**Final Report**

**for**

**ThinkFood**

**By**

**ICS Group 06**

Contents

[1. Project Overview 3](#_Toc5629739)

[2. Project Specifications 3](#_Toc5629740)

[2.1 Function Requirements 3](#_Toc5629741)

[2.2 Non-Function Requirements 4](#_Toc5629742)

[2.3 Group Roles 4](#_Toc5629743)

[3. Usability Guide 5](#_Toc5629744)

[4. Problems and Challenges 5](#_Toc5629745)

[5. Skill Learned 5](#_Toc5629746)

[6. Proposed Changes 5](#_Toc5629747)

[7. Code Snippets 5](#_Toc5629748)

**Table of Figures**

[Figure 1 - UML Use Case Diagram 3](#_Toc513207004)

[Figure 2 - Data Model Diagram 4](#_Toc513207005)

[Figure 3 - Cart 5](#_Toc513207006)

[Figure 4 - Login Screen 5](#_Toc513207007)

## Project Overview

ThinkFood is a web application where customer can order foods online. Instead of googling each restaurant

to order foods, with ThinkFood, it lets you easily find and order foods in one website with out much hustle.

The feature for the website is to have a menu for each category by foods and restaurant. Make purchases,

view order history and much more. For the administrator, they can add and remove menu from the

website, and as for restaurant owner they receive orders from the administrator.

The objective is to design a website where customers can add, remove, rate, comment, make purchases, and view

history of their orders. As for the Owner users, they can add and remove product, view the order of their customers,

and deliver the food to the customer. For the Restaurant users they can view the orders they received. We will

implement a database and have it all integrate into the website.

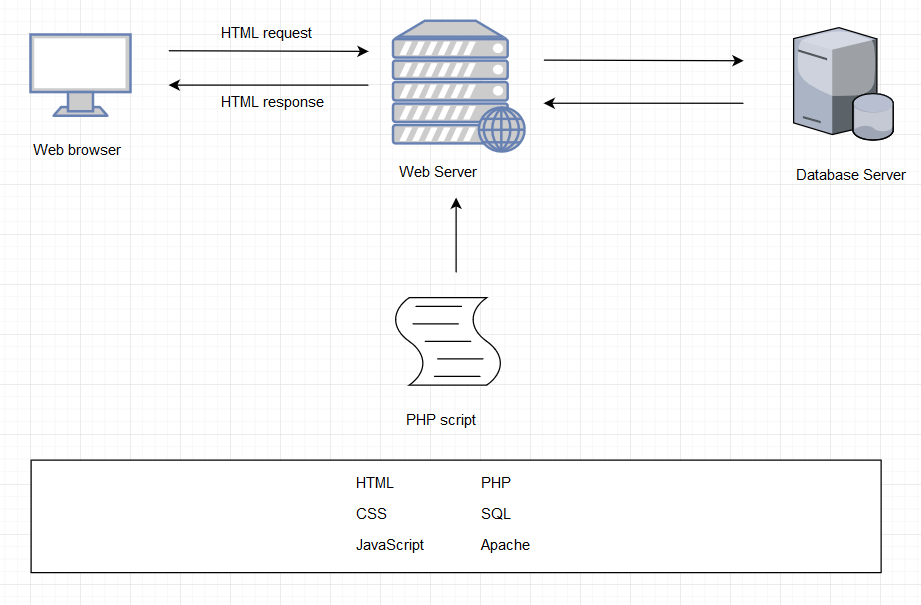


Figure 1 – System Architecture

## Project Specifications

### 2.1 Function Requirements

* R01 – The client must be able to log in to an admin account
* R02 – The client must be able to add products to the web site
* R03 – Must be able to display products by categories
* R04 - Customers must register to place orders
* Must provide a fully functioning shopping cart utility where a customer can:
  + R05 – display the current items in the cart
  + R06 - add selected products to the cart
  + R07 - delete products (individually or all) from their cart
  + R08 - adjust the quantity of a selected product currently in the cart
  + R09 - preserve the contents of the cart if the user’s session is disconnected
  + R10 - have the cart accessible regardless from any machine the customer is logging in
  + R11 - Create a unique file for each order that contains the order confirmation information
  + R12 - include a secure, online payment process
* R13 - Allow customers to view their order history

Extra:

* R20 – Allow customers to provide reviews for any of the products they have purchased
* R21 – Must be able to display the evaluation reviews for any products
* R22 – Provide customers with appropriate product recommendations when they log in
* R23 – Provide customers with any product recommendations when they look at a particular product
* R24 – Ask customers to agree with the term condition on their first log in

### 2.2 Non-Function Requirements

* + R12 - include a secure, online payment process
  + R15 - Use free open source software
  + R16 - Have the system fully tested and up and running in 8 weeks

### 2.3 Group Roles

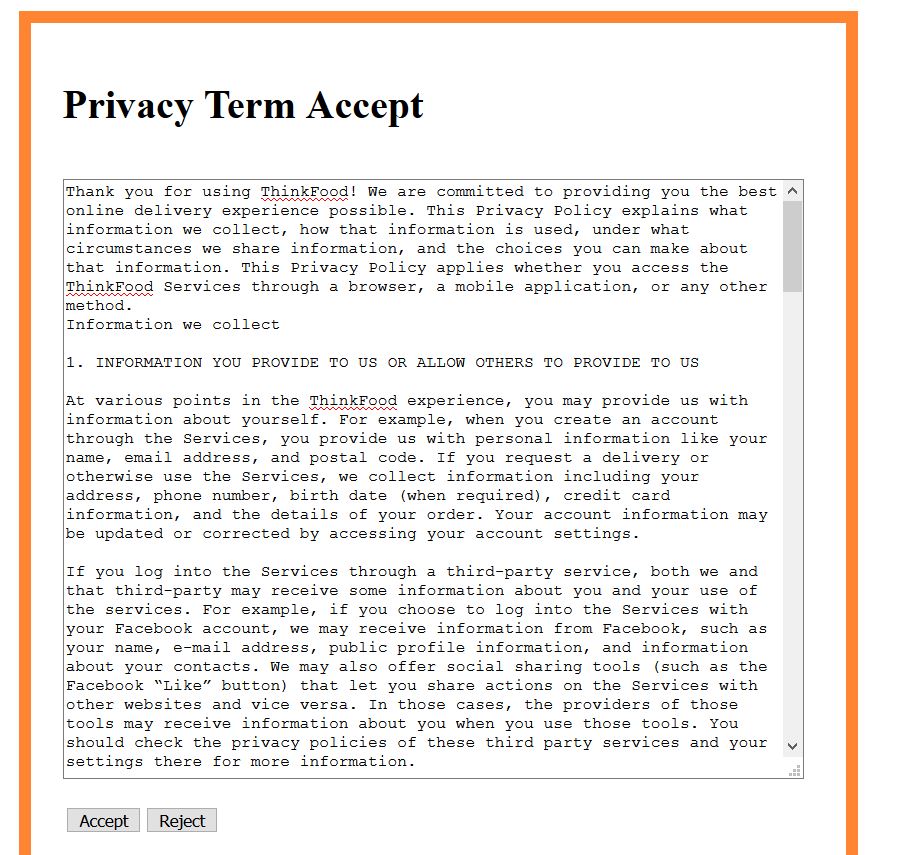
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Week** | **ChangSin Park** | **Qi Li** | **Rolando Pacho** | **Quang Pham** | **Jonghyun Choi** |
| 02 | DBA/Back-End developer | Front-End developer | Middle-ware developer | Front-End developer | Team Lead  Technical Writer |
| 03 | DBA/Back-End developer | Team Lead  Front-End developer | Middle-ware developer | Front-End developer | Technical Writer  DBA/Back-End developer |
| 04 | Front-End developer | DBA/Back-End developer | Team Lead  Front-End developer | Middle-ware developer | Technical Writer  DBA/Back-End developer |
| 05 | Team Lead  Middle-ware developer | DBA/Back-End developer | Technical Writer  Front-End developer | Front-End developer | Middle-ware developer |
| 06 | DBA/Back-End developer | DBA/Back-End developer  Technical Writer | Front-End developer | Team Lead  Front-End developer | Middle-ware developer |
| 07 | Front-End developer | Middle-ware developer | DBA/Back-End developer  Technical Writer | DBA/Back-End developer | Team Lead  Front-End developer |
| 08 | Middle-ware developer | Front-End developer | Front-End developer | Team Lead | Technical Writer |

## Usability Guide

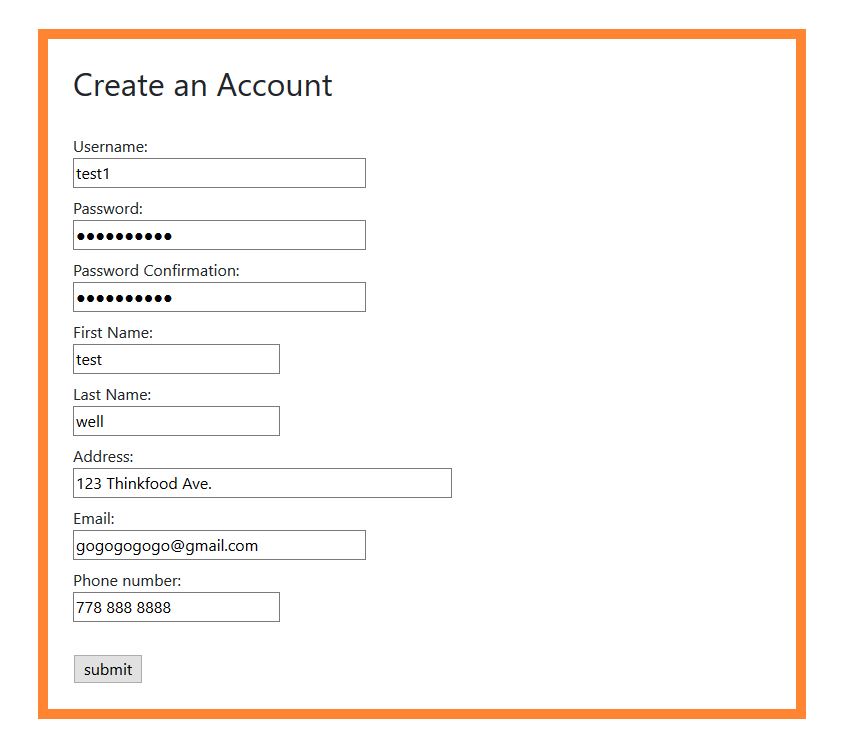
**3.1 Customer User Manual**

**3.1.1 Registration**

Step 1. Go to register page, and accept or reject the privacy policy agreement.

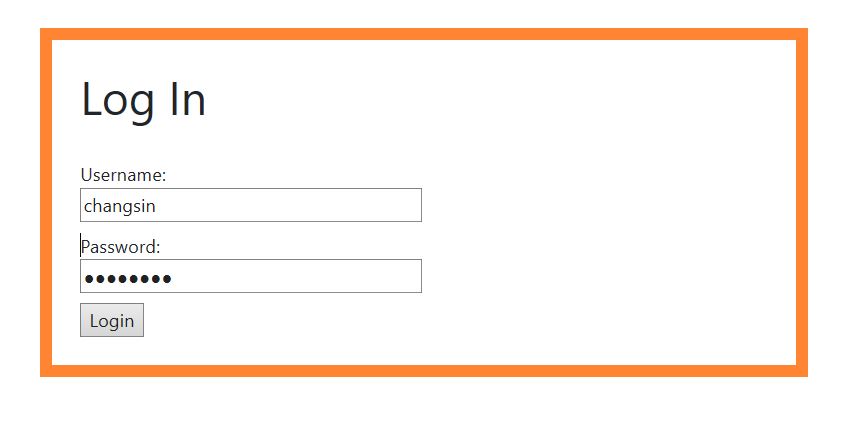


Step 2. fill the form and submit

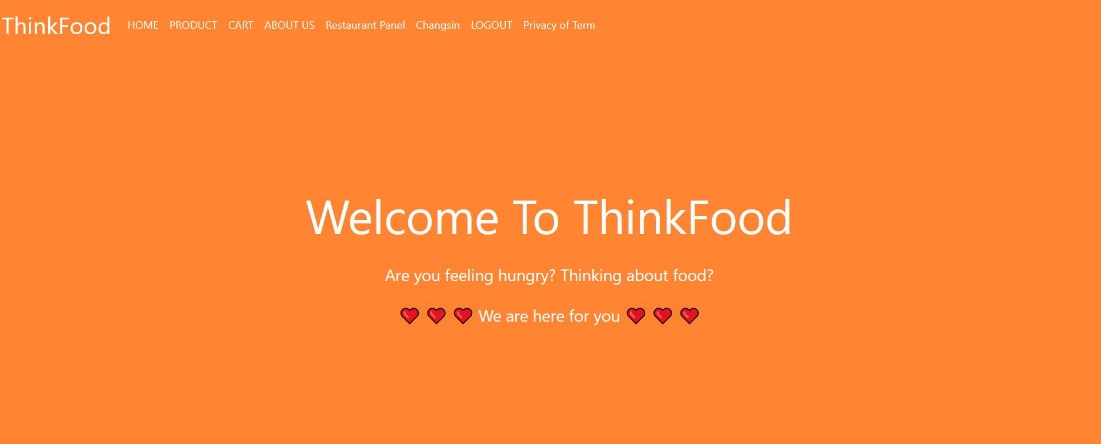


**3.1.2 Login**

Step 1. Go to login page.

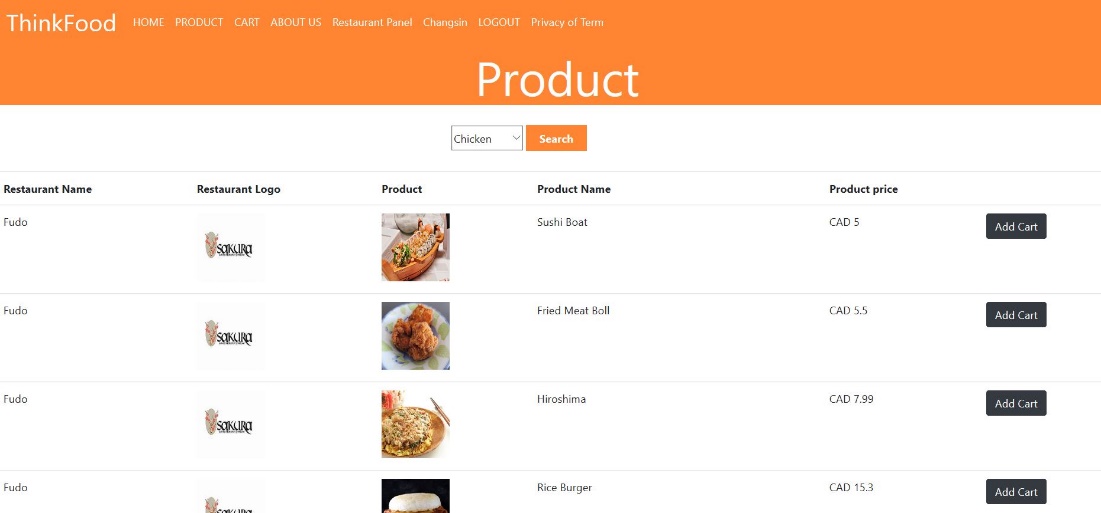


Step 2. Input username and password, and login.

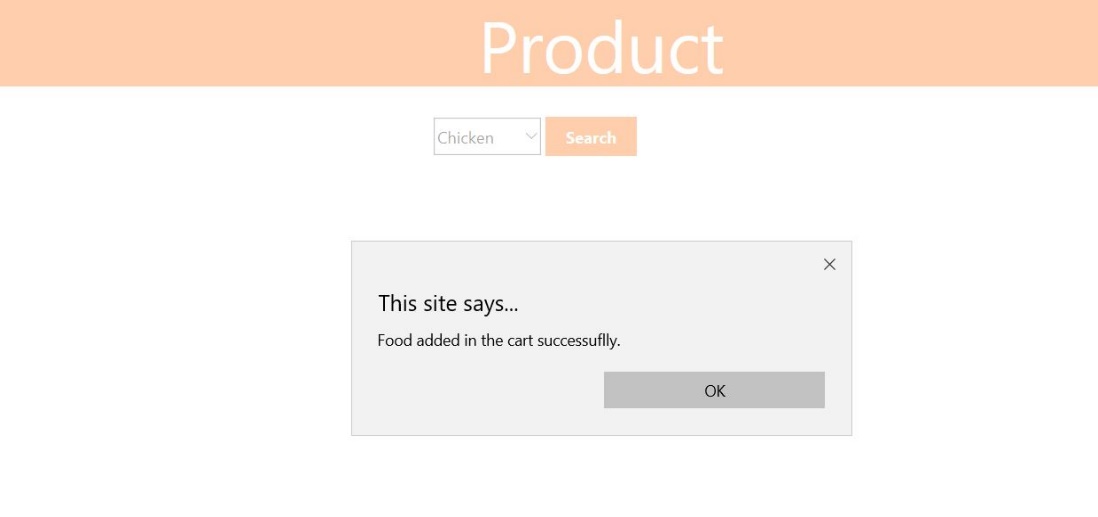


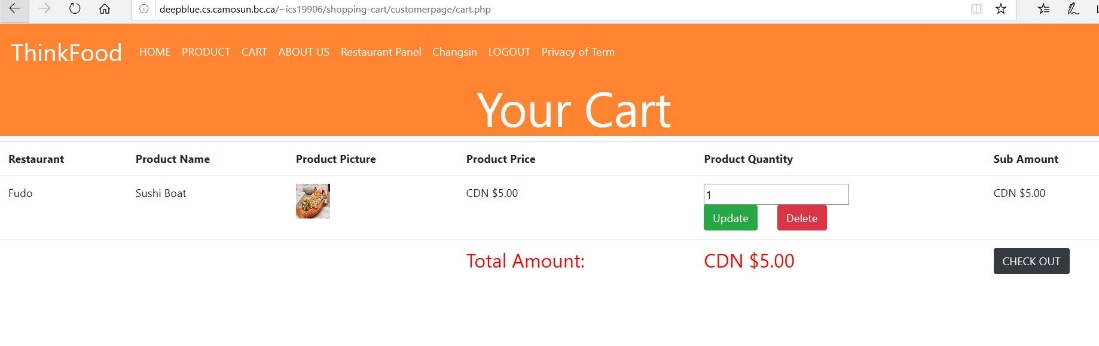
**3.1.3 Product**

Step 1. go to product page



Step 2. add items into cart



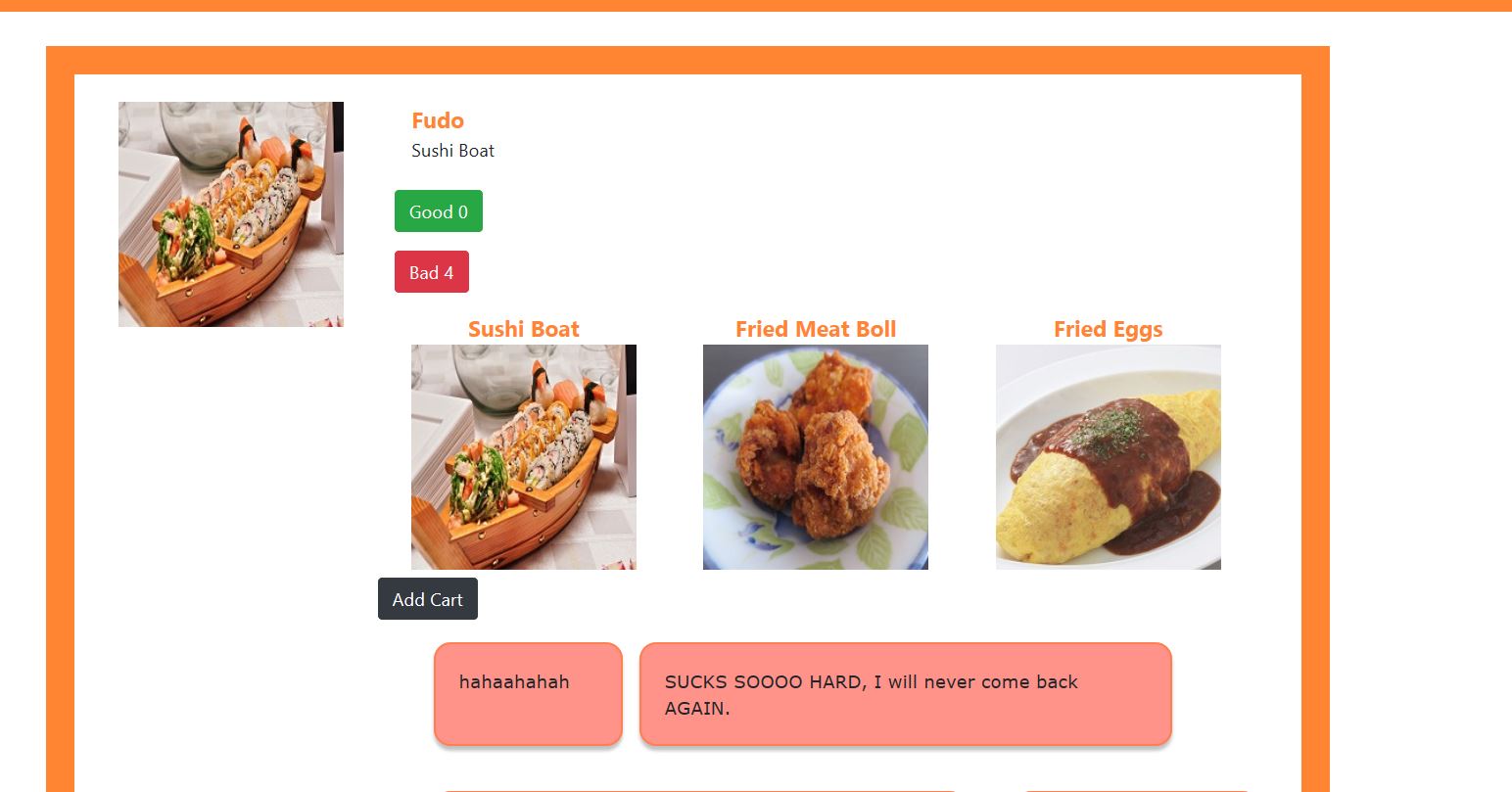


**3.1.4 Recommendation**

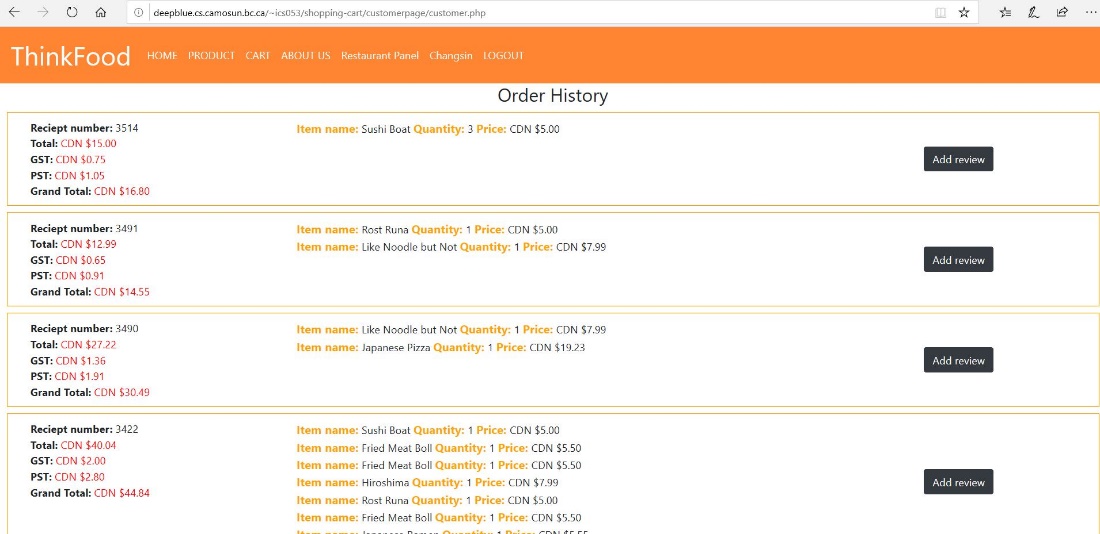
Step 1.

**3.1.5 Review**

Step 1. Click product image to see the review of any product's reputation.



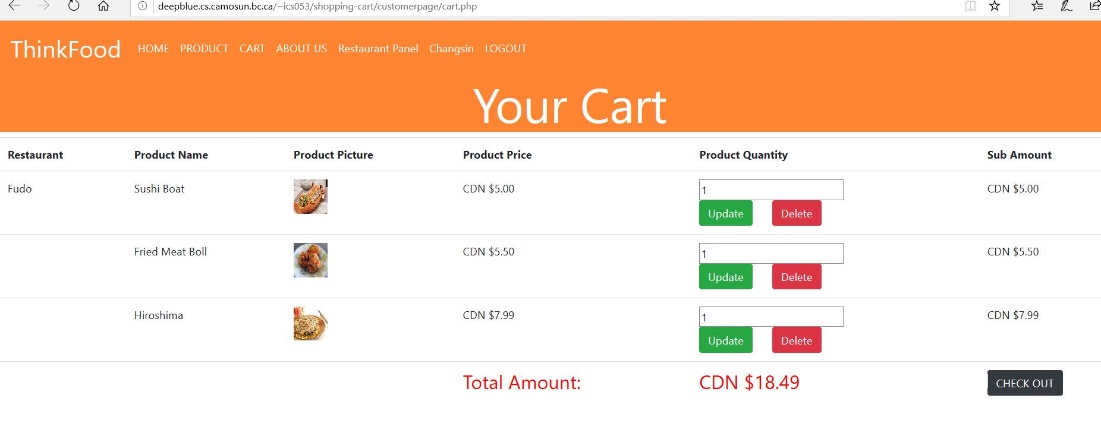
Step 2. After checkout user allowed to go the food what they ordered in their account and rating and reviewing the items.



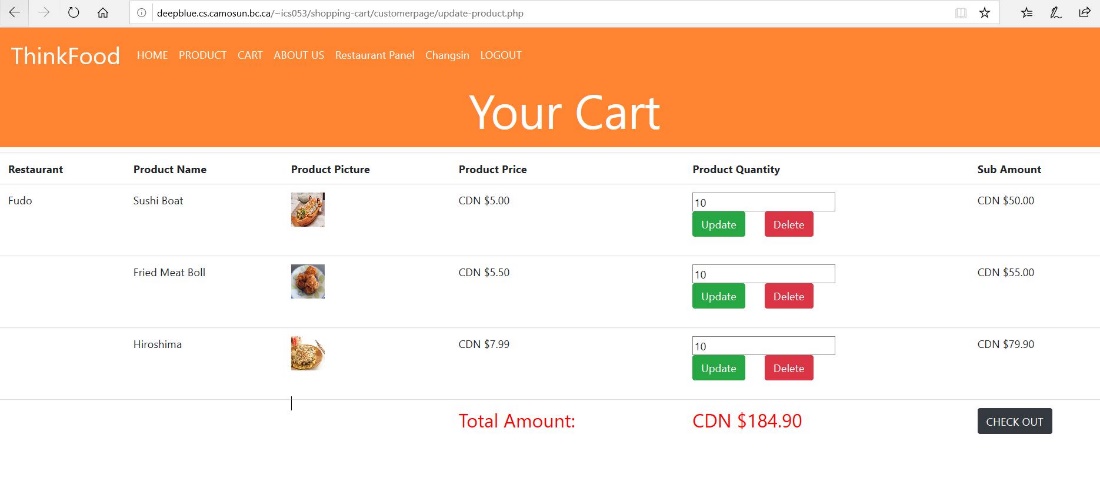


**3.1.6 Cart**

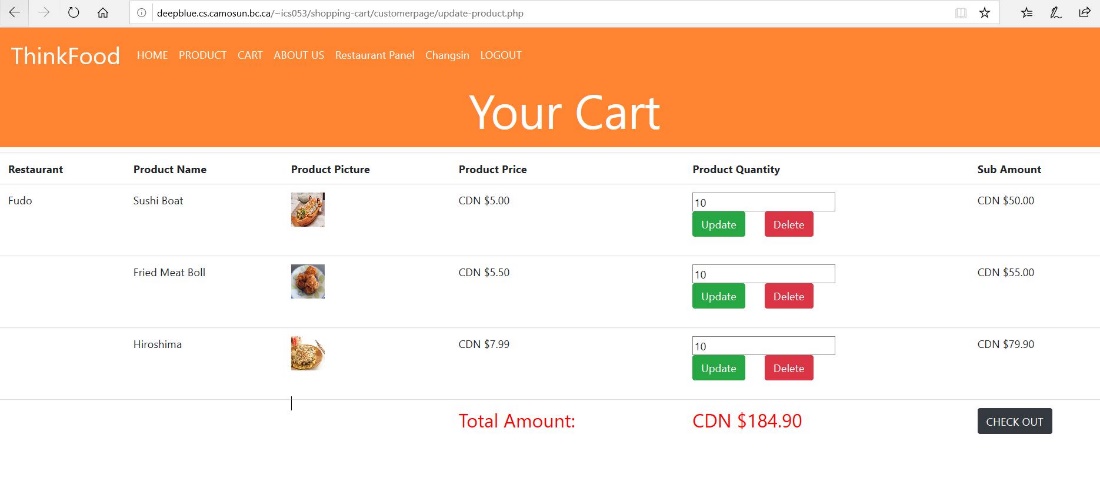
Step 1. go to cart to check the ordered product



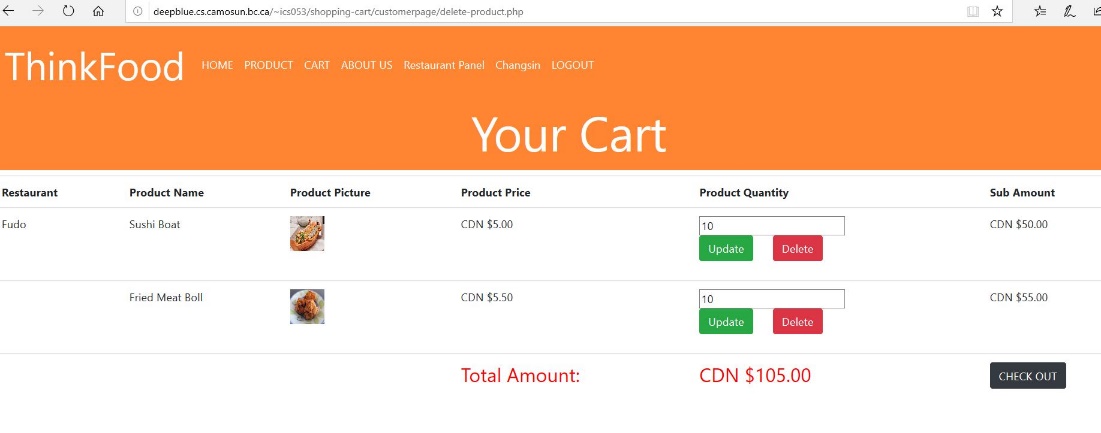
Step 2. change each items' quantity



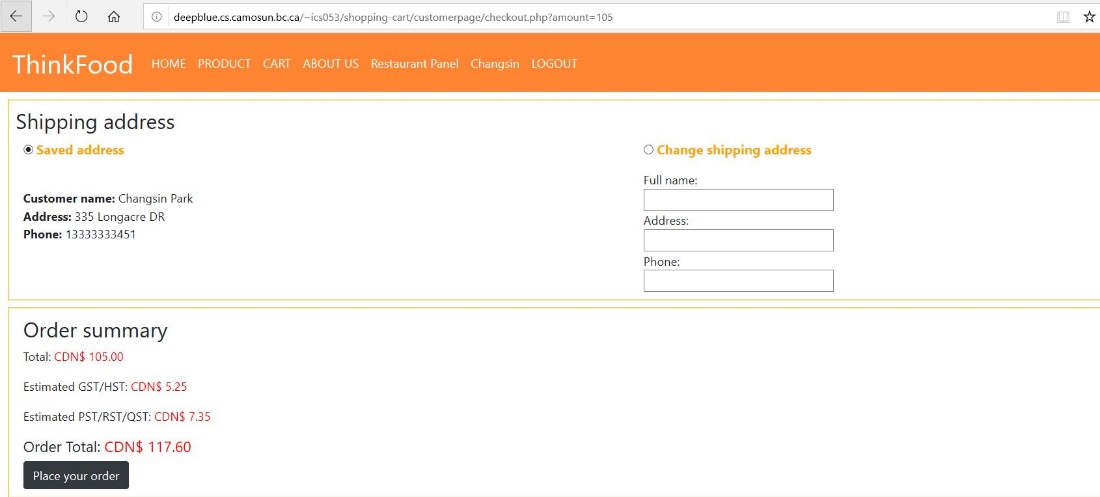
Step 3. check the amount of money



Step 4. delete items

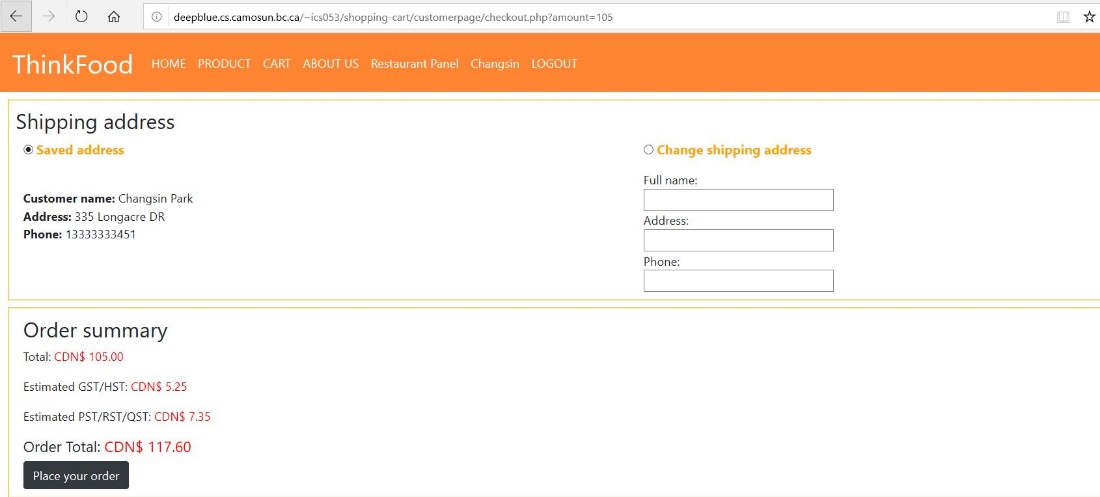


Step 5. press check out

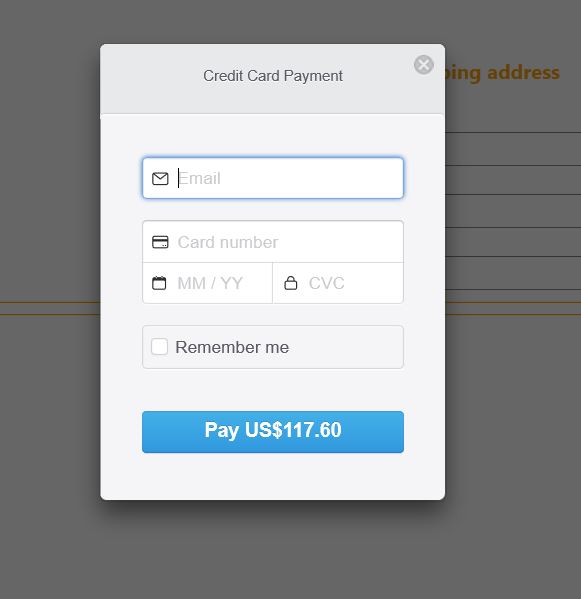


**3.1.7 Checkout**

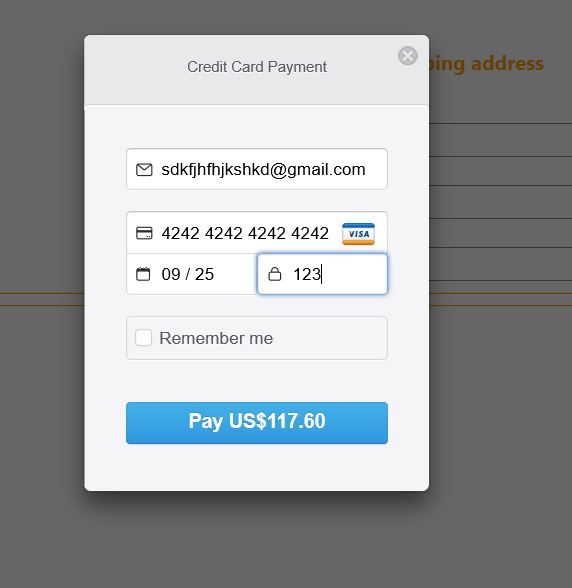
Step 1. in the checkout page users should be able to modify their shipping address and check the info

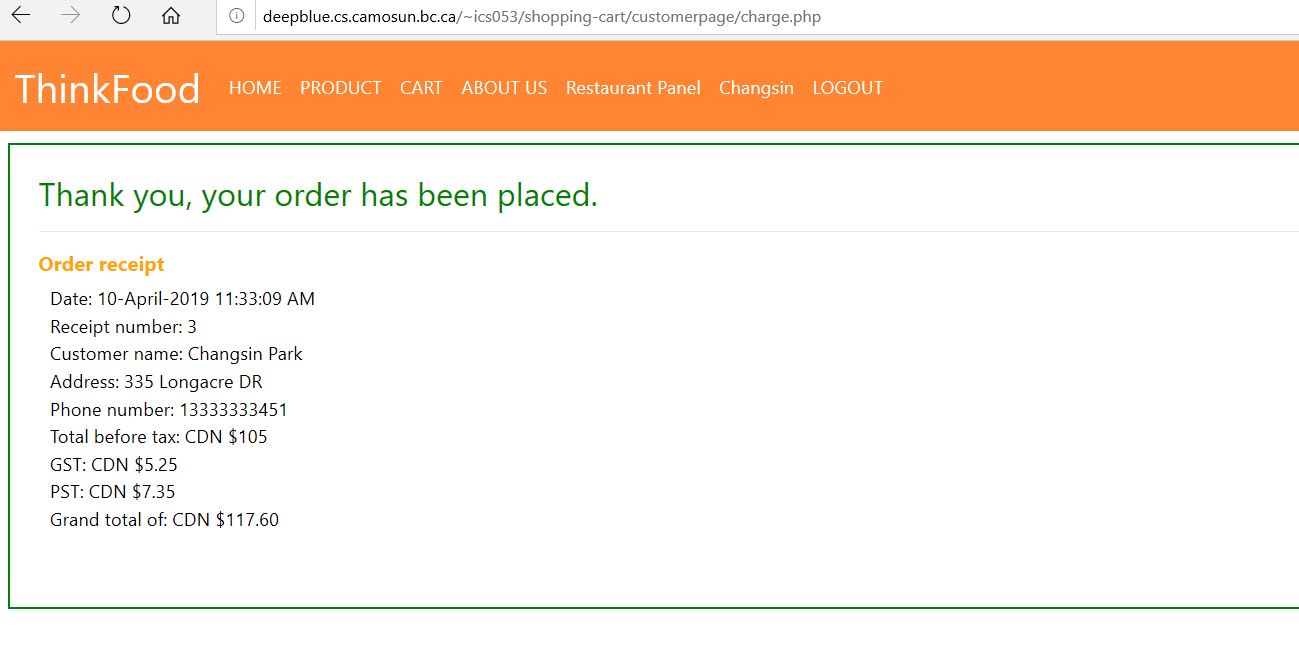


Step 2. click 'place Your Order' button

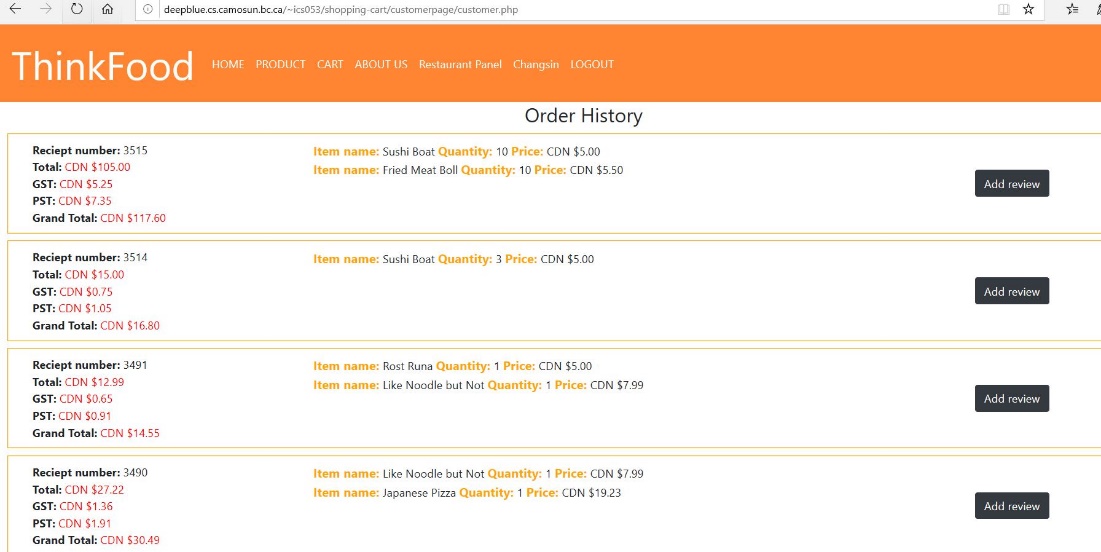


Step 3. put the credit card info into the payment form, and the payment info sent to client's strip account





Step 4. after payment, users can go to their account view session, and see their order history



**3.2 Client User Manual**

**\*assumption: a client has basic privileges as same as a customer, and additionally the client has restaurant privileges as set to limit in one's restaurants and manages all the orders, and logged in as a client account.**

**3.2.1 Restaurant**

Step 1. once a customer orders confirmed, a client is be able to view all the orders

Step 2. by monitoring food status, when the restaurant are ready to deliver their order, it changes status from 'in progress' to 'on the go'

Step 3. after delivery is success, the order status changes to 'completed'

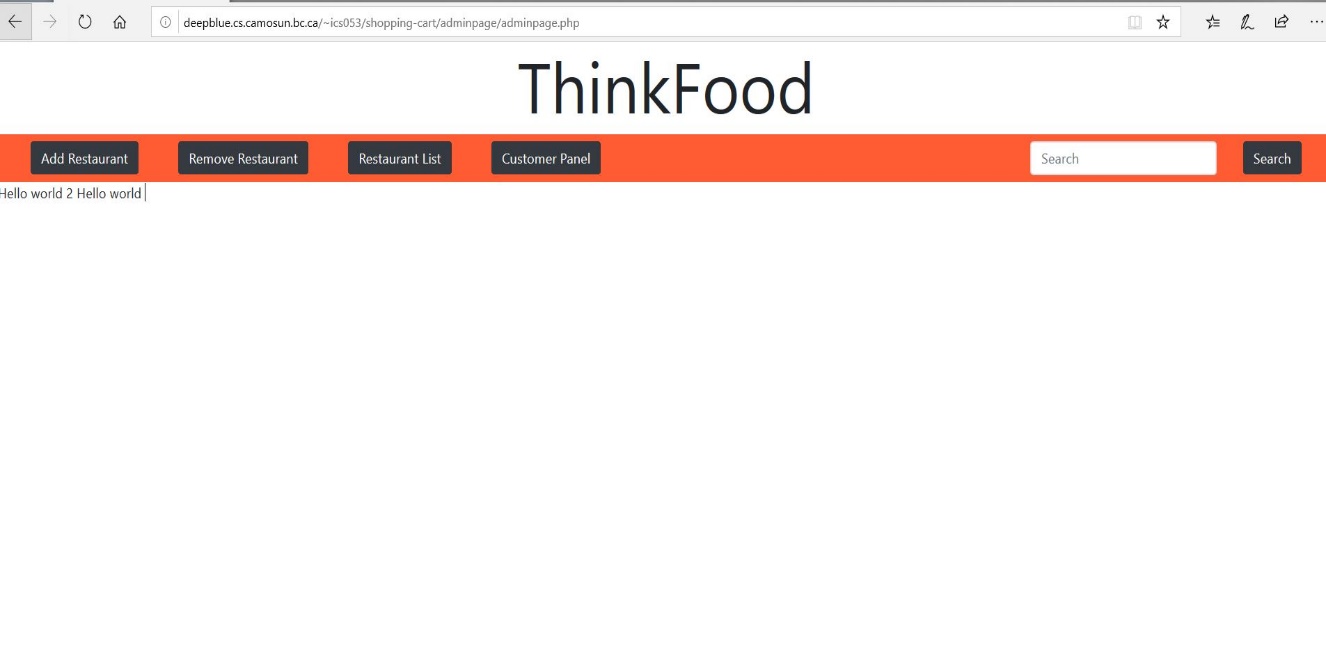
Step 4. when the order delivery completed, restaurant owner should be able to remove the order

**3.3 Administrator User Manual**

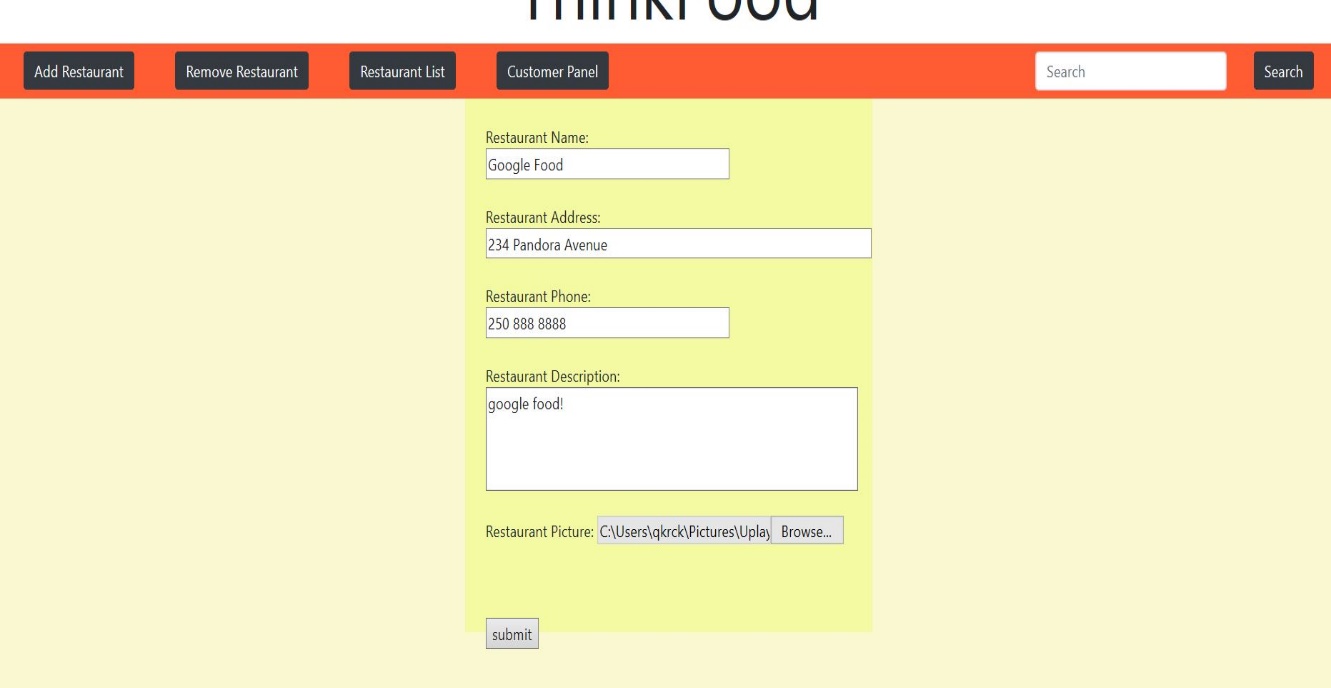
**\*assumption: an administrator has basic privileges as same as a customer, and additionally the administrator has administrator privileges, and logged in as an administrator account.**

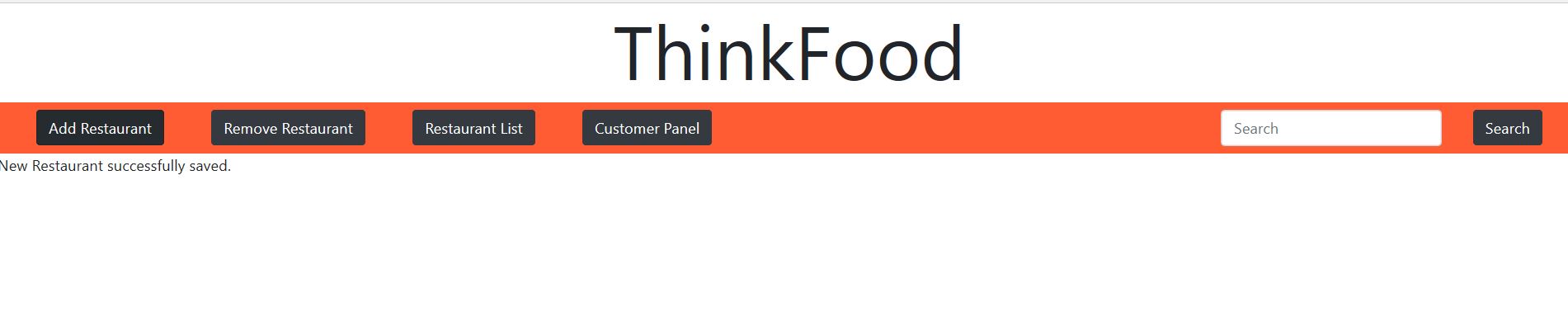
**3.3.1 Administrator**

Step 1. go to admin page

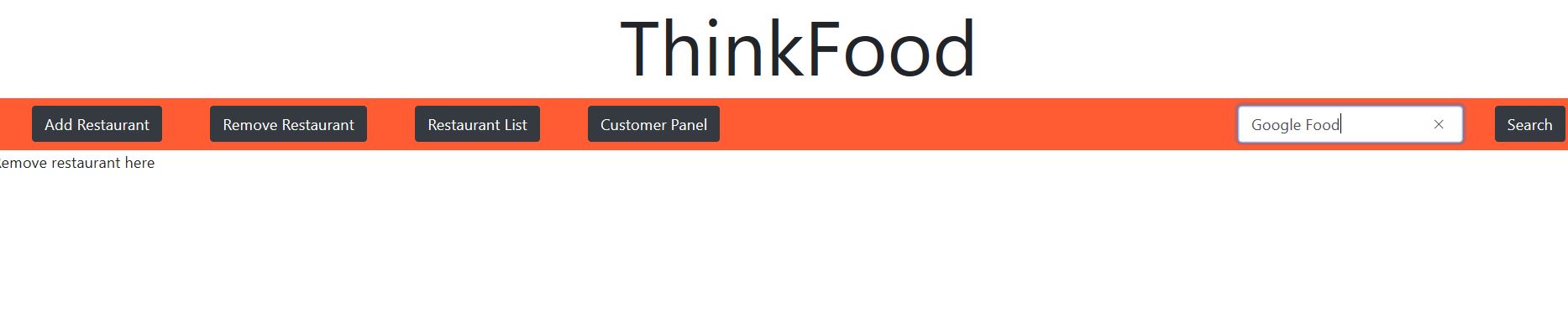


Step 2. add restaurants

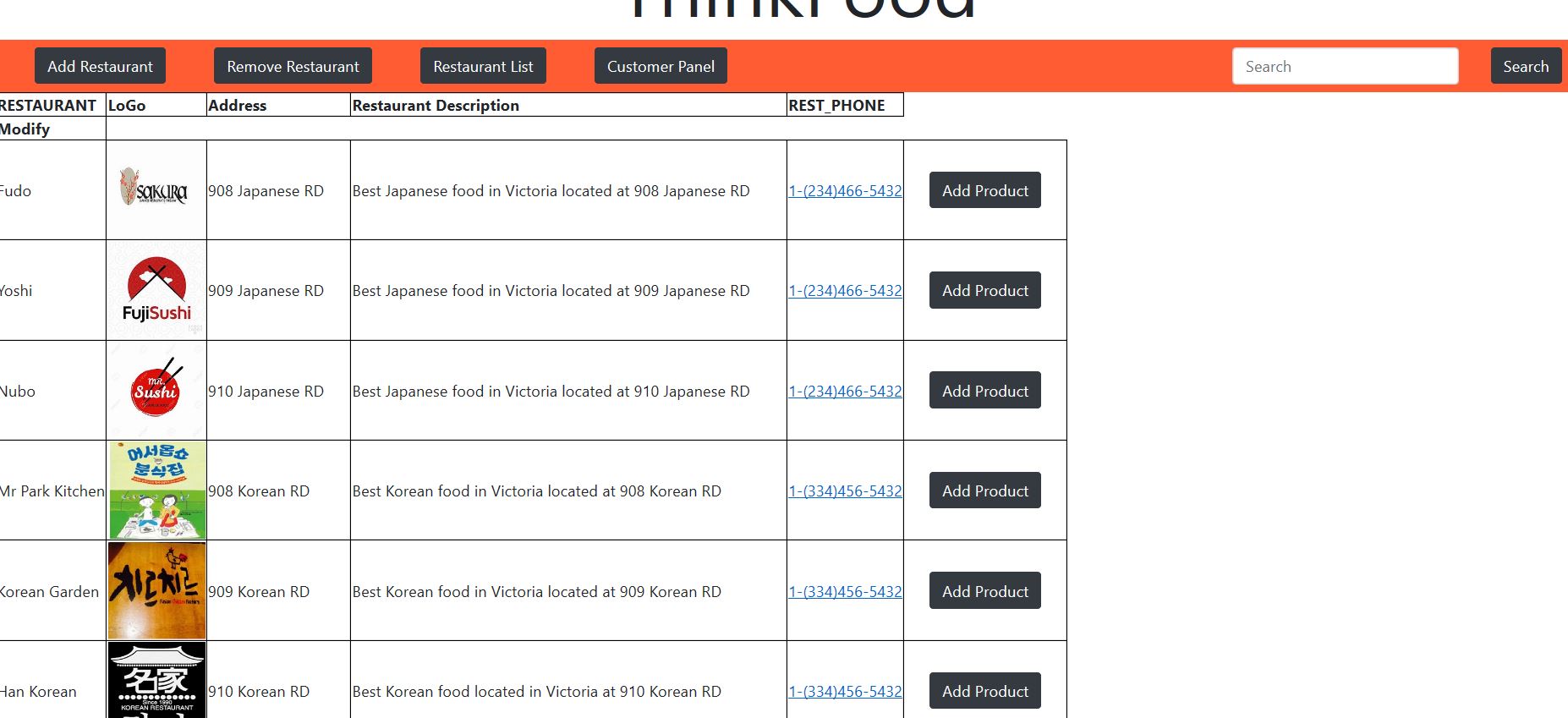




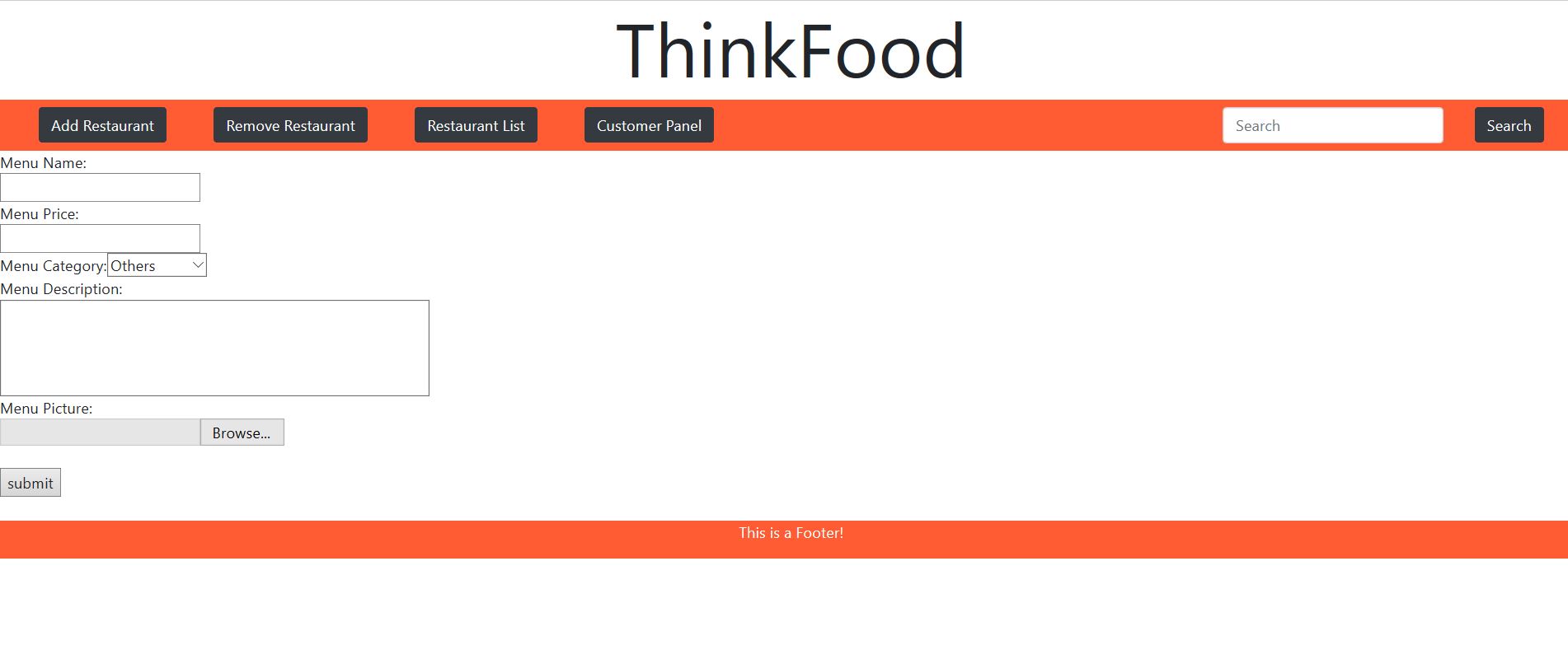
Step 3. search a restuarant from it's already exeists in the restaurant list

Step 4. remove restaurants

Step 5. click restaurant list to view all the restaurants sigined in by add restaurant



Step 6. click add product button in the restaurant list



## Problems and Challenges

### 4.1 Managing tasks on schedule

We faced some problems to manage the schedule for dealyed tasks, missing tasks, new tasks by changement  
 - We handeld it by modifying the priorities of some tasks and changed the schedule.

### 4.2 Standards

We didn’t define naming rules on filename, variable, table & column, so it was hard to understand the others tasks. It was time consuming to us to figure out the sources and documents for deliverables.- We defined the base files for standards and tried to follow the names in those docuemnts.

### 4.3 GitLab

It was hard to be used at managing the resources using GitLab, we lost some code files and faced conflicted files between GitLab maste and deepblue servers.  
- We studied a GitLab and practiced some commands like pull, fetch and push command after making guideline.

### 4.4 Analying requirements

We implemented the sites based on the written requirements but didn’t figure out all of them in detail, so we   
didn’t meet some unexpected checklist on checking by Doug and John.  
- We got the check at the beginning of the class to prepare a missing requirement and immediate changement, so we could make a proper action on time.

## Skill Learned

### 5.1 Cooperation

We had regular meetings to check the tasks’ progress and cooperate. We used Slack for quick communication and set the alarm using it for new changement on our GitLab. We could practice how to work togegher in efficient ways.

### 5.2 GitLab Using a gitlab encourged us to manage our each sources consistent. We used only one way from our GitLab master to deepblue server whenever it come to deploy new versions of sources for version management.

### 5.3 MySQL and Workbench We learned Oracle database and SQL Developer during past database classes. One of the non-functional requirement of this project is using an open source software, so we used MySQL for our data. The syntax is very similar with Oracle, We also used MySQL Workbench for modeling and reverse , forward engineer function for extracting metadata and creating objects each.

### 5.4 PHP, HTML, CSS We could review web technologies such HTML, CSS and PHP through this project. Bootstrap is used for predefined styles for UI. PHP is used for server side implementation between user interface and database. It was a meaningful experience to us to practice all knowledge we learned at 1st year.

## Proposed Changes

### 6.1 Avoid duplicate statements

### 6.1 Avoid duplicate statements

### 6.1 Avoid duplicate statements

### 6.1 Avoid duplicate statements

### 6.1 Avoid duplicate statements

## Code Snippets

*<A couple of blocks of your team’s code showing examples from the back-end, middle-ware and front-end. Explain what it does and why your team is proud of it!>*