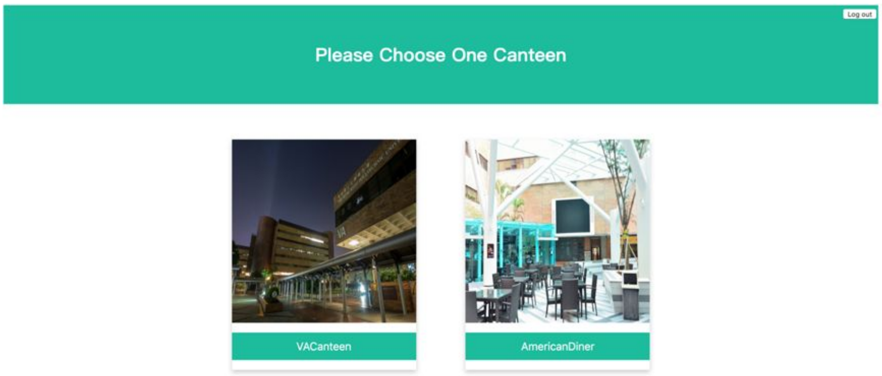
确定


+ 选项

Name	Password	Email
John	111	111@polyu.hk
John	111	111@polyu.hk
Steve	333	333@nba.hk
Steve	333	333@nba.hk
Harden	2929	2929@nba.hk
Harden	2929	2929@nba.hk
Lucy	4545	4545@polyu.hk
Lucy	4545	4545@polyu.hk
Jason	6767	6767@polyu.hk
Jason	6767	6767@polyu.hk
Lucy	4545	4545@polyu.hk
Willams	999	33333@polyu.hk

<u>Canteen.php</u>

If users log in successfully, the page will jumped to canteen page. Users can choose VACanteen or AmericanDiner.

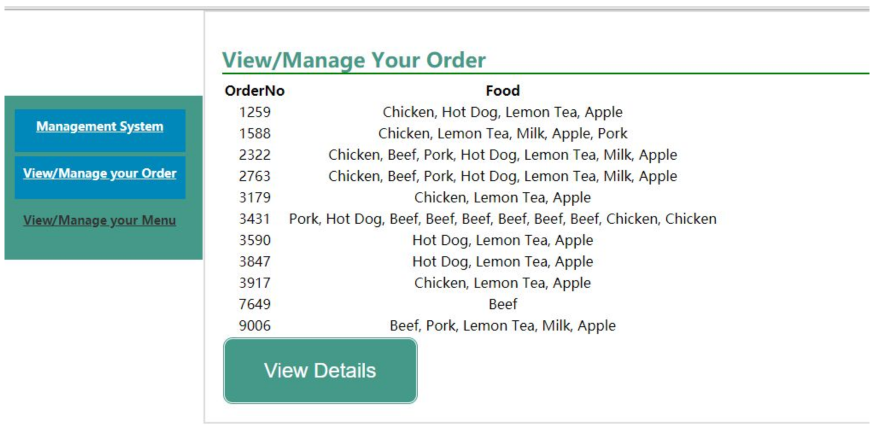


<u>Logout.php</u>

Users can log out as long as they click the log-out button. All the information will be unset and the page will jump to LogIn.php.

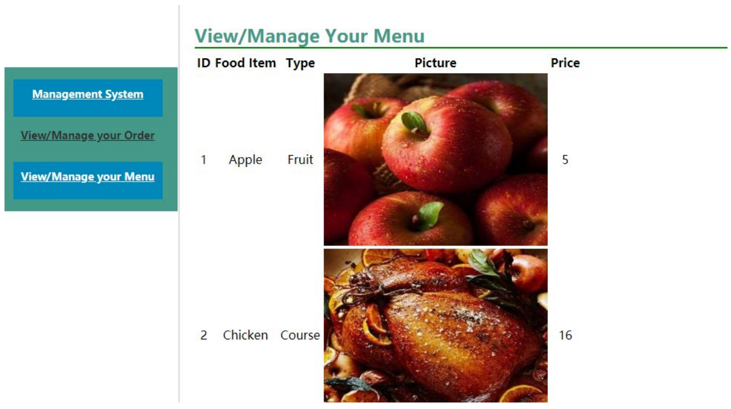
<u>Manage.php</u>

After an admin log in the system, it will directly jump to manage.php. In this page, the admin will see the current order list and menu list. If the admin clicks “View Details”, it will jump to OrderList.php. There is a sidebar that can directly reach the movement page. When the admin click the update menu, it will jump to Admin.php



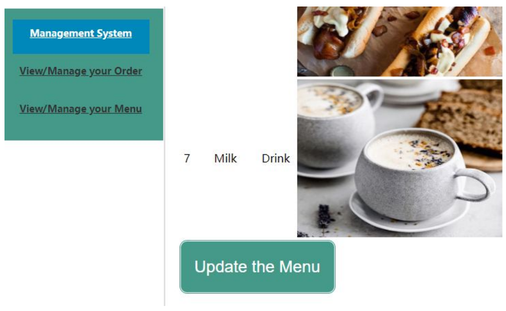
View/Manage Your Order

OrderNo	Food
1259	Chicken, Hot Dog, Lemon Tea, Apple
1588	Chicken, Lemon Tea, Milk, Apple, Pork
2322	Chicken, Beef, Pork, Hot Dog, Lemon Tea, Milk, Apple
2763	Chicken, Beef, Pork, Hot Dog, Lemon Tea, Milk, Apple
3179	Chicken, Lemon Tea, Apple
3431	Pork, Hot Dog, Beef, Beef, Beef, Beef, Beef, Beef, Chicken, Chicken
3590	Hot Dog, Lemon Tea, Apple
3847	Hot Dog, Lemon Tea, Apple
3917	Chicken, Lemon Tea, Apple
7649	Beef
9006	Beef, Pork, Lemon Tea, Milk, Apple



View/Manage Your Menu

ID	Food Item	Type	Picture	Price
1	Apple	Fruit		5
2	Chicken	Course		16



7	Milk	Drink		8

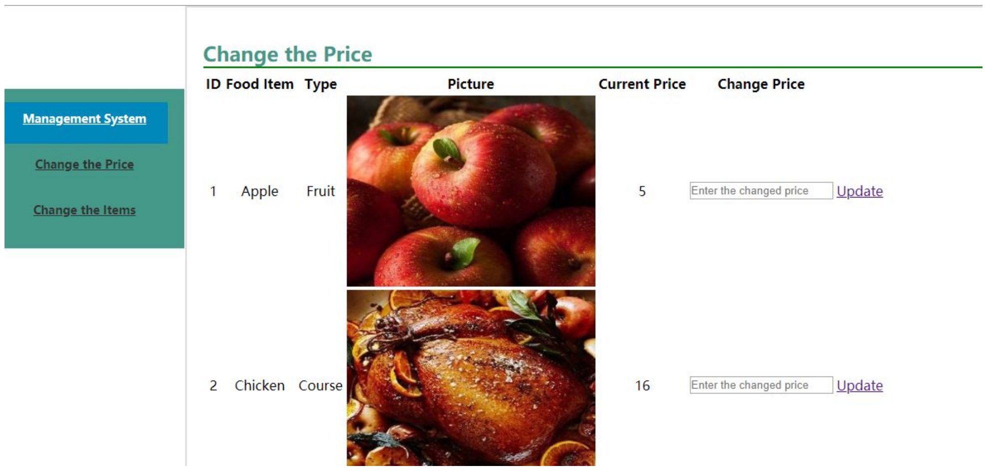
Here are the code sources contains two database queries. The first query is to get the orderlist. The second query is to get the food menu.

```
}
$sql = "SELECT OrderNo, OrderItems FROM orders ORDER BY OrderNo";
$result = $conn->query($sql);
$result->data_seek(0);

echo "<tr><th>OrderNo</th><th>Food</th></tr>";
while ($row=$result->fetch_assoc()) {
    echo "<tr align=\"center\"><td>".$row["OrderNo"]."</td><td>".$row["OrderItems"]."</td></tr>";
}
```

```
$query1 = "SELECT ID, FoodItem, Type, Price, Picture FROM ".$can."";
$result1 = $conn->query($query1);
$result1->data_seek(0);

echo "<tr><th>ID</th><th>Food Item</th><th>Type</th><th>Picture</th><th>Price</th></tr>";
while ($row1=$result1->fetch_assoc()) {
    echo "<tr align=\"center\"><td>".$row1["ID"]."</td><td>".$row1["FoodItem"]."</td>
    <td>".$row1["Type"]."</td>
    <td><img src='".$row1["Picture"].".jpg' height='250' width='250'/></td>
    <td>".$row1["Price"]."</td></tr>";
}
 $count++;
```

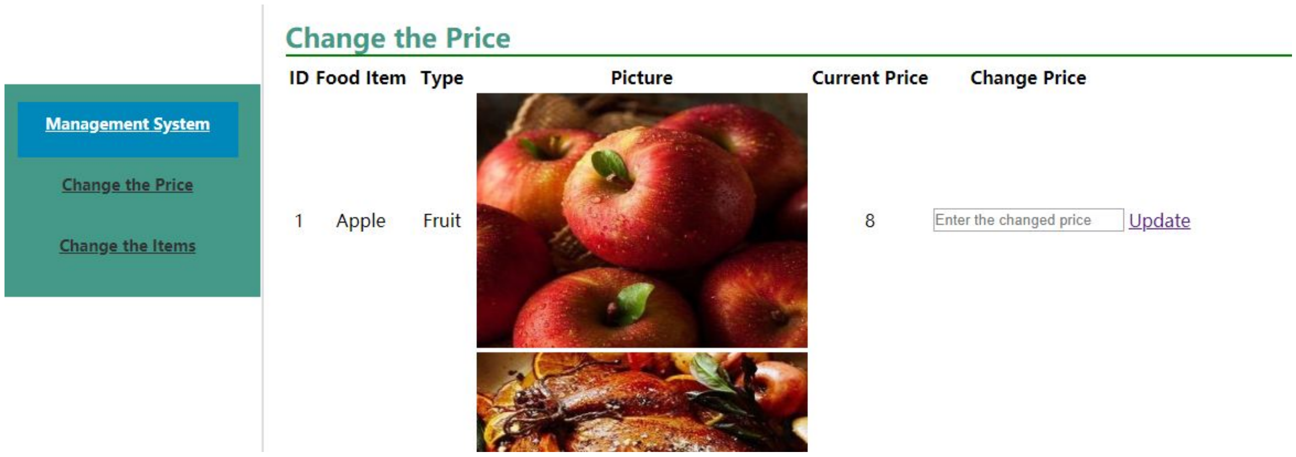
Admin.php

This page shows how the current menu. First part is to change the price. The admin can set the price into the text box. And when he clicks “Update”, it will refresh the page and the new menu will display.



Change the Price

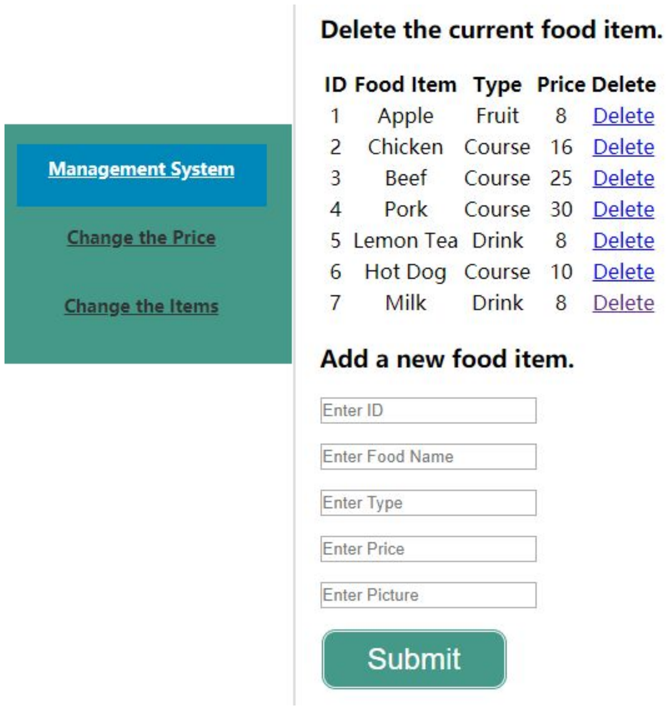
ID	Food Item	Type	Picture	Current Price	Change Price
1	Apple	Fruit		5	Enter the changed price Update
2	Chicken	Course		16	Enter the changed price Update



Change the Price

ID	Food Item	Type	Picture	Current Price	Change Price
1	Apple	Fruit		8	Enter the changed price Update

The second part is about adding or deleting the items in the menu. The admin can delete the item by clicking “Delete”. He needs to fill in all the input text boxes to add a new item.



Delete the current food item.

ID	Food Item	Type	Price	Delete
1	Apple	Fruit	8	Delete
2	Chicken	Course	16	Delete
3	Beef	Course	25	Delete
4	Pork	Course	30	Delete
5	Lemon Tea	Drink	8	Delete
6	Hot Dog	Course	10	Delete
7	Milk	Drink	8	Delete

Add a new food item.

Enter ID

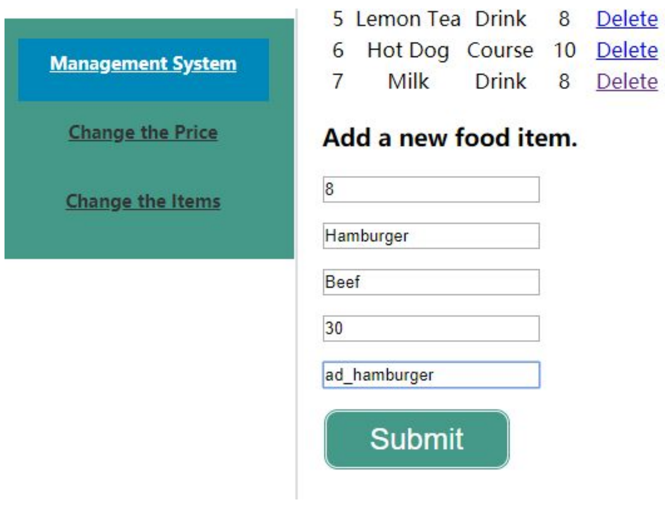
Enter Food Name

Enter Type

Enter Price

Enter Picture

Submit

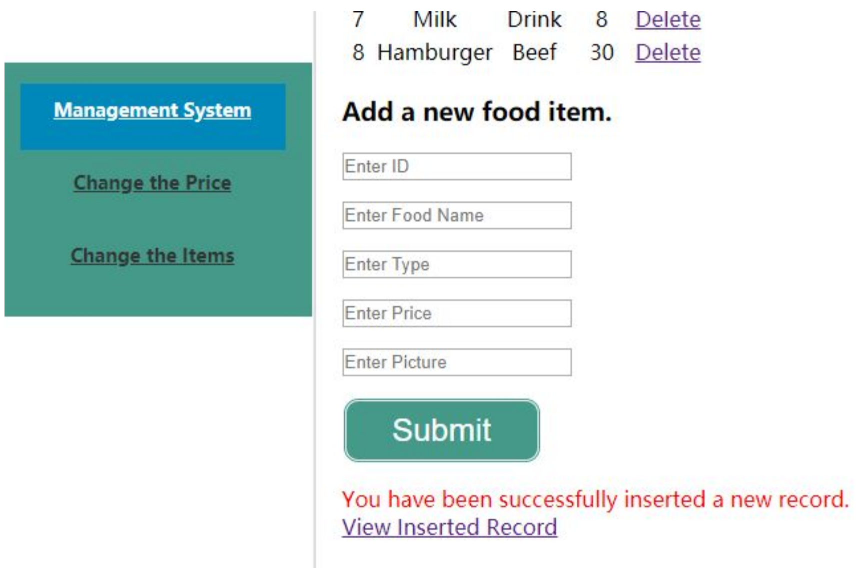


5	Lemon Tea	Drink	8	Delete
6	Hot Dog	Course	10	Delete
7	Milk	Drink	8	Delete

Add a new food item.

8
Hamburger
Beef
30
ad_hamburger

When the admin input the new food item information, it will add to the menu. The information “You have been successfully inserted a new record.” will display. And the upper part also shows the new item.

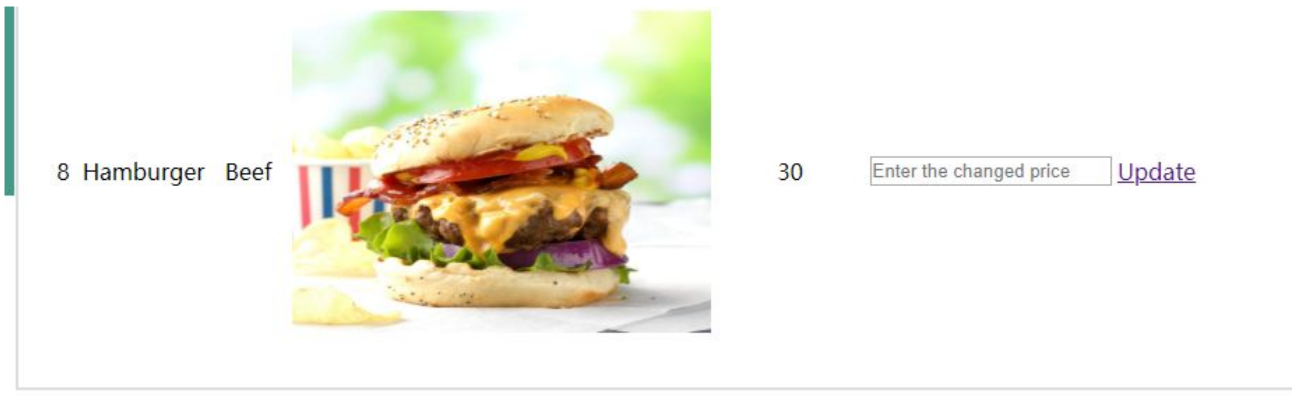


7	Milk	Drink	8	Delete
8	Hamburger	Beef	30	Delete

Add a new food item.

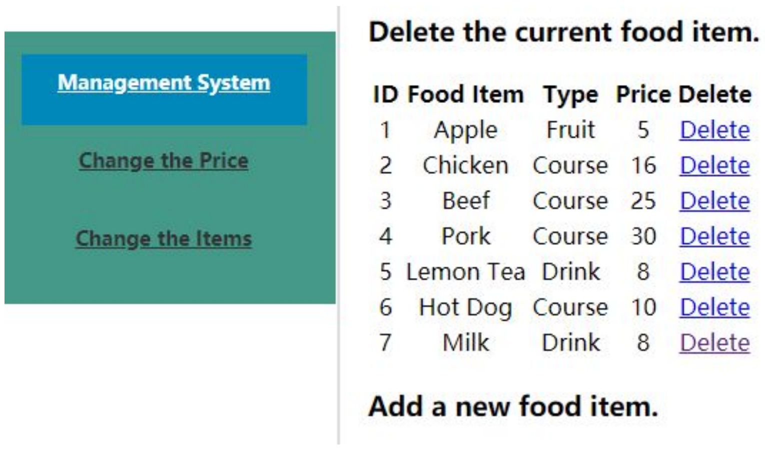
Enter ID
Enter Food Name
Enter Type
Enter Price
Enter Picture

You have been successfully inserted a new record.
View Inserted Record



8	Hamburger	Beef		30	Enter the changed price Update

After adding the item, the admin can also delete this new item.

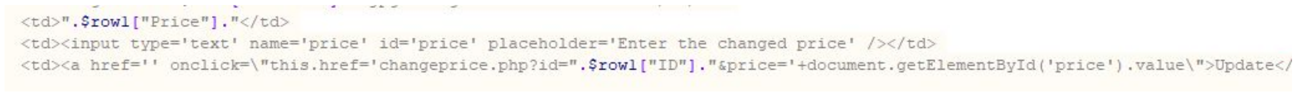


Delete the current food item.

ID	Food Item	Type	Price	Delete
1	Apple	Fruit	5	Delete
2	Chicken	Course	16	Delete
3	Beef	Course	25	Delete
4	Pork	Course	30	Delete
5	Lemon Tea	Drink	8	Delete
6	Hot Dog	Course	10	Delete
7	Milk	Drink	8	Delete

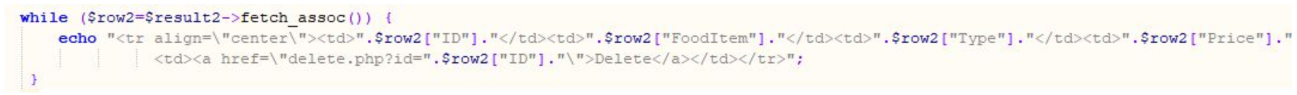
Add a new food item.

Here are the code screen captures.



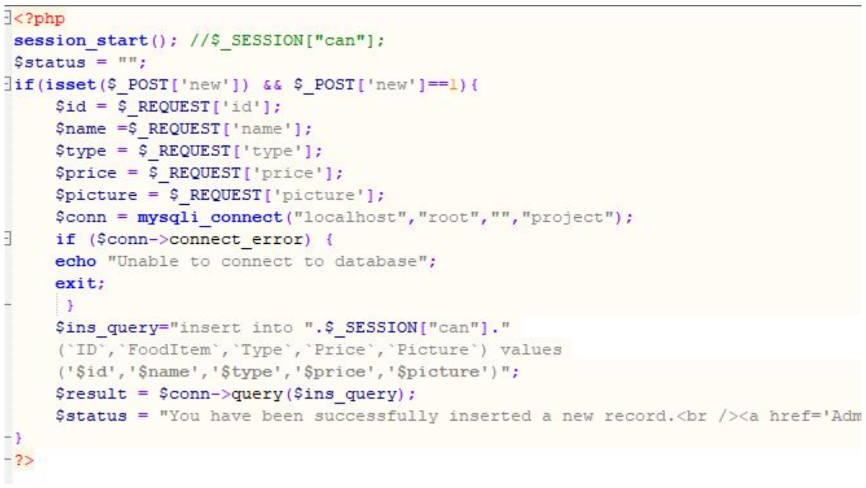
```
<td>".$rowl["Price"]."</td>
<td><input type='text' name='price' id='price' placeholder='Enter the changed price' /></td>
<td><a href='' onclick=\"this.href='changeprice.php?id=".$rowl["ID"]."&price='+document.getElementById('price').value\">Update</
```

Admin.php ask the admin to input a price and send this price to changeprice.php.



```
while ($row2=$result2->fetch_assoc()) {
    echo "<tr align=\"center\"><td>".$row2["ID"]."</td><td>".$row2["FoodItem"]."</td><td>".$row2["Type"]."</td><td>".$row2["Price"]."
            <td><a href=\"delete.php?id=".$row2["ID"]."\">Delete</a></td></tr>";
}
```

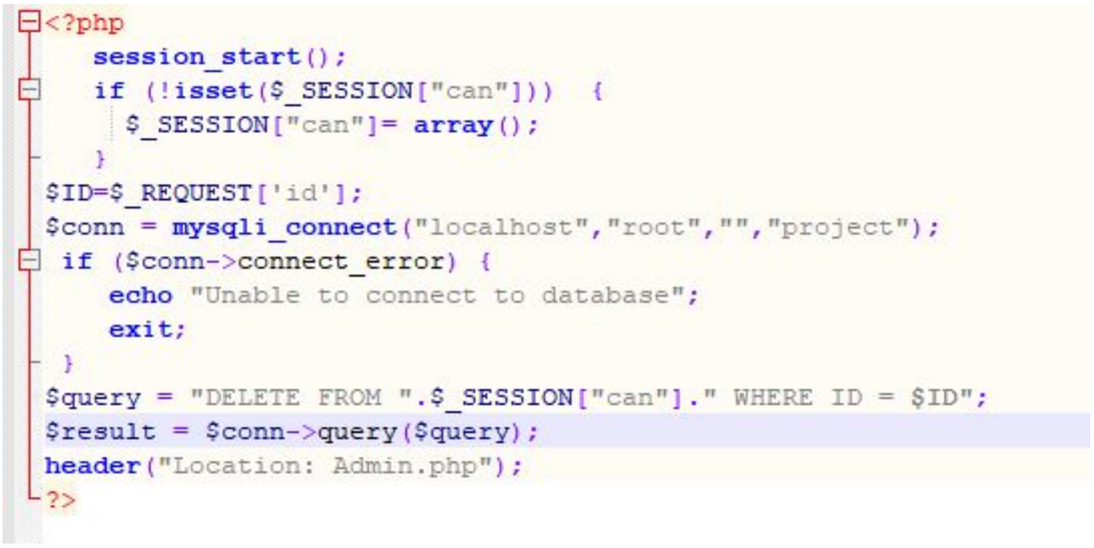
When clicking the delete, it will send the food item and delete it from database.
The admin also can send the request to insert a new food item.



```
<?php
session_start(); //$_SESSION["can"];
$status = "";
if(isset($_POST['new']) && $_POST['new']==1){
    $id = $_REQUEST['id'];
    $name =$_REQUEST['name'];
    $type = $_REQUEST['type'];
    $price = $_REQUEST['price'];
    $picture = $_REQUEST['picture'];
    $conn = mysqli_connect("localhost","root","","project");
    if ($conn->connect_error) {
    echo "Unable to connect to database";
    exit;
     }
    $ins_query="insert into ".$_SESSION["can"]."
    (`ID`,`FoodItem`,`Type`,`Price`,`Picture`) values
    ('$id','$name','$type','$price','$picture')";
    $result = $conn->query($ins_query);
    $status = "You have been successfully inserted a new record.<br /><a href='Adm
}
?>
```

<u>delete.php</u>

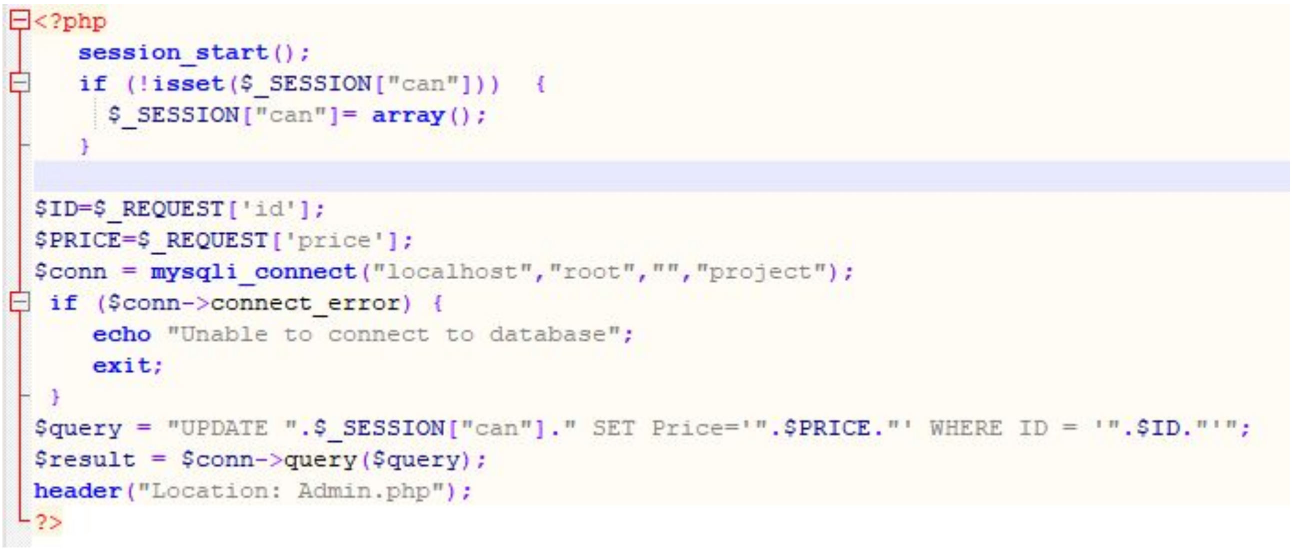
The delete.php is called when the admin click “Delete”. It will send the item id and query the database to delete this item. After deleting the item, it will return to Admin.php.



```
<?php
    session_start();
    if (!isset($_SESSION["can"]))  {
        $_SESSION["can"]= array();
    }
$ID=$_REQUEST['id'];
$conn = mysqli_connect("localhost","root","","project");
 if ($conn->connect_error) {
     echo "Unable to connect to database";
     exit;
 }
$query = "DELETE FROM ".$_SESSION["can"]." WHERE ID = $ID";
$result = $conn->query($query);
header("Location: Admin.php");
?>
```

changeprice.php

The changeprice.php is called when the admin click “Update”. It will send the item and the input price and update the database. After querying the database, it will relocate to Admin.php.



```
<?php
    session_start();
    if (!isset($_SESSION["can"]))  {
        $_SESSION["can"]= array();
    }

$ID=$_REQUEST['id'];
$PRICE=$_REQUEST['price'];
$conn = mysqli_connect("localhost","root","","project");
 if ($conn->connect_error) {
     echo "Unable to connect to database";
     exit;
 }
$query = "UPDATE ".$_SESSION["can"]." SET Price='".$PRICE."' WHERE ID = '".$ID."'";
$result = $conn->query($query);
header("Location: Admin.php");
?>
```

Shopping menu (Menu.php)

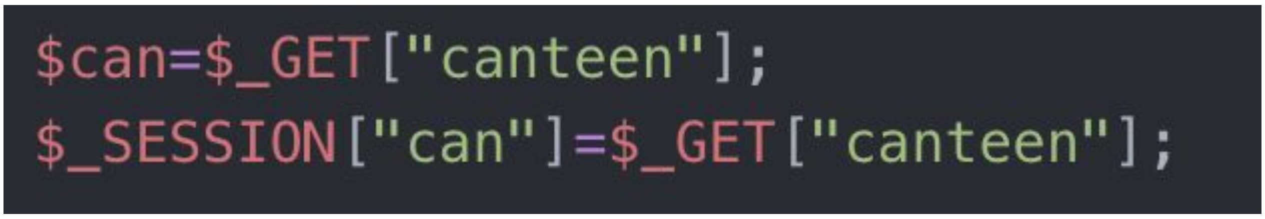
Technology used: php, html, css(local and from internet source), mysql, javascript, AJAX.

Menu.php

This page is used for customers to view the food and add the items they want into the shopping card.

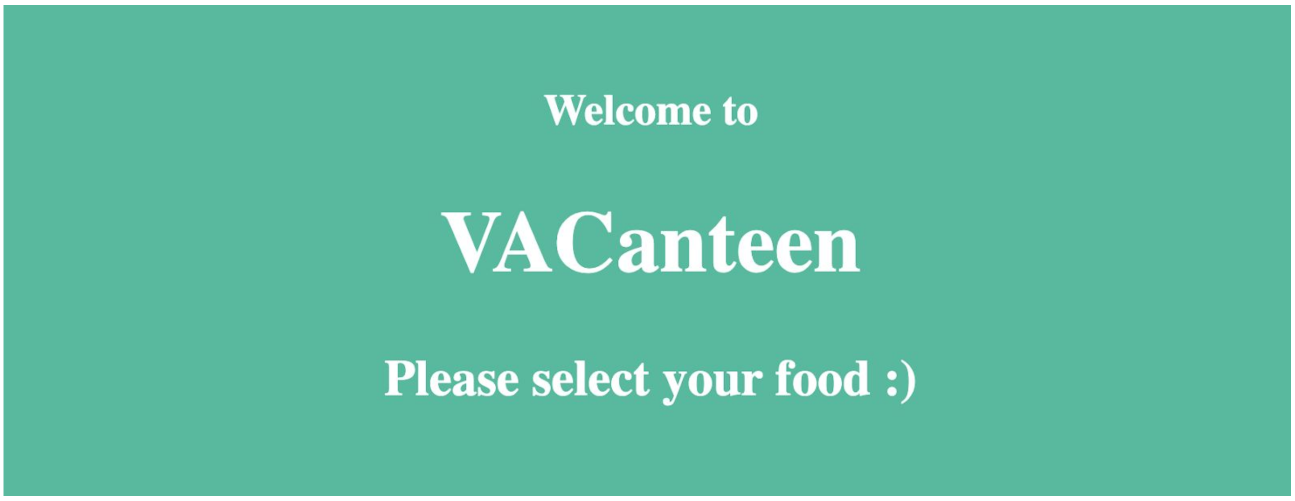
The following is the output with the technical details of this page.

After the customer chose a canteen in the previous page, the name of the selected canteen is passed to menu.php. This page will get the name of the canteen and store it in a variable. For future use, this name will also be stored in a session variable.



```
$can=$_GET["canteen"];
$_SESSION["can"]=$_GET["canteen"];
```

Customers will be welcomed by this canteen.

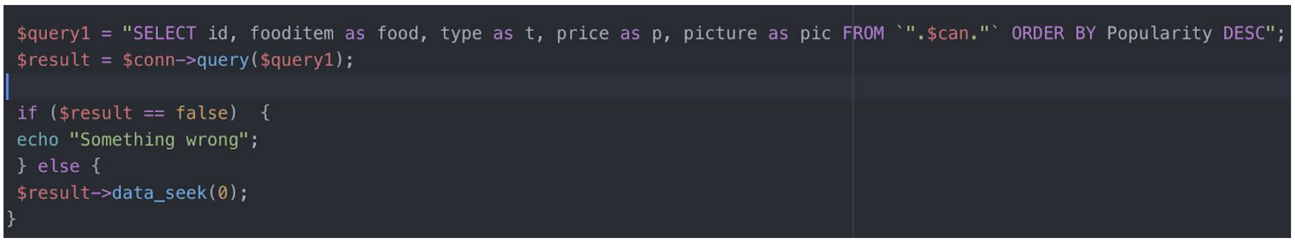


Welcome to

VACanteen

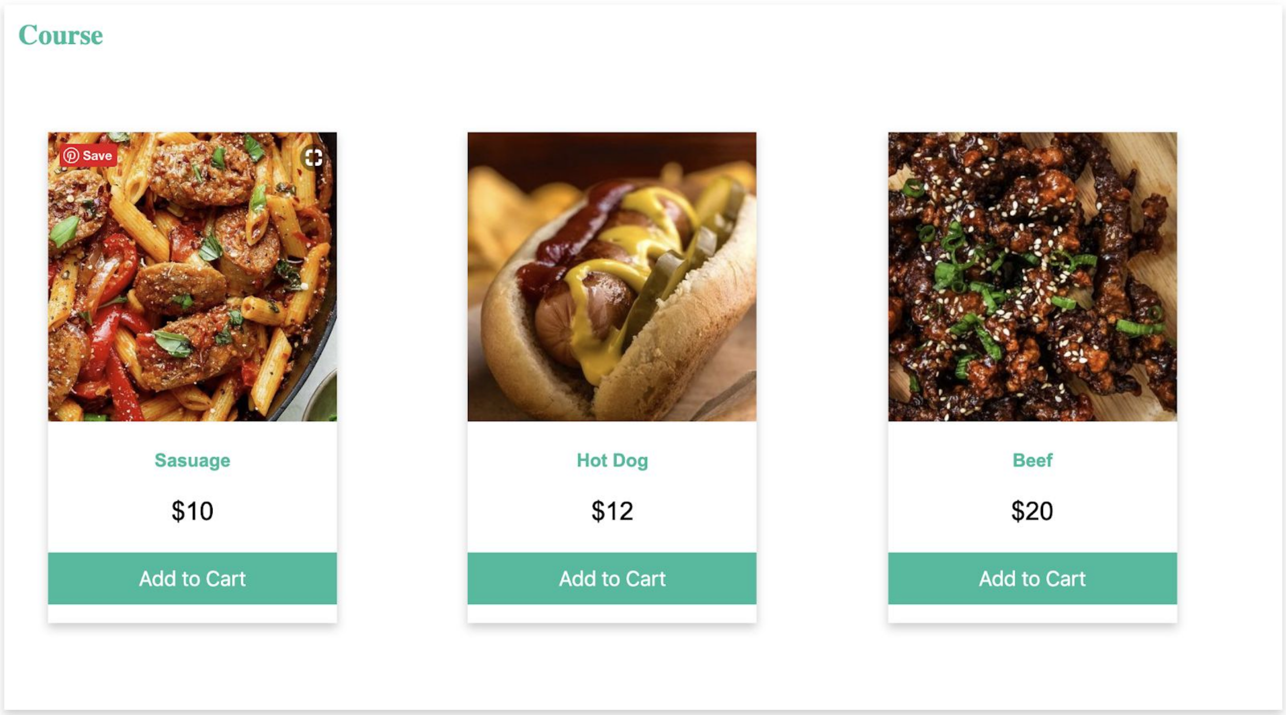
Please select your food :)

Menu.php connects to the food database to get the information. Using the canteen variable, the information of food items are selected from the chosen canteen. The food are sorted by their popularities in descending order. This means the popular food will be seen first.



```
$query1 = "SELECT id, fooditem as food, type as t, price as p, picture as pic FROM `".$can."` ORDER BY Popularity DESC";
$result = $conn->query($query1);

if ($result == false)  {
echo "Something wrong";
} else {
$result->data_seek(0);
}
```

There are 3 types of food: course, drink and fruit.



Course

Sasuage

$10

Add to Cart

Hot Dog

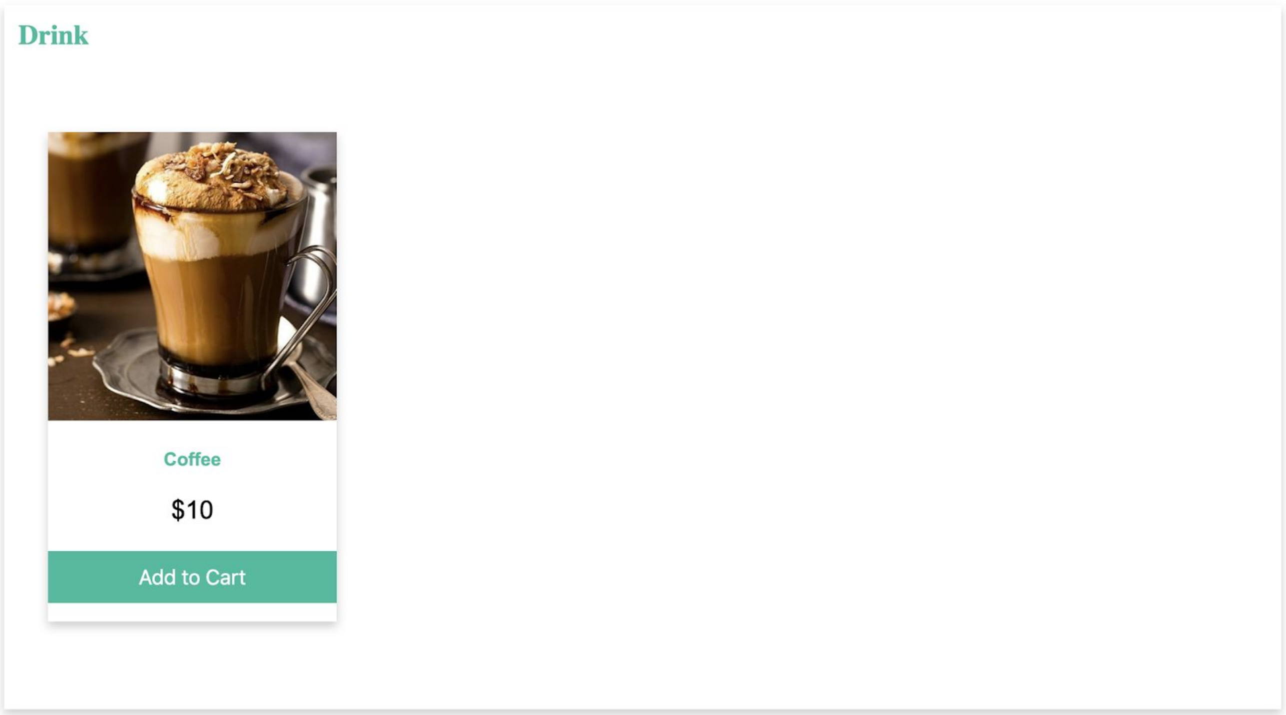
$12

Add to Cart

Beef

$20

Add to Cart

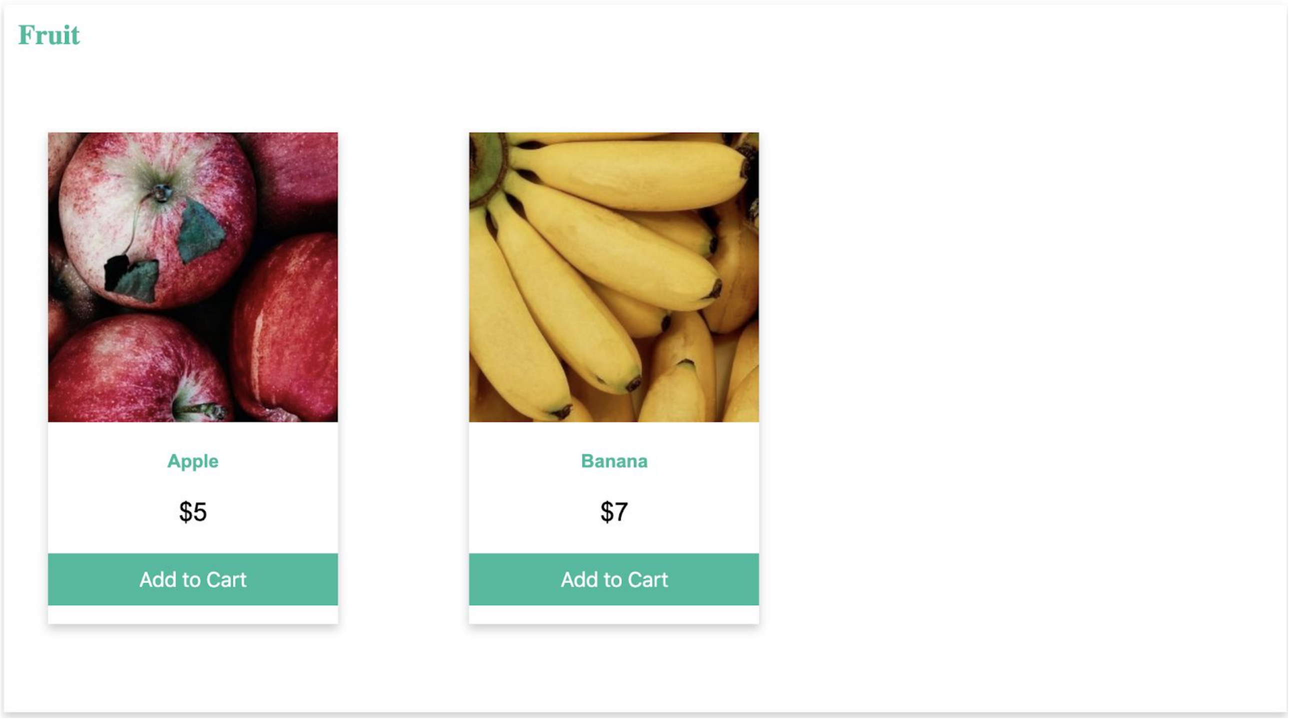


Drink

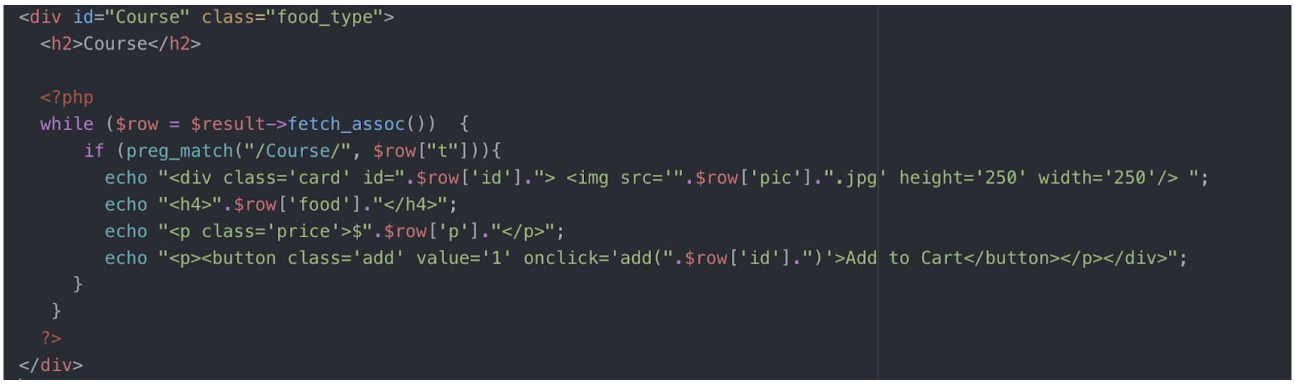
Coffee

$10

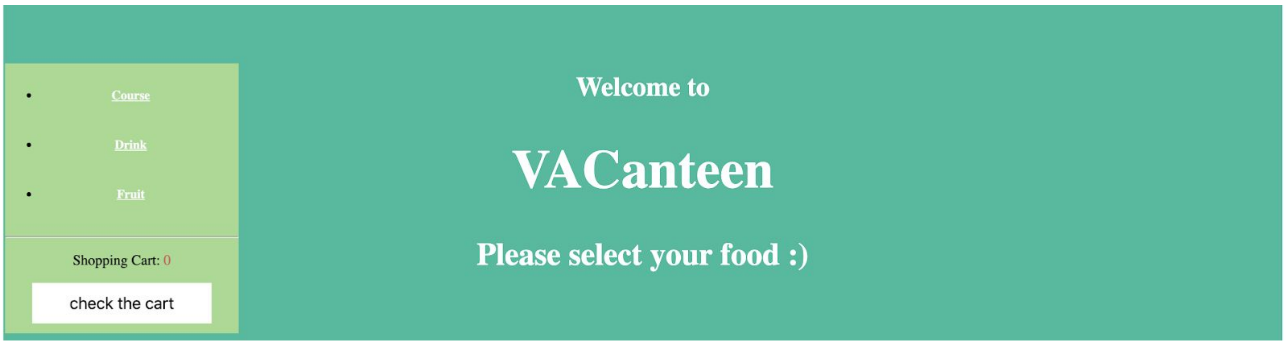
Add to Cart



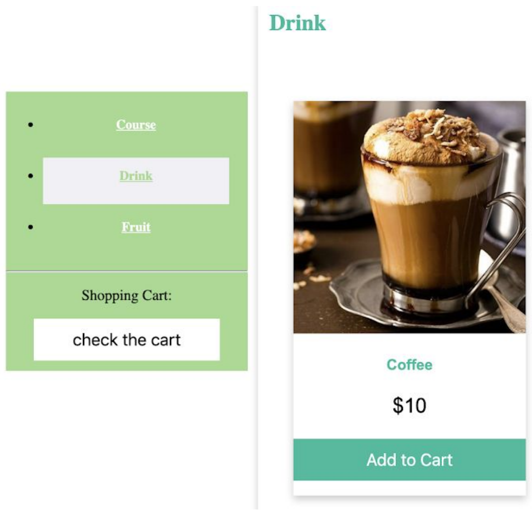
Under each type of food, a pattern comparison is performed to find food with the right type. For each food item that matches the type, a food card is generated to display the information of this food. The image source of this food uses the name of the picture in the database.


```
<div id="Course" class="food_type">
  <h2>Course</h2>

  <?php
  while ($row = $result->fetch_assoc())  {
      if (preg_match("/Course/", $row["t"])){
        echo "<div class='card' id=".$row['id']."> <img src='".$row['pic'].".jpg' height='250' width='250'/> ";
        echo "<h4>".$row['food']."</h4>";
        echo "<p class='price'>$".$row['p']."</p>";
        echo "<p><button class='add' value='1' onclick='add(".$row['id'].")'>Add to Cart</button></p></div>";
      }
    }
  ?>
</div>
```

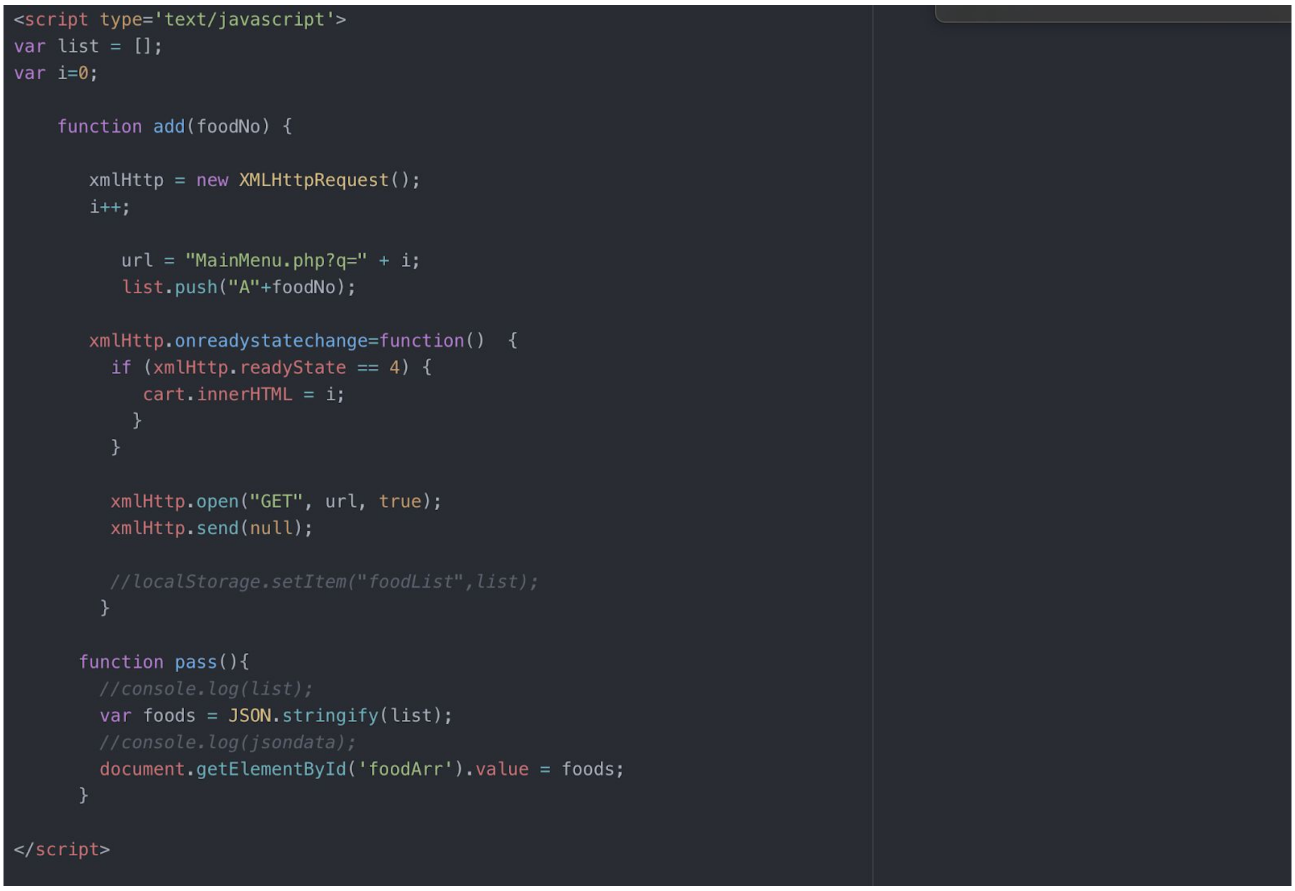
There is a side-bar at the left side of the page. This bar has two parts: the directory and the shopping cart brief.



By clicking the items in the directory, customer can be quickly directed to the food type they want.

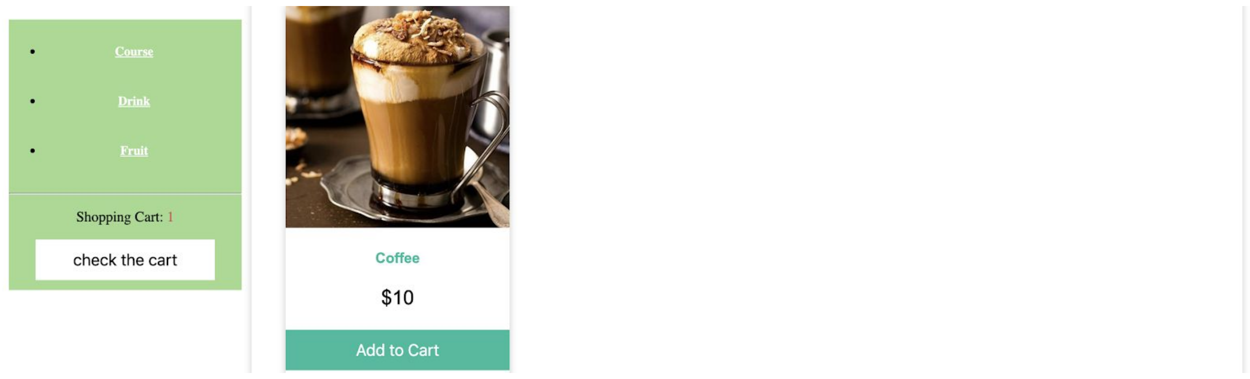


The shopping cart brief displays the number of items that are currently in the shopping cart.
Customers can add this food into the shopping cart by clicking the button “Add to Cart”. A function called “add()” will be invoked then. This function uses the id of the selected food as the input.

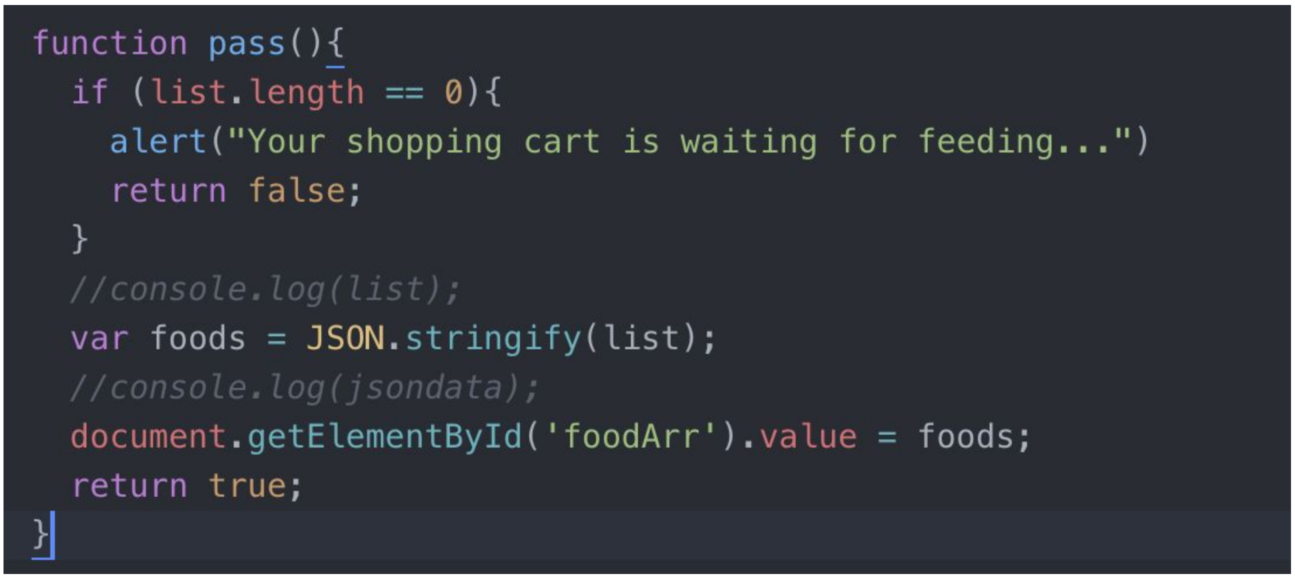


```
<script type='text/javascript'>
var list = [];
var i=0;

    function add(foodNo) {

       xmlHttp = new XMLHttpRequest();
       i++;

          url = "MainMenu.php?q=" + i;
          list.push("A"+foodNo);

       xmlHttp.onreadystatechange=function()  {
         if (xmlHttp.readyState == 4) {
            cart.innerHTML = i;
           }
         }

         xmlHttp.open("GET", url, true);
         xmlHttp.send(null);

         //localStorage.setItem("foodList",list);
        }

      function pass(){
        //console.log(list);
        var foods = JSON.stringify(list);
        //console.log(jsondata);
        document.getElementById('foodArr').value = foods;
      }

</script>
```

There is a variable called i. It represents the number of items inside the cart. Each time this function is called, it will add one to i. The AJAX is used to update the number after the shopping cart.

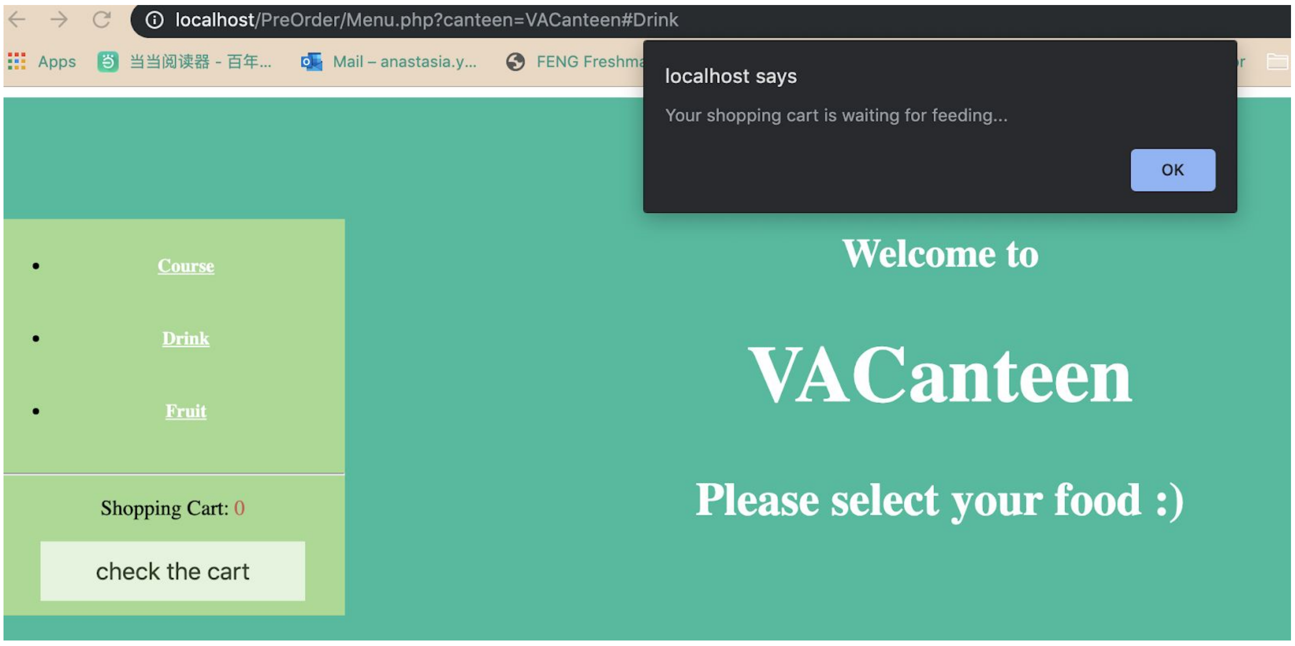


The function will also add the id of the food selected to an array called list. Once the customer click the “check the cart” button, a function called pass() will be called. This function stores list in a JSON array.



```
function pass(){
  if (list.length == 0){
    alert("Your shopping cart is waiting for feeding...")
    return false;
  }
  //console.log(list);
  var foods = JSON.stringify(list);
  //console.log(jsondata);
  document.getElementById('foodArr').value = foods;
  return true;
}
```

If there’s no item in the shopping cart, a window asking the customer to add something into their cart will prompt up.



localhost/PreOrder/Menu.php?canteen=VACanteen#Drink

localhost says

Your shopping cart is waiting for feeding...

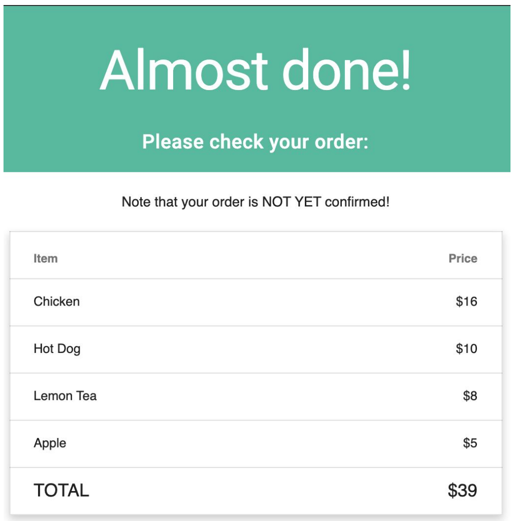
If there is something in the cart, the page will then forward to the ConfirmScreen.php.

```
<center>
    <form method="get" action="confirmScreen.php" onsubmit="return pass();">
      <p>Shopping Cart: <span class="sp" id="cart">0</span></p>
      <input type="submit" value="check the cart" class="check"/>

      <!-- A hidden txt field for passing the json food list string to -->
      <input type="hidden" id="foodArr" name="foodArr">

    </form>
  </center>
```

Order Confirmation and Checkout (ConfirmScreen.php + Checkout.php)
Technology used: php, html, css(local and from internet source), mysql, javascript.

These two pages are for the users to confirm their order and make some additional orders, such as order for seasonings, ask for an extra plastic bag and enter the desired pick up time. After the user presses the confirm button, a receipt will be generated and the order will be sent to the database.
The following is the output along with the technical details of these pages:

<u>ConfirmScreen.php</u>
After the user presses "view cart" in the previous page, the selected meals will be sent to confirmScreen.php as a JSON array that contains the IDs of the meals.
 and this array will be decoded into a php array in confirmScreen.php. Based on the IDs in the array, confirmScreen.php will perform query in the database "food" to fatch the names and prices of the ordered items, and finally generate a list for the user to confirm:

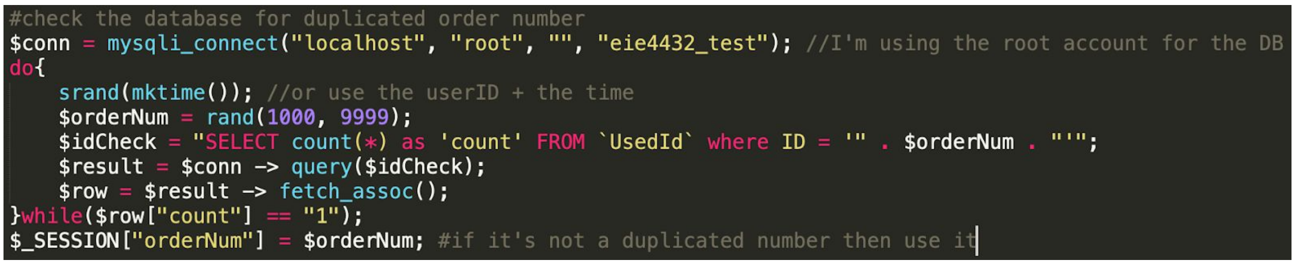

Almost done!

Please check your order:

Note that your order is NOT YET confirmed!

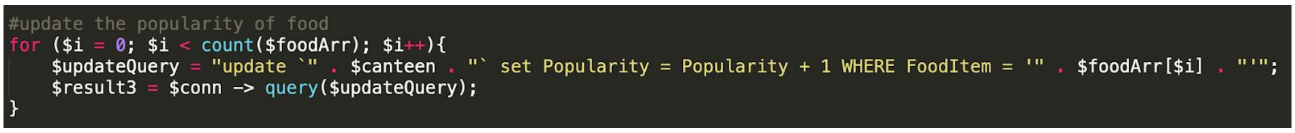
Item	Price
Chicken	$16
Hot Dog	$10
Lemon Tea	$8
Apple	$5
TOTAL	$39

Meanwhile the page will generate a random 4-digit code as the Order Number. To prevent duplication, the page will perform a query in the database "UsedID" that contains all the used order numbers of the day. If the generated order number is found used, the page will generate another one until the number is confirmed unused. The following is the code segment for generating the order number:



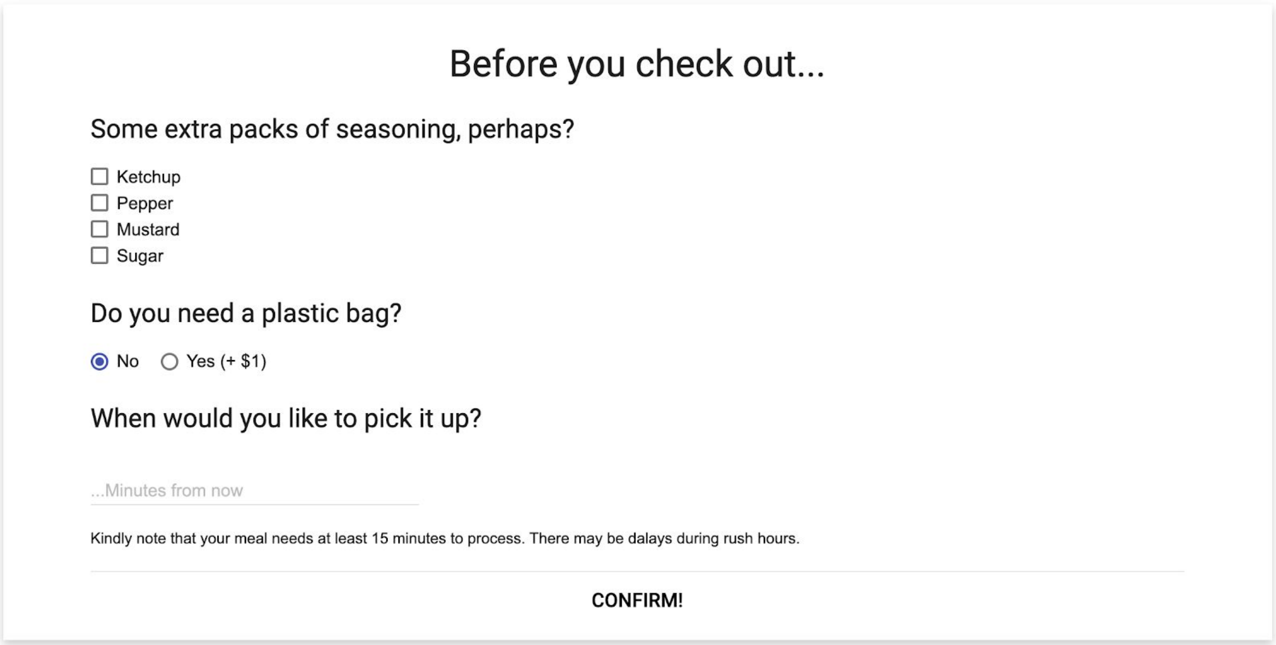
```
#check the database for duplicated order number
$conn = mysqli_connect("localhost", "root", "", "eie4432_test"); //I'm using the root account for the DB
do{
    srand(mktime()); //or use the userID + the time
    $orderNum = rand(1000, 9999);
    $idCheck = "SELECT count(*) as 'count' FROM `UsedId` where ID = '" . $orderNum . "'";
    $result = $conn -> query($idCheck);
    $row = $result -> fetch_assoc();
}while($row["count"] == "1");
$_SESSION["orderNum"] = $orderNum; #if it's not a duplicated number then use it
```

Other than that, this page will also update the popularity of the meals based on the ordered items. This is to realize the function of sorting the menu based on the popularity of the meals.



```
#update the popularity of food
for ($i = 0; $i < count($foodArr); $i++){
    $updateQuery = "update `" . $canteen . "` set Popularity = Popularity + 1 WHERE FoodItem = '" . $foodArr[$i] . "'";
    $result3 = $conn -> query($updateQuery);
}
```

Below the food list is the generated list of order items, there's a form for the user to make additional orders and indicate the preferred pickup time:



Before you check out...

Some extra packs of seasoning, perhaps?

☐ Ketchup
☐ Pepper
☐ Mustard
☐ Sugar

Do you need a plastic bag?

[✓ No] Yes (+ $1)

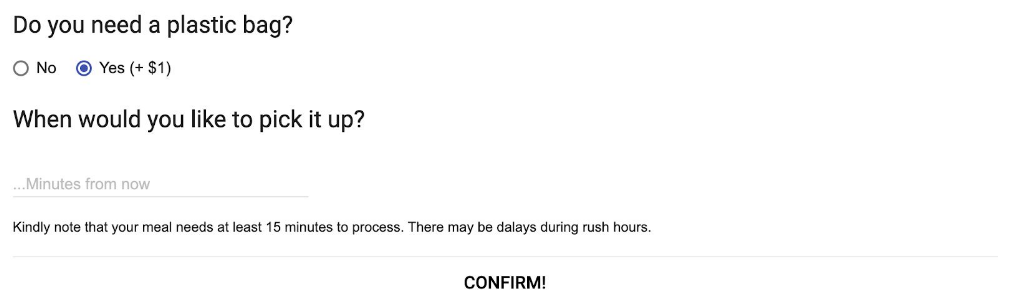
When would you like to pick it up?

...Minutes from now

Kindly note that your meal needs at least 15 minutes to process. There may be dalays during rush hours.

CONFIRM!

The user is allowed to ask for extra packs of seasoning through the checklist. Below that there's two radio buttons for the user to indicate whether he or she needs a plastic bag. As we planned to charge an extra dollar for a plastic bag, when the user clicked "yes" at the "Do you need a plastic bag" session, a message will pop up from the bottom of the screen, notifying the user that an extra dollar is charged. (This pop up message function is included in the css and javascript source hosted on this online source: https://getmdl.io/)



Do you need a plastic bag?

No [✓ Yes (+ $1)]

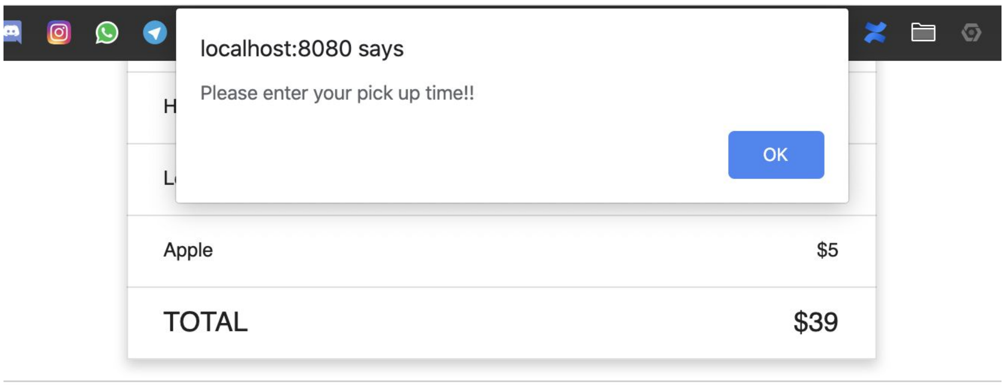
When would you like to pick it up?

...Minutes from now

Kindly note that your meal needs at least 15 minutes to process. There may be dalays during rush hours.

CONFIRM!

After that there's a textbox that allows the user to specify the pickup time by inputting how many minutes after now will the user pick up the food. If the user attempts to submit the form with this field left blank, an error message will pop up to alert the user about the blank text field.

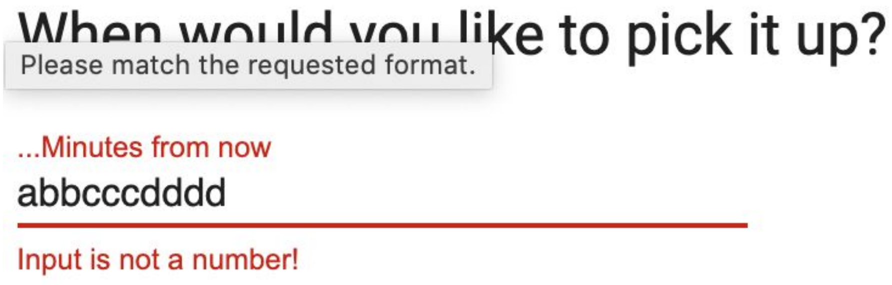


localhost:8080 says

Please enter your pick up time!!

Apple	$5
TOTAL	$39

Additionally, the script from the online source has provided a function to check for non-numeric input within the textbox.



When would you like to pick it up?

Please match the requested format.

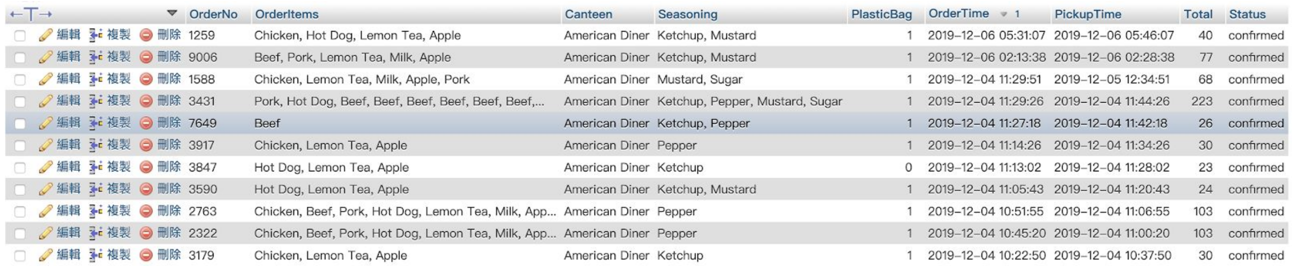
...Minutes from now

abbcccdddd

Input is not a number!

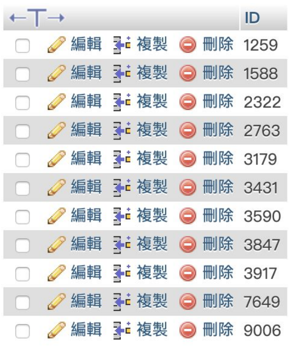
Checkout.php

Finally, after everything is confirmed, the order information, including the ordered items, the seasoning picked, whether a plastic bag is needed and the specified pickup time, will be passed to *checkout.php* by session variables. *checkout.php* will gather all the order information and write the order into the "Orders" table.



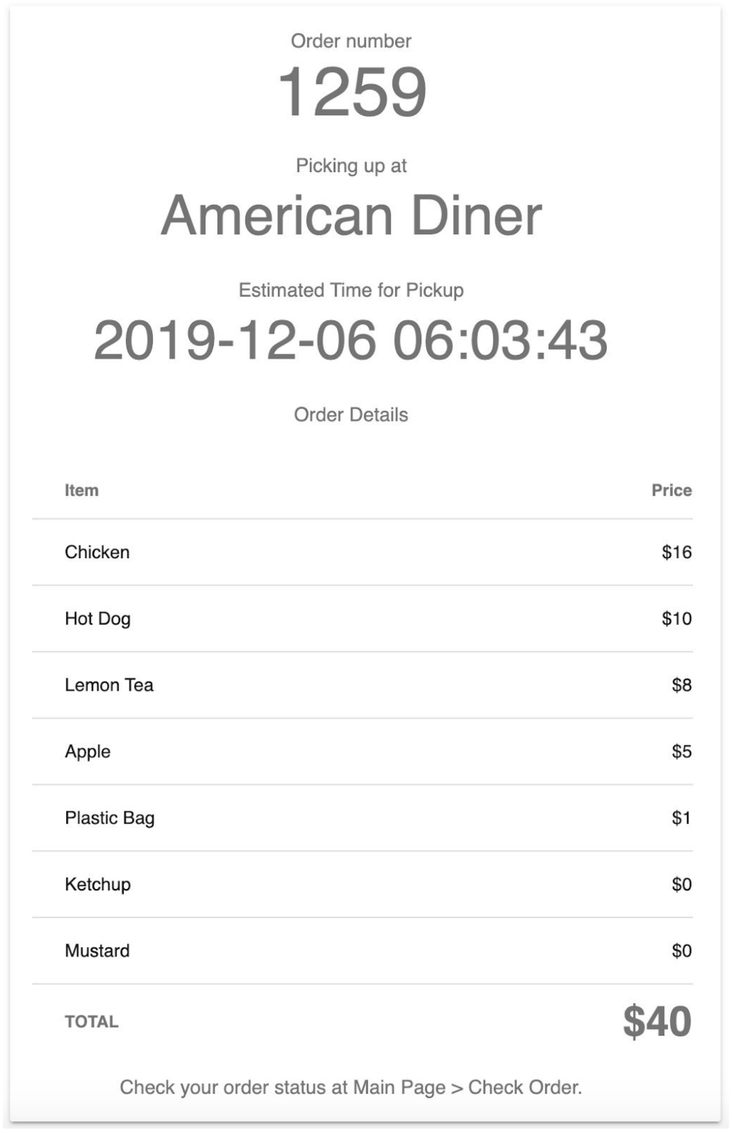
←T→	OrderNo	OrderItems	Canteen	Seasoning	PlasticBag	OrderTime ▼ 1	PickupTime	Total	Status
編輯 複製 刪除	1259	Chicken, Hot Dog, Lemon Tea, Apple	American Diner	Ketchup, Mustard	1	2019-12-06 05:31:07	2019-12-06 05:46:07	40	confirmed
編輯 複製 刪除	9006	Beef, Pork, Lemon Tea, Milk, Apple	American Diner	Ketchup, Mustard	1	2019-12-06 02:13:38	2019-12-06 02:28:38	77	confirmed
編輯 複製 刪除	1588	Chicken, Lemon Tea, Milk, Apple, Pork	American Diner	Mustard, Sugar	1	2019-12-04 11:29:51	2019-12-05 12:34:51	68	confirmed
編輯 複製 刪除	3431	Pork, Hot Dog, Beef, Beef, Beef, Beef, Beef, Beef,...	American Diner	Ketchup, Pepper, Mustard, Sugar	1	2019-12-04 11:29:26	2019-12-04 11:44:26	223	confirmed
編輯 複製 刪除	7649	Beef	American Diner	Ketchup, Pepper	1	2019-12-04 11:27:18	2019-12-04 11:42:18	26	confirmed
編輯 複製 刪除	3917	Chicken, Lemon Tea, Apple	American Diner	Pepper	1	2019-12-04 11:14:26	2019-12-04 11:34:26	30	confirmed
編輯 複製 刪除	3847	Hot Dog, Lemon Tea, Apple	American Diner	Ketchup	0	2019-12-04 11:13:02	2019-12-04 11:28:02	23	confirmed
編輯 複製 刪除	3590	Hot Dog, Lemon Tea, Apple	American Diner	Ketchup, Mustard	1	2019-12-04 11:05:43	2019-12-04 11:20:43	24	confirmed
編輯 複製 刪除	2763	Chicken, Beef, Pork, Hot Dog, Lemon Tea, Milk, App...	American Diner	Pepper	1	2019-12-04 10:51:55	2019-12-04 11:06:55	103	confirmed
編輯 複製 刪除	2322	Chicken, Beef, Pork, Hot Dog, Lemon Tea, Milk, App...	American Diner	Pepper	1	2019-12-04 10:45:20	2019-12-04 11:00:20	103	confirmed
編輯 複製 刪除	3179	Chicken, Lemon Tea, Apple	American Diner	Ketchup	1	2019-12-04 10:22:50	2019-12-04 10:37:50	30	confirmed

Meanwhile the order number will also be written to the "UsedID" table. This table holds all the used order numbers in order to prevent duplication. Theoretically the table will be clear at the end of the day.



←T→	ID
編輯 複製 刪除	1259
編輯 複製 刪除	1588
編輯 複製 刪除	2322
編輯 複製 刪除	2763
編輯 複製 刪除	3179
編輯 複製 刪除	3431
編輯 複製 刪除	3590
編輯 複製 刪除	3847
編輯 複製 刪除	3917
編輯 複製 刪除	7649
編輯 複製 刪除	9006

Finally a receipt will be generated by the page, indicating the order number, pick up venue and time, and the detailed information of the order.



Order number

1259

Picking up at

American Diner

Estimated Time for Pickup

2019-12-06 06:03:43

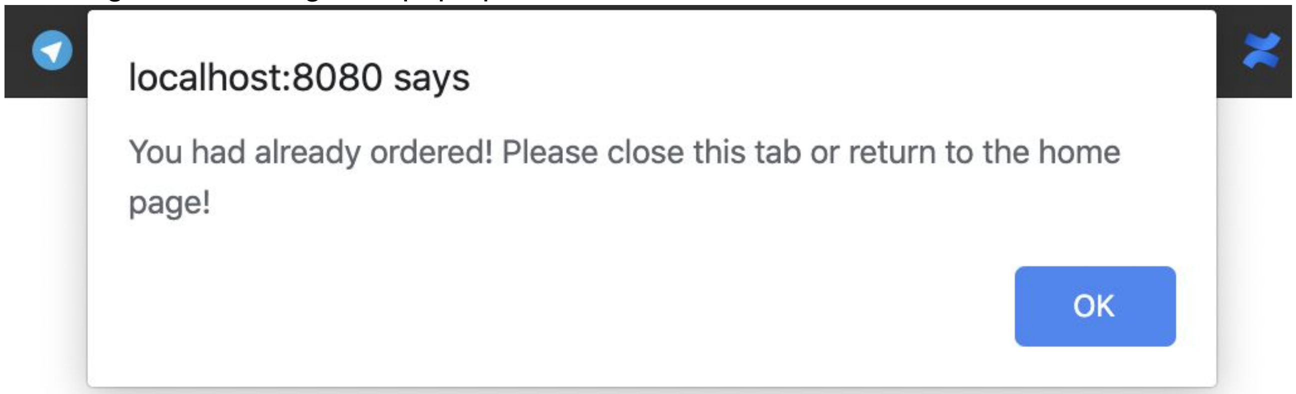
Order Details

Item	Price
Chicken	$16
Hot Dog	$10
Lemon Tea	$8
Apple	$5
Plastic Bag	$1
Ketchup	$0
Mustard	$0
TOTAL	$40

Check your order status at Main Page > Check Order.

By this point the order is finished and confirmed by the canteen staff. The user can choose to close the tab or return to the main menu page to make another order.

To avoid duplicate orders (mainly caused by the user refreshing the checkout page), a checking mechanism will be triggered whenever the page is loaded, as the php code for writing the order into the database is always executed when the page is loaded. Since the order number will remain the same for the same session, the checking mechanism will perform a query to see if the order number of the current session is already in the database. If yes then it means that the user has already completed the order, and the following error message will pop up:



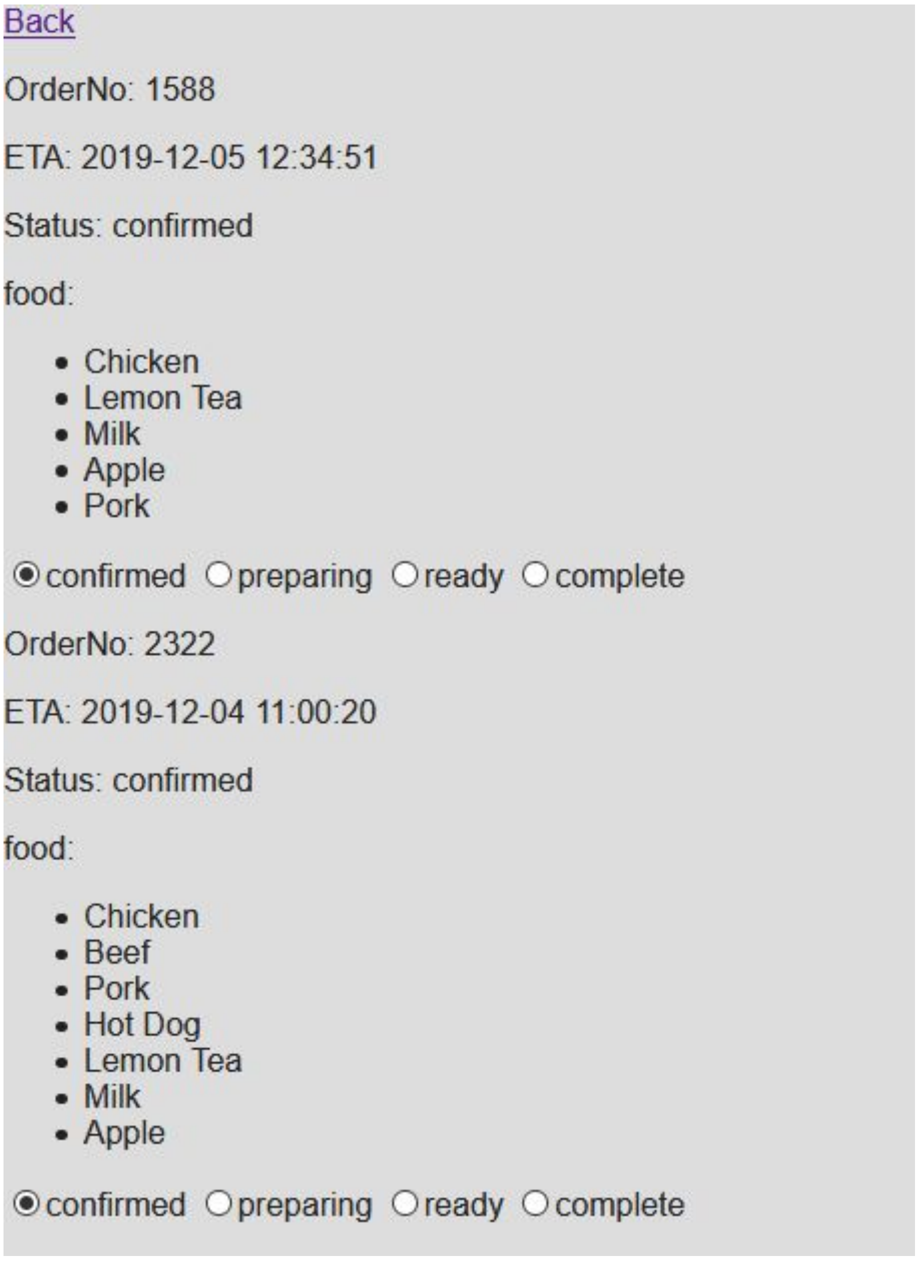
localhost:8080 says

You had already ordered! Please close this tab or return to the home page!

This way, only the order with an unused order number (meaning that the order is “fresh”) will be written into the database.

Updating Orders (OrderList.php + processing.php)
Technology used: .php, html, css, mysql, javascript.

The orderList.php generates a list of orders, displaying the OrderNo, Estimated Time of Arrival (ETA), status and ordered food. The top and the bottom page has a back page to redirect to the manage.html. Here is an excerpt of the page.


Back

OrderNo: 1588

ETA: 2019-12-05 12:34:51

Status: confirmed

food:

- Chicken
- Lemon Tea
- Milk
- Apple
- Pork

[✓ confirmed] preparing ready complete

OrderNo: 2322

ETA: 2019-12-04 11:00:20

Status: confirmed

food:

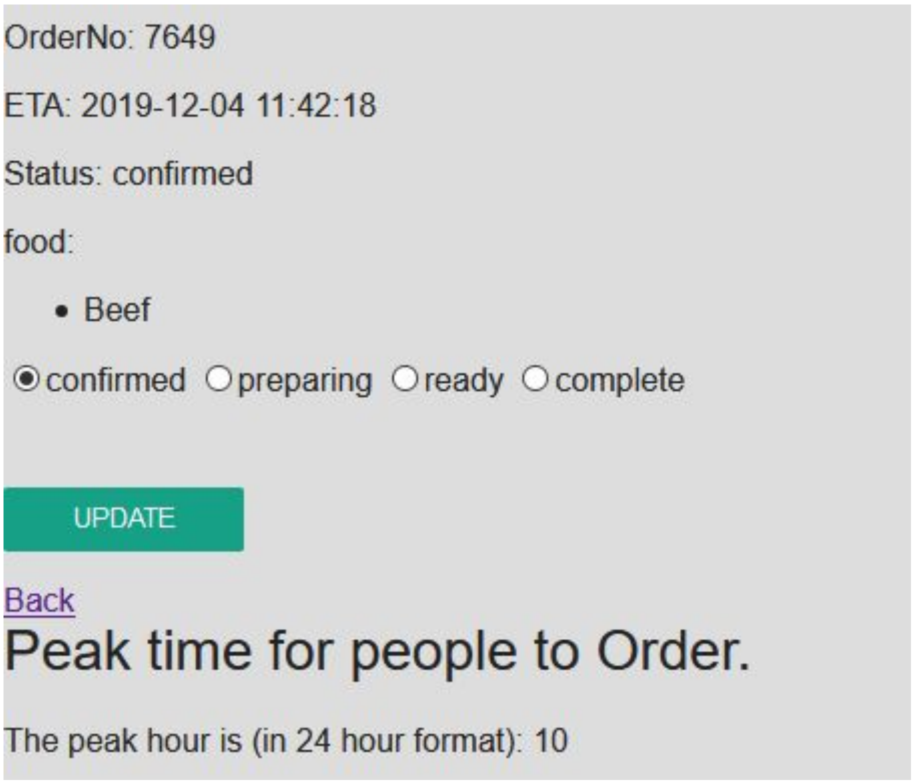
- Chicken
- Beef
- Pork
- Hot Dog
- Lemon Tea
- Milk
- Apple

[✓ confirmed] preparing ready complete

In order to display the orders, the following query is processed in php:

```
SELECT OrderNo, OrderItems, PickupTime, Status FROM orders ORDER
BY OrderNo
```

The admin can update the status by issuing either confirmed, preparing, ready and complete. Once all changes have been made, click the update button at the bottom of the page. There peak time is also displayed at the bottom of the page.


OrderNo: 7649

ETA: 2019-12-04 11:42:18

Status: confirmed

food:

- Beef

[✓ confirmed] preparing ready complete

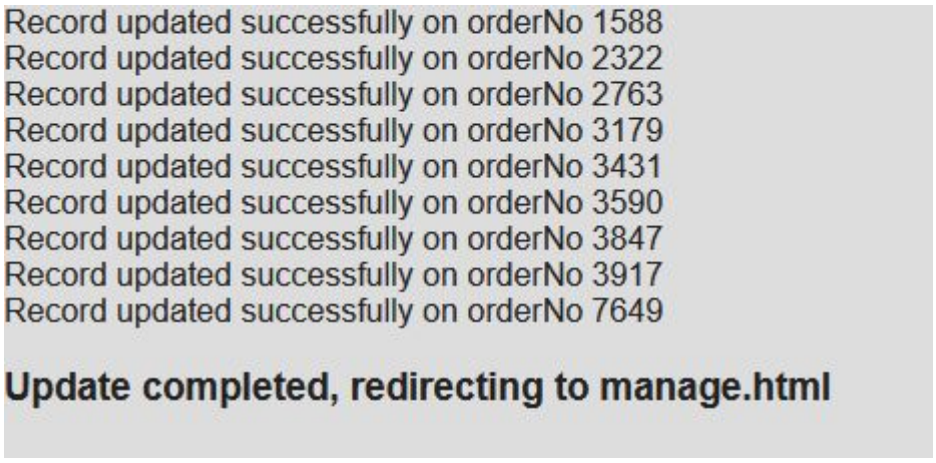
Peak time for people to Order.

The peak hour is (in 24 hour format): 10

The updated records are parsed to processing.php and processing.php will display the successful records and redirect to the manage.php in 3 seconds using javascript. The query processed is:

```
$sql = "UPDATE orders SET Status='". $stats[$i] ."' WHERE
OrderNo=" . $orderNo[$i];
```

the $i will count the number of records and update it accordingly. Each record will display a statement to indicate if it can successfully handle the updates. The bolded statement will remind the admin that he/she will be redirected shortly.The example output is generated as:



```
Record updated successfully on orderNo 1588
Record updated successfully on orderNo 2322
Record updated successfully on orderNo 2763
Record updated successfully on orderNo 3179
Record updated successfully on orderNo 3431
Record updated successfully on orderNo 3590
Record updated successfully on orderNo 3847
Record updated successfully on orderNo 3917
Record updated successfully on orderNo 7649

Update completed, redirecting to manage.html
```

While the function in javascript is:

```
        //After 3 seconds, the website will redirect to List.html.
        function timeout()
        {
            setTimeout(function(){ window.location.href='manage.html'; },
3000);
        }
```

4. Technologies and applications

- Valid semantic markup(HTML5)

- Valid CSS

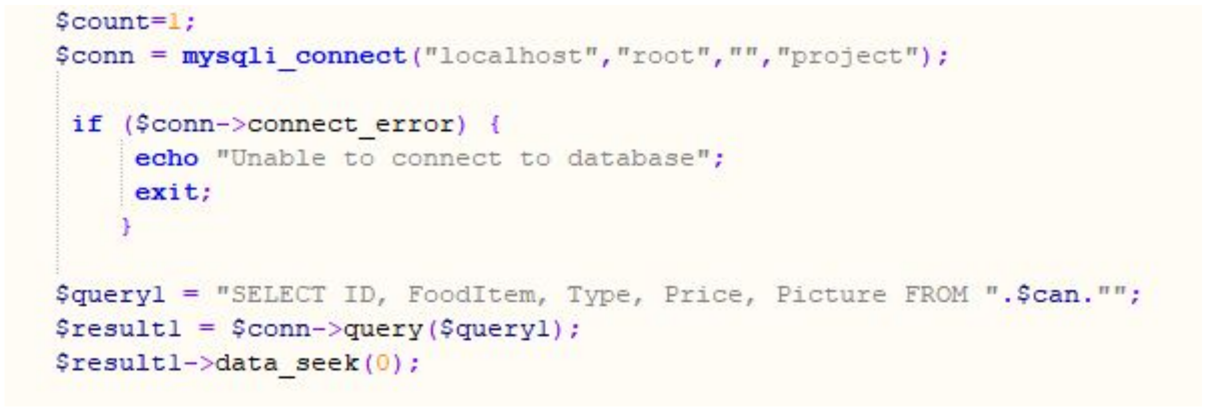
- Client-side scripting(JavaScript)
  - For checking blank fields in forms

- Server-side scripting(php)
  - SQL query and data processing

- Database connectivity

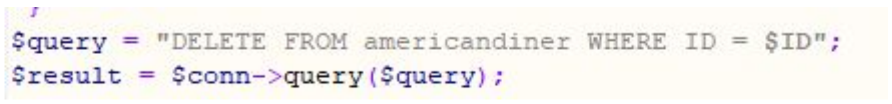
a.The database of menu

For the Admin side, they need to know the current menu list. Therefore, the php needs to connect with database Food table to get data. After querying the database, it will show the menu.

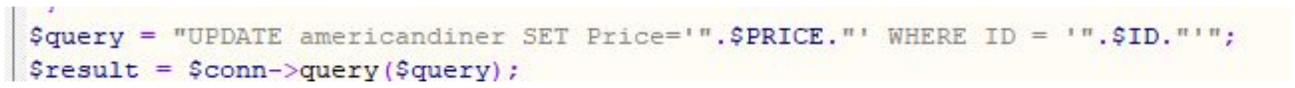


```
$count=1;
$conn = mysqli_connect("localhost","root","","project");

 if ($conn->connect_error) {
     echo "Unable to connect to database";
     exit;
    }

$query1 = "SELECT ID, FoodItem, Type, Price, Picture FROM ".$can."";
$result1 = $conn->query($query1);
$result1->data_seek(0);
```

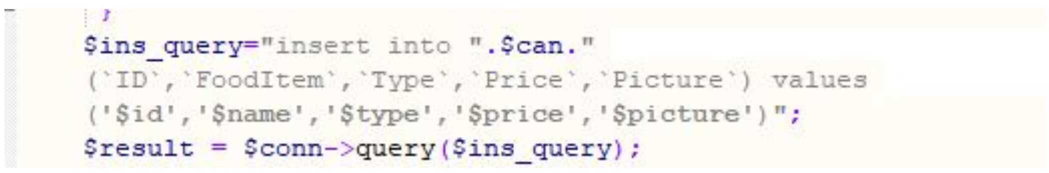
When admins need to update the menu, they need to change the database. They can delete the exit items or update the current items. Besides, they can just add a new item with input information.



```
$query = "DELETE FROM americandiner WHERE ID = $ID";
$result = $conn->query($query);
```



```
$query = "UPDATE americandiner SET Price='".$PRICE."' WHERE ID = '".$ID."'";
$result = $conn->query($query);
```



```
    $ins_query="insert into ".$can."
    (`ID`,`FoodItem`,`Type`,`Price`,`Picture`) values
    ('$id','$name','$type','$price','$picture')";
    $result = $conn->query($ins_query);
```

b. The database of

- AJAX

5. Group Contributions

Members	Code	Report
LIU Cheng Kai	- confirmScreen.php - checkout.php - project.sql (tables: Orders, UsedID)	- Introduction - Functionalities (ConfirmScreen.php + Checkout.php) - Data engineering (ideas) - Future Development (improvements on ConfirmScreen.php & Checkout.php)

[redacted]

6. Data Engineering

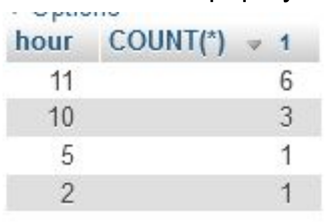
During the development of PolyU-Eat, we came up with several data engineering ideas.
For example, by analyzing the order time and the user-specified time for picking up food, we may get a pattern of when will users usually order food and when will they usually pick their food up. This way, the manager of the canteen can better know when will be the peak hours for the canteen and can be better prepared (i.e. assign more employees to work within peak hours or prepare more ingredients).
Another idea is to analyze the users' orders and see what kind of meal is most people's preference or what meal is being the least ordered. One of the applications of this idea is to include a section in the menu screen called "Today's pick," and randomly put one of the most ordered items there as a recommendation to the user, or even put the least ordered item to prompt the user to try it out.
One more idea is to implement data mining. By data mining the users' orders, we may find the users' patterns of commonly ordered items. For example, we may find that surprisingly, orange juice is usually ordered with hot dogs. By using the acquired pattern, the staff can consider to make combo meals based on those patterns to improve sales.
As for the admin side, we have collected all the orders and sort them by time. We display the hour that has the highest counts. This way, the chef and the manager can easily manage stocking and prepare cooked dishes beforehand.

One of the data engineering ideas is implemented in our project.
OrderList.php
In OrderList.php, a SQL query is perform to check the peak hours of orders. The SQL:

```
SELECT HOUR(OrderTime) as `hour` , COUNT(*) FROM `orders` GROUP BY
HOUR(OrderTime) order BY COUNT(*) DESC
```

retrieves the peak hours of ordering. This helps the administrators to prepare beforehand. The peak hours will be subject to change according to the increasing orders.

In the database phpmyadmin, we retrieve the following:



hour	COUNT(*)
11	6
10	3
5	1
2	1

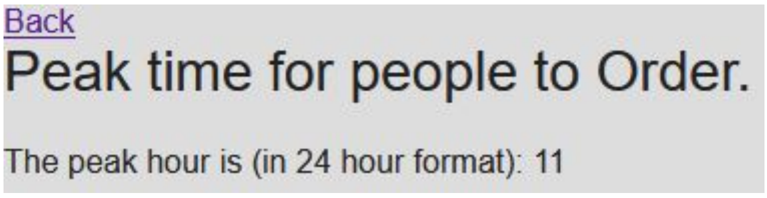
We want to call the peak hours (which is 11). An array could be created to store all the possible hours, and count which one has the highest frequency. The highest count is the peak hour.

To call the peak hours, we store them in an array and call the array's first element.

```
while($row2 = $result2->fetch_assoc()) {


            $hour[$i] = $row2["hour"];
            $i++;
            }
```

```
<p> The peak hour is (in 24 hour format): <?php echo $hour[0];?>
```

The generated output is:



Back

Peak time for people to Order.

The peak hour is (in 24 hour format): 11

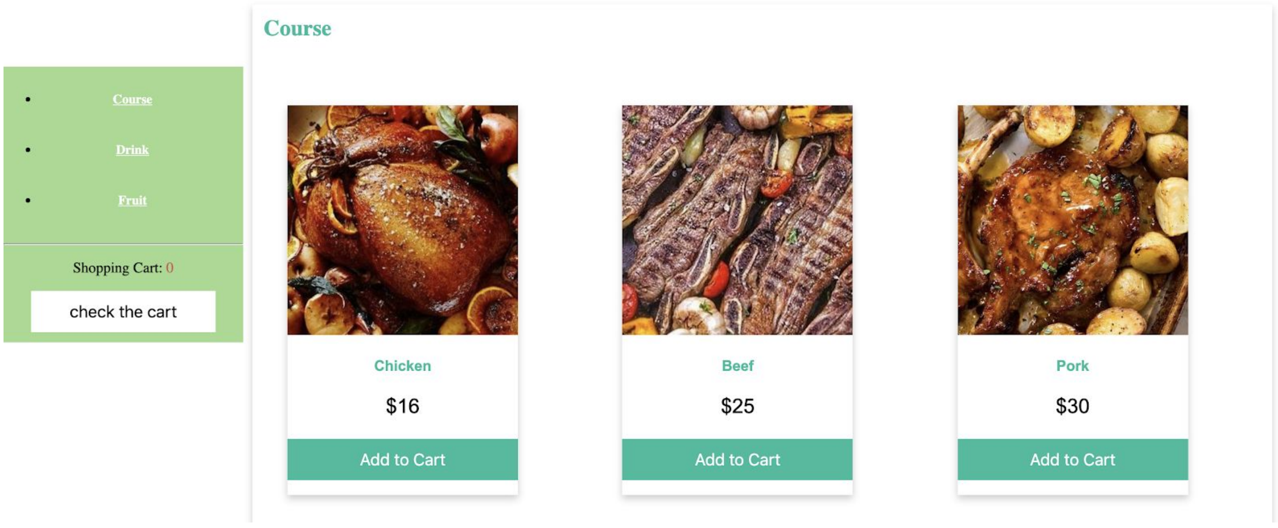
Another implementation is in Menu.php and confirmScreen.php.
In confirmScreen.php, every time an order is placed, the popularities of the ordered food in the database will be updated.

```
#update the popularity of food
for ($i = 0; $i < count($foodArr); $i++){
  $updateQuery = "update `" . $canteen . "` set Popularity = Popularity + 1 WHERE FoodItem = '" . $foodArr[$i] . "'";
  $result3 = $conn -> query($updateQuery);
}
```

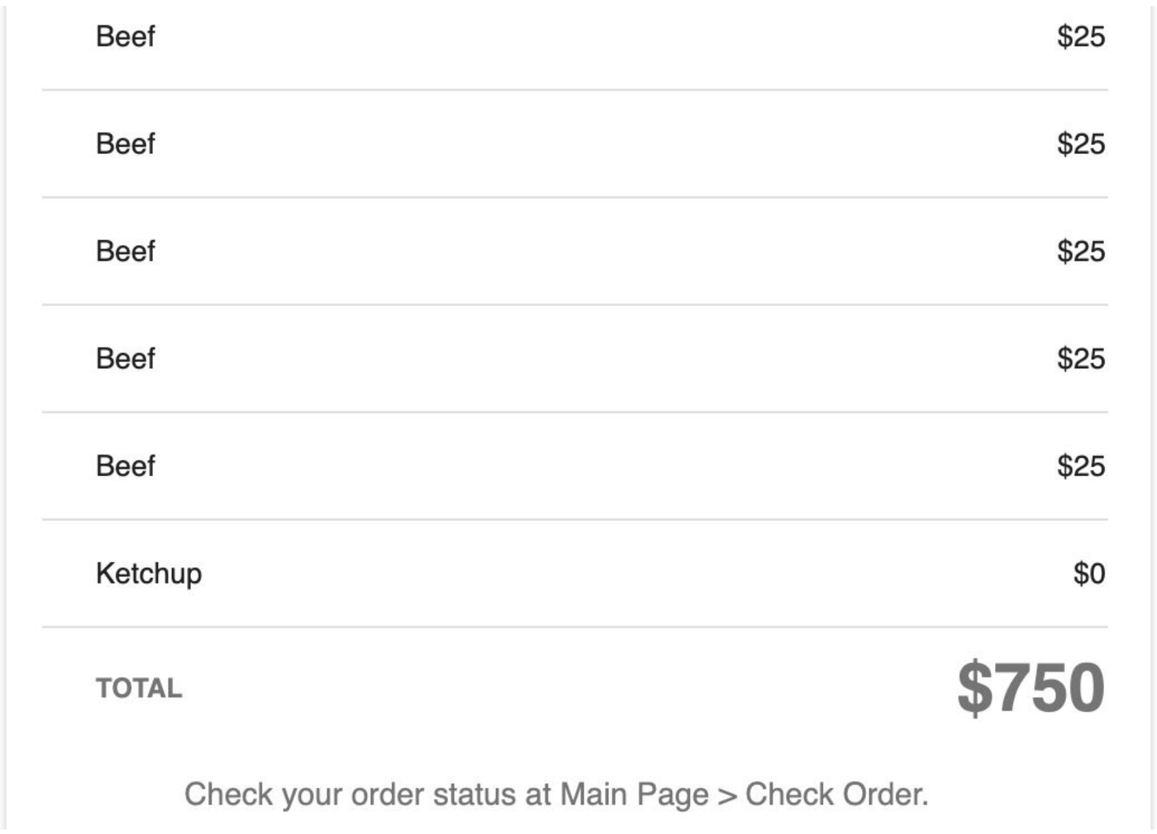
In menu.php, the selected items are sorted by popularity in descending order. The more popular a food is, the easier it is to be discovered by customers.

```
//select query
$query1 = "SELECT id, fooditem as food, type as t, price as p, picture as pic FROM `".$can."` ORDER BY Popularity DESC";
$result = $conn->query($query1);
```

For example, in the original page, chicken is before beef.



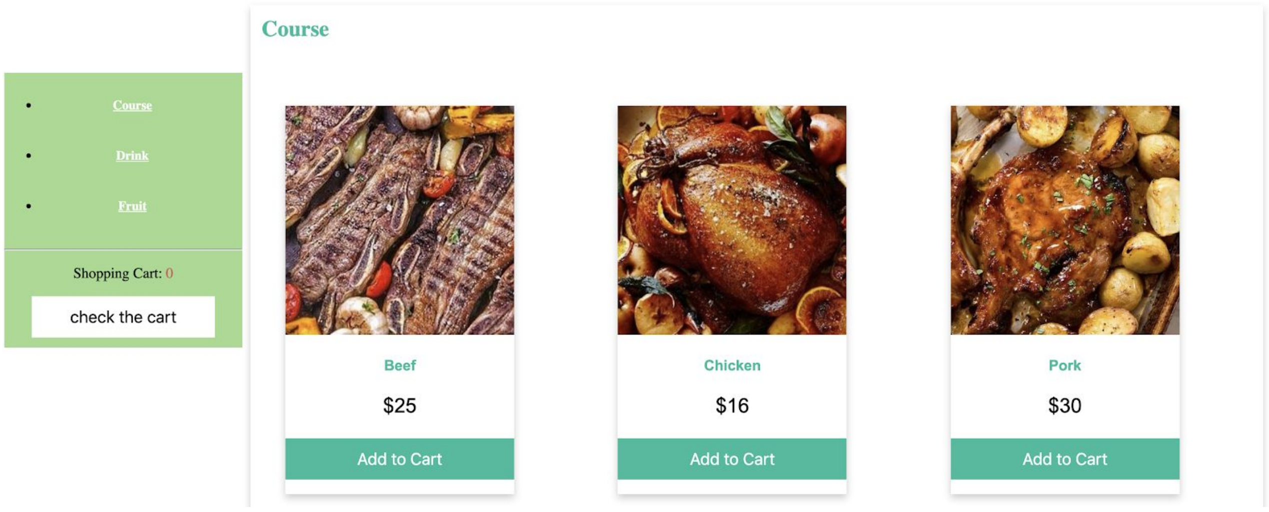
We add 20 beef items to the shopping cart and place the order.



Beef	$25
Beef	$25
Beef	$25
Beef	$25
Beef	$25
Ketchup	$0
TOTAL	$750

Check your order status at Main Page > Check Order.

Now in the new page, we can see beef is in the first place.



Course

- Course
- Drink
- Fruit

Shopping Cart: 0

check the cart

Beef
$25
Add to Cart

Chicken
$16
Add to Cart

Pork
$30
Add to Cart

7. Future Development

Generally speaking, the codes of LogIn and Register parts are still simple. There is no any security about accounts of users. More secure elements should be considered in the future development. For example, we can set weak, middle, strong ranks for passwords. Besides, there should be more CSS part in the log-in page to make it more pleasant.

In database design, we can add relationship by referencing Customer’s email to the Order table, such that the customer can enter his/her email to trace all order made by him/her. Moreover, we can create a table called “Canteen” to record canteen names, foodItems, pictures and more.

The checkout.php page does generates a virtual receipt, but if the user closes the browser, he or she needs to open our website to view the order again (which is currently not implemented). In the future we can add a function to let the user save a picture of the receipt so the user can access it offline.

For now, the number of canteens and food we included is very small. We need to add more options for users.

The menu.php now doesn’t offer an option for customers to log out or modify the shopping cart. A new page should be added for modifying the shopping cart.

The database engineering can be enhanced. We should make it more customized. Each customer should have their own table in the database to store their preferences.

8. Conclusion

Through the joint efforts of the group, the online canteen ordering system can be easily operated. Through our in-depth study of CSS, HTML, JS, PHP, AJAX, MySQL and other technologies, this system covers all the above technologies. Users can order food through the system and track the status of orders in real-time. The restaurant manager can also use this system to view orders from the background and update the menu items.
By this cooperation, we can also better understand the meaning of cooperation. We just started to divide the system into multiple parts, and after each person completed his own part, he integrated all our files together. The most important step is passing the data. Therefore, for a cooperative program project, everyone needs constant communication, and everyone can understand every step of the system.
The system structure is relatively simple, and there are many improvements that can be made in the future. For example, the user's favourite dishes can be found through the user's previous order, and the user's favourite dishes can be placed at the front of the menu directly.

9. References

Material Design Lite - https://getmdl.io/