Biteman Final Report

CS542 Team 10

Team member: Meng Wang, Yalei Peng, Jianjun Luo, Jiaoyan Chen

Abstract

Our project is aimed at implementing an online food order Application -- Biteman. In our website, customers can view restaurants around them and order the food online. For restaurants, they are able to update the menu and manage the orders. All team members participated in user case and database schema design. We discussed about the user demands and built ER models together. Jianjun Luo and Meng Wang were responsible for implementing ER models in Mysql to create tables. Yalei Peng and Jiaoyan Chen worked on the architecture of our front-end webpage. All of members made their contribution to this project.

Keywords: online food order, database schema design, ER model, MySQL, react, JavaScript

1. INTRODUCTION

With the popularity of online ordering, many people choose to use online food ordering platform instead of waiting in line. It is becoming essential to find an easy-to-use, fully featured order platform that quickly connects diners with the restaurants of their choice. For its restaurant clients, the platform can provide an instant boost in takeout sales while increasing efficiency in processing orders. For the customer clients, it is convenient for diners to place, track, and manage their orders. Online ordering services like GrubHub and Yelp Eat24 have been established, but the war of food takeout is just getting started.

The aim of our project is to implement an online ordering website, by which we can not only learn how to build up a database application, but also have deeper understanding in the food delivery market. Our application allows users to view the available restaurant, customers to order food and track the orders, restaurants to process the orders and edit their menu. To realize our application, we met following objectives:

- Select our application environment and approach.
- Collect requirements and design the database schema.
- Mock and load data into database system and write queries.
- Develop a web-based front end as an interactive user interface.
- Build the server to communicate between databases and websites.

This report presents the background materials in section 2, including two components: detailed features of the application we built and technical skills we used to implement the application. Section 3 is the project process, in which we first introduce our general approach, then show designs of the application (system, database, website, server). Also, we use some screenshots to present an example run to validate our result. Finally, a description of the completion and challenge of the project, lesson we learned and contribution is showed as our own evaluation in section 4, followed by a brief conclusion in section 5.

2. BACKGROUND

2.1 Intent

We implemented a database application on online food ordering. Biteman is a take-out ordering system for customers to easily enjoy delicious food without going out and marketing platform helping restaurants feed their hungry customers.

We build two systems respectively for restaurants and customers:

Restaurant related features

- Login and logout page: we designed a main page that enables restaurants to login and do future actions. After their work has finished, restaurants can logout the webpage.
- Upload and update the menu: restaurants can upload their menu and they also can update the menu once it is modified anytime.
- Manage orders: restaurants can check the unserved orders, decide whether to accept the order and prepare the food at customer's preferences. Once the order is served, it can be marked as "confirmed".
- Check the status: restaurants will get an overall review of recent orders, comments from customers and total income. They also can check the orders at their own preferences.

Customer related features

- Login and logout: Customers can login in to view the available restaurants, to order take-out, check order status and view their personal history orders.
- Create and cancel the order: Customers can select certain food and place the orders. They can also cancel the undelivered orders.
- Score the restaurant and the food: When the orders are delivered, customers can score the quality of food and service.

- Recommend food for customer: The website can highlight certain food option for customers based on their history orders.

2.2 Technical Parts

We use Mysql to build the database system and node.js to build server. For frond-end, we use react.js to build an interactive user interface.

Mysql is open source and has a lot of online resources. It can handle everything from a megabyte of data to terabytes just fine.

Node.js is used to create and run a wide variety of web applications, and it uses only JavaScript for the server side. The key benefit of using Node.js is to keep the backend and frontend using javascript, which simplify the technology stack. Other web technologies like Spring or ASP.NET requires developers to have knowledge in another language to write code on the server-side, be it VB.NET, Java, or C#. This means all functions necessarily include two languages – one on the client-side and the other on server-side.n the contrary. Node uses only JavaScript for the client and server side. Thus, web developers only need to interact with a single language across all layers.

The reason for using react.js is that it is fast and the framework is quite understandable. Anyone with a basic previous knowledge in programming can easily understand React while Angular and Ember are referred to as 'Domain specific Language', implying that it is difficult to learn them. Another cool feature of React is it is a die hard fan of reusability, meaning extensive code reusability is supported (using components). What's more, it supports server-side rendering, which helps in solving performance and SEO problems.

3. PROJECT

3.1 General Approach

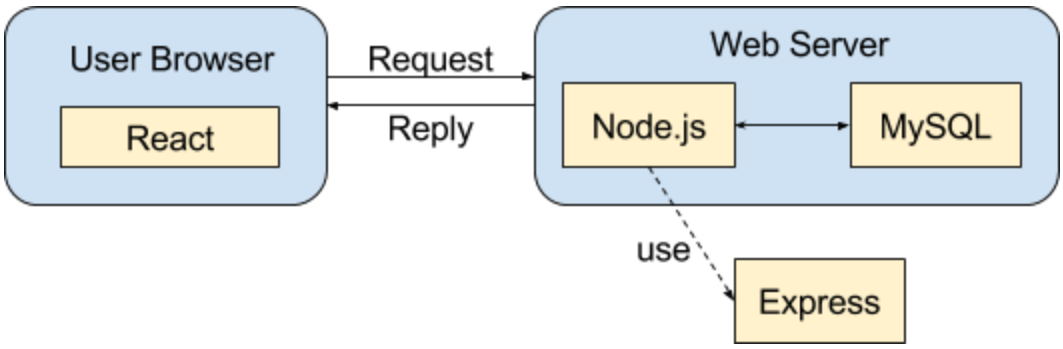
To build our web application, we used Node.js and MySQL as our basic tools. Node.js is a lightweight and efficient programing language for web application, while MySQL is a popular database used for storing data. Also, we can get MySQL database driver for Node.js through NPM repository. Specifically, we used Express, a minimal and flexible Node.js web application framework, to build server. A JavaScript library React, is used to create the interactive user interfaces on the front end. Benefits of these tools have been mentioned in the last section.
We discussed and made demand analysis for the database first. After that, we got down to system design. We split the whole system into independent parts: databases, websites, server. We listed the desired features and design the corresponding ER-model and then transformed it into schema. We obtain and load our data into the database system, and write a number of SQL queries. We develop dynamic websites with React, using Ajax to send message to the server. Then we tried to connect MySQL database using embedded SQL queries to do INSERT, UPDATE, SELECT and DELETE operations on MySQL database table. Finally, we completed the application by improving the user experience and testing.

3.2 System Design

3.2.1 Web application framework

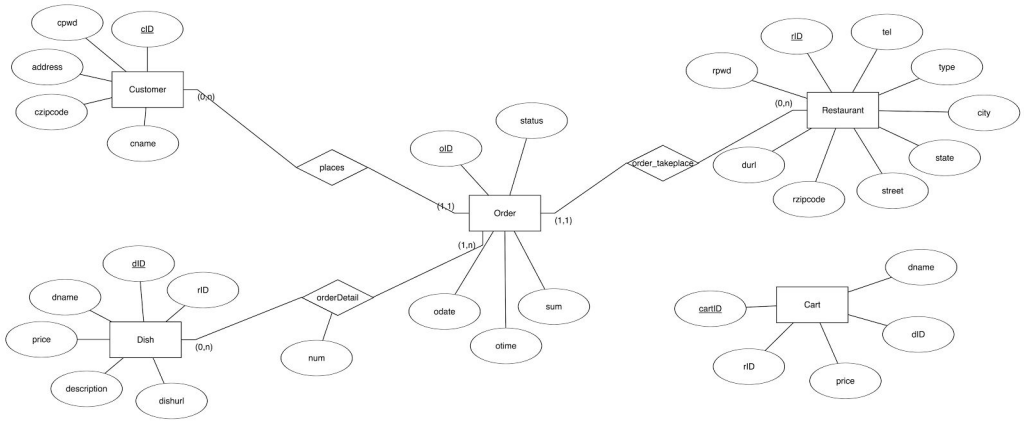
We use react.js to build client-side applications. React is an open-source JavaScript library for building user interfaces which is maintained by Facebook. The main advantage of react is we can develop different features independently and asynchronously. So each member in our team can focus on his/her work and won't be affected by others. On server side, we use Node.js and Express 4.0 to get user requests and send responses back.Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.It allows us to set up middlewares to respond to HTTP Requests. Using

given API in Express, we can connect MySQL and query, update, delete, insert data in our database. The data send back from database will be passed to client and then rendered in web page. This progress is shown below:



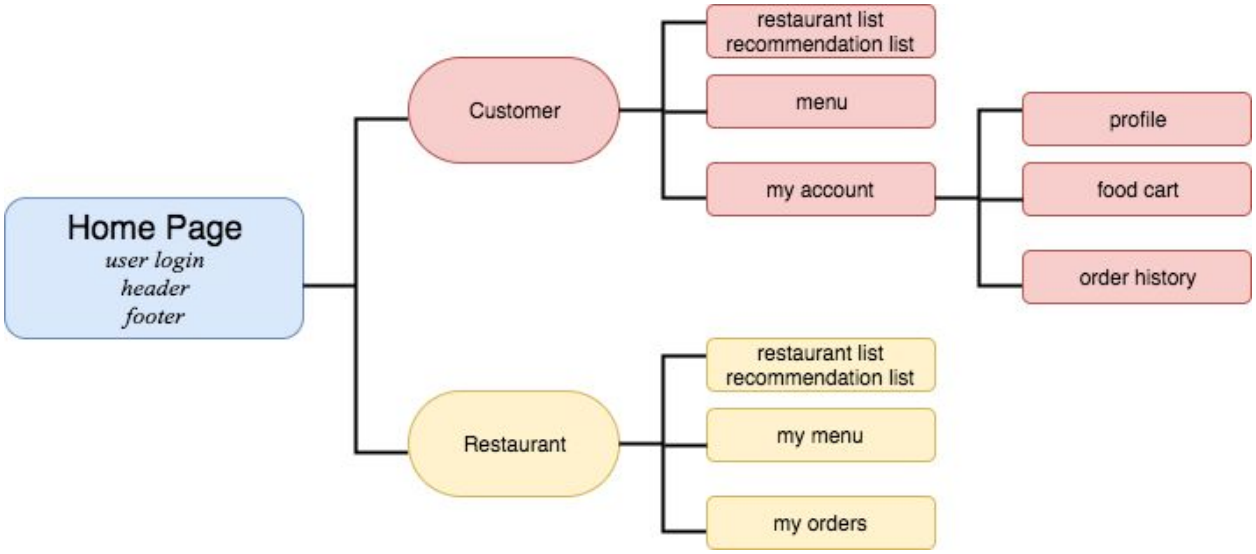
3.2.2 Database

The database schema can be see in appendix. There are five tables in our database. Customer table and restaurant table are used to store our website user information. Dish table stores all kinds of food among restaurants, which can be grouped by restaurant id to return menu for certain restaurant. The core relation of these tables is shown in Order table. Each record in Order table represent an order created by certain customer. It records customer Id, restaurant Id, when it created, the total price of this order and the status. So our website users can either get the orders they want to see by customer Id or restaurant Id. Notice that there is a relatively independent table called Cart. In this table, the temporary order detail will be saved, but customer doesn't pay for his cart so it cannot be saved to Order table.



3.2.3 Website Design

The overview of our client side design is shown below:



Home Page:

We create different components for customer users and restaurant users respectively. The Login page is the same for them. Users should choose which group they belong to and then input their email(which should be unique) and password to log in. User group, user email and password will be send to server. Server then pass down these data to database connection function. What database connection function does is select the user Id(customer Id or restaurant Id) and

password by email from the corresponding table(customer table or restaurant table). Now we can campare the user input password and password sent from database. If two strings are exactly matched, a successful sign will be given as response to client side.

Customer:

After logging in successfully, three components will be shown in client side. On restaurant page, all restaurant in our database will be sent to server and listed on the website. If customer wants to view detail menu in certain restaurant, he can click the button and jump to menu page. When customer chooses one of the restaurants, the restaurant Id is sent to server and all food in Dish table has the same restaurant Id will be sent back. Apart from restaurant list, the top five popular restaurant will be shown in recommendation list.

On restaurant menu page, customer can add whatever food in the restaurant they want to eat. We also allow customer to switch to another restaurant picking food. When customer clicks add button, all information of selected food will be add to the Cart table. Now customer can go to my account component to manager their cart, profile and order status information. The food in Cart table will be shown on cart page grouped by dish Id. Customer can have a overview of the quantity and total price or delete what he doesn't want on this page. Deletion feature can be achieved by sending the dish Id to server. If customer decides to place the order, the rows in cart table where contains current customer Id will be emptied and the food will be grouped by restaurant Id to generate new order rows in Order table.

The order history page mainly shows all the history order details related to current customer.

On server side, all order records where customer Id is the same as current customer will be sent to front-end. Customer user can change the order status according to their needs. Profile page shows the name and address of the current customer. It allows customer to modify his ship address.

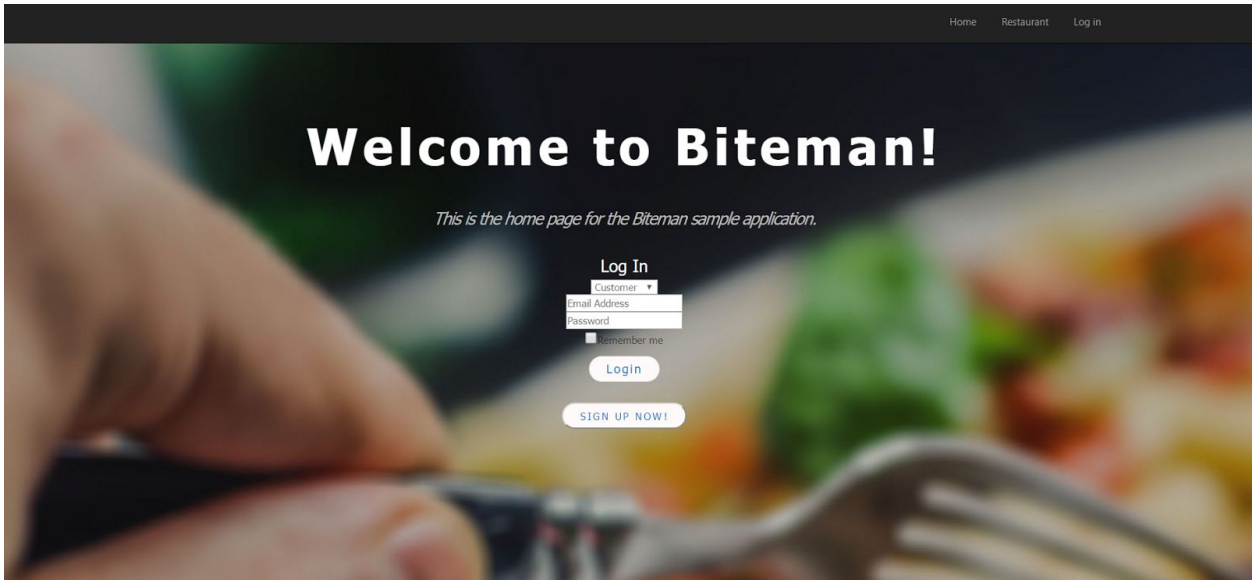
Restaurant:

If our client logs in as retaurant user, the editable menu and customer orders component will be shown. However, we keep the restaurant list and recommendation for restaurant user to refer. On my menu page, restaurant can update food detail, delete food row and add new food. The update and delete operation are achieved by send the dish Id to server side, so the data in Dish table can be changed correspondingly in database. When restaurant user wants to add new food, first he needs to edit the new food in front-end and then the data will be sent to server and inserted in Dish Table.

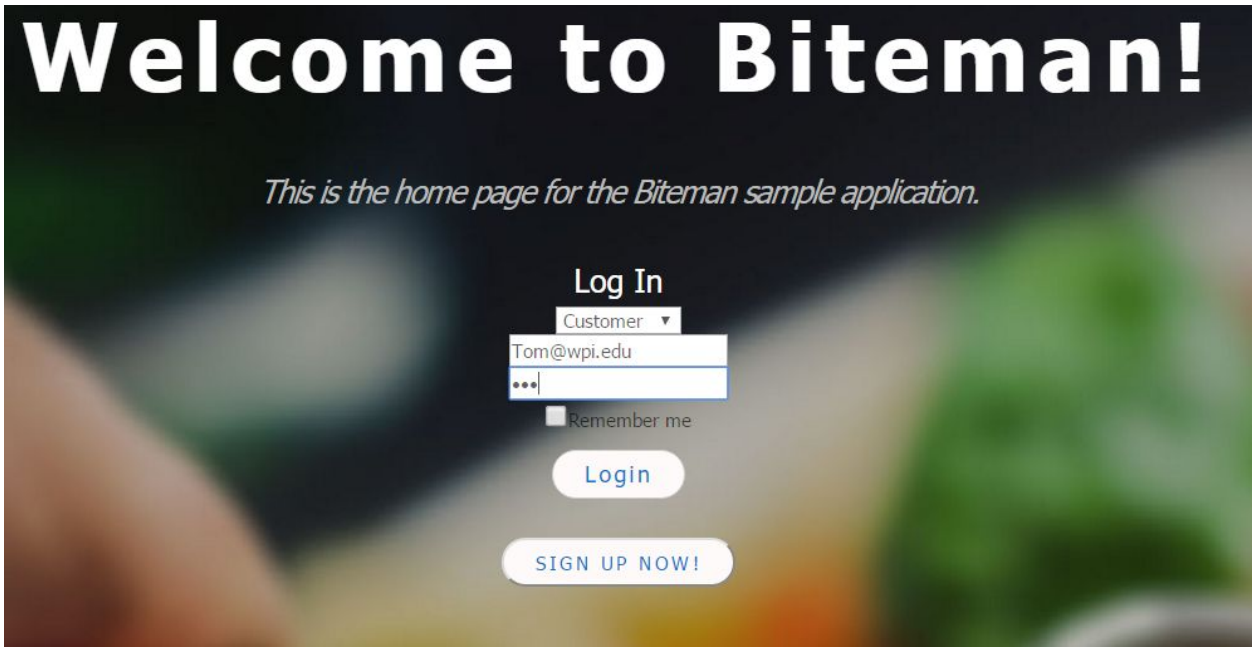
Restaurants are also allowed to update order status. On my customer page, all orders have the same restaurant Id as the current restuarant user will be returned by server. This is a little different on customer side, which is returned by same customer Id. Restaurant users can set the order status as ‘accepted’ after the customer placed order or set to status as ‘delievered’ when they finish delivering. When restuarant user changes the status, customer can notice the change on his website.

3.3 Validation

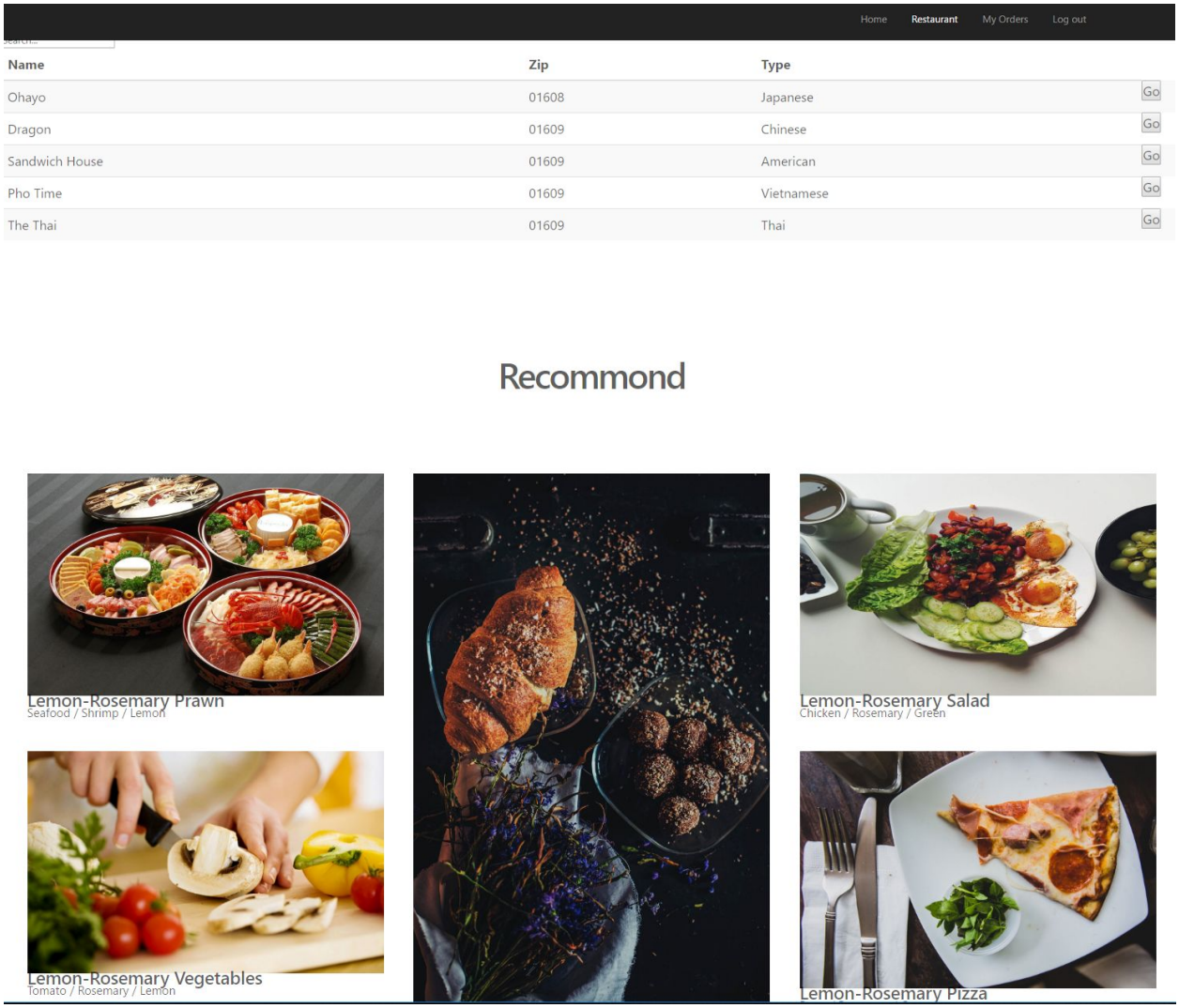
Here we used some screenshots to show the example runs of our demo.
First, we can see the homepage of our website. The header is the navigation bar, and we can login in at the main component, otherwise we can click the sign up button to create a new customer account.



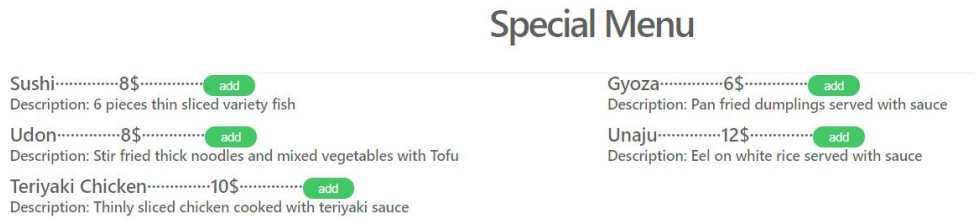
In the example, we logged in using an existing customer account ‘tom’.



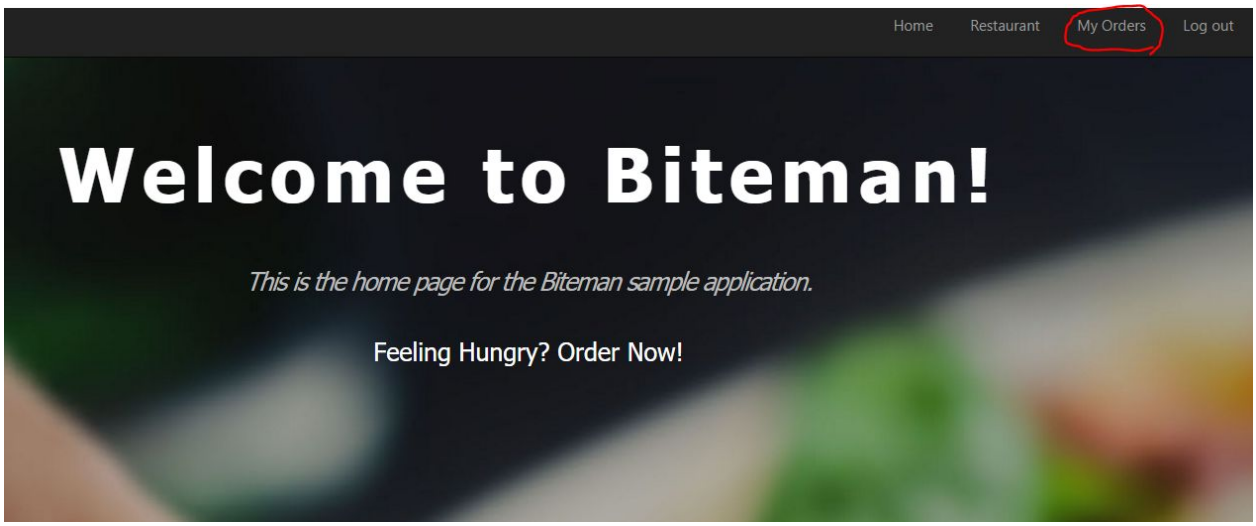
Then in the Restaurant page, we can see different available restaurants. Another two components in this page is the search bar and recommendation boxes.



By clicking the Go button besides each restaurant, we can see its menu and click ‘add’ button to add dishes to shopping cart correspondingly.

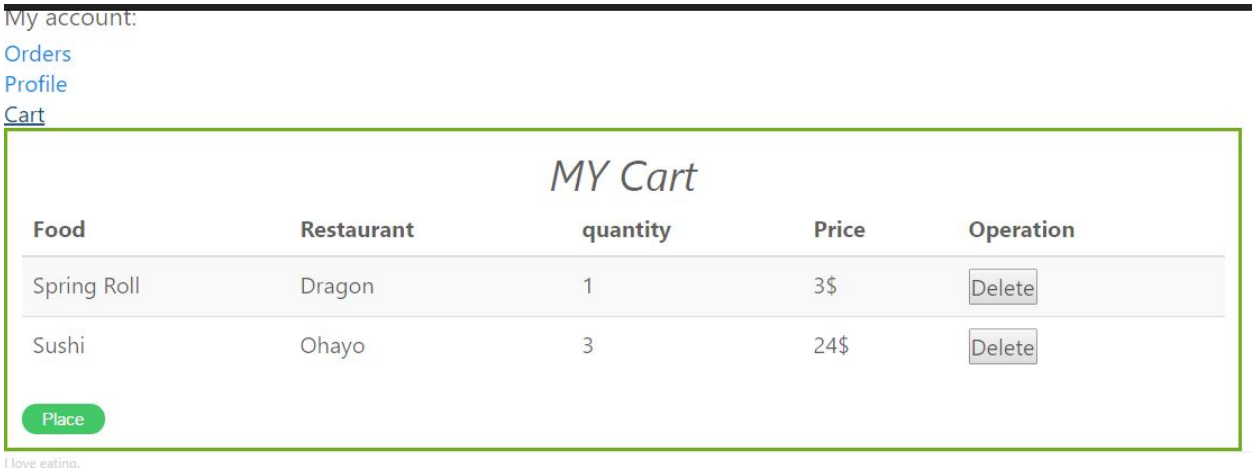


To take a look at the shopping cart or order history, we can go to ‘My orders’ page.



Three kinds of information can be checked here. The ‘orders’ provide access to view all the orders the customer has made, and managed them. The ‘Profile’ can show the simple profile of this customer which is given by the user when he or she signed up. The ‘Cart’ is the shopping cart storing all the dishes this customer added in previous step.

In ‘Cart’, the dishes just added can be deleted. By clicking the ‘Place’ button, all the dishes information will be taken out of table from the databases and grouped by restaurant ID to place multiple orders. However, different dishes from the same restaurant will be only placed as one order.

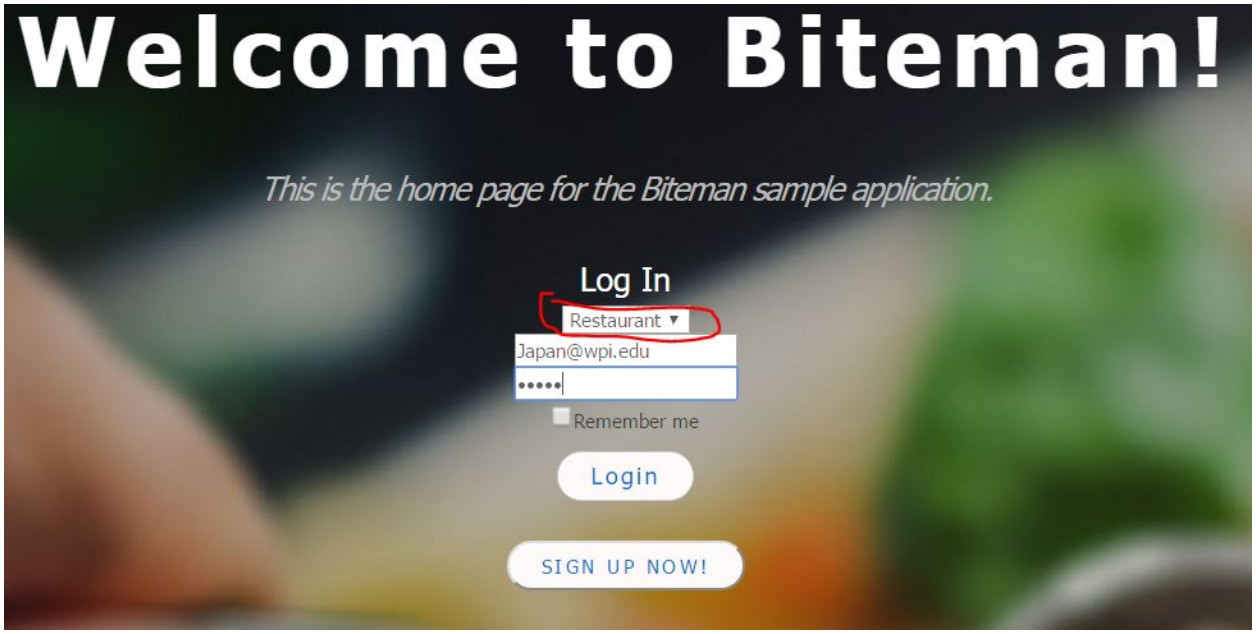


Then we can see the order we placed is shown in the order list, with a status called ‘Placed’. At

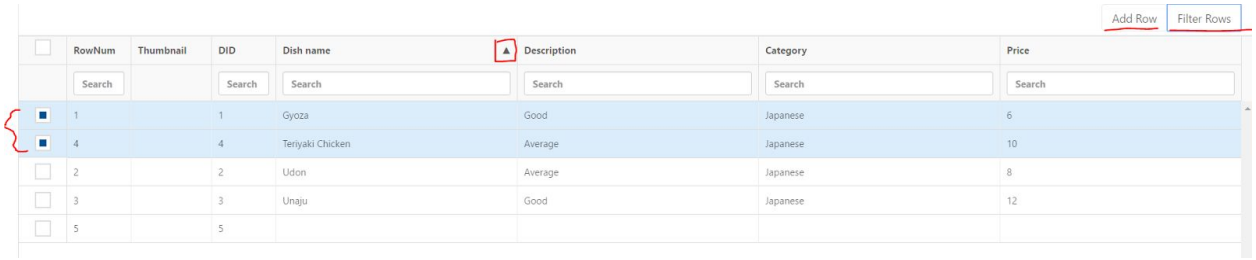
this moment, if you change your mind, you can just click ‘Cancel’ button to cancel the order, which means this order is invalid and you can do no more action on it. Otherwise, you can wait for the restaurant to accept and deliver the order, while the status will be changed by it and you can see where your order is in the process. After you got the food and the status of the order is ‘Delivered’, you can confirm the order by clicking the ‘Confirmed’ button and turn the status of the orders into ‘Finished’.

Order ID	Restaurant	Date	Time	Total price	Status	Operation
1	Ohayo	2017-04-24T04:00:00.000Z	12:30:47	16	Placed	Cancel
2	Ohayo	2017-04-24T04:00:00.000Z	12:02:43	30	Accepted	
3	Ohayo	2017-04-24T04:00:00.000Z	12:20:32	20	Accepted	
27	Dragon	2017-04-30T04:00:00.000Z	20:25:19	3	Cancelled	
28	Ohayo	2017-04-30T04:00:00.000Z	20:25:19	24	Placed	Cancel
29	Dragon	2017-04-30T04:00:00.000Z	20:27:19	3	Cancelled	
30	Ohayo	2017-04-30T04:00:00.000Z	20:27:19	24	Placed	Cancel

For the restaurant users, they can select their login type and log in as restaurant users.



We have ‘My Restaurant’ pages for users to edit their menu.



Also, the ‘My Customers’ page provide access for restaurant to view and manage their orders. As we mentioned before, we can accept the placed order and deliver the accepted order by clicking the corresponding button.

Order ID	Data	Time	Total price	Status	Operation
1	2017-04-24T04:00:00.000Z	12:30:47	16	Placed	Accepted
1	2017-04-24T04:00:00.000Z	12:30:47	16	Placed	Accepted
2	2017-04-24T04:00:00.000Z	12:02:43	30	Accepted	Delivered
3	2017-04-24T04:00:00.000Z	12:20:32	20	Accepted	Delivered
4	2017-04-24T04:00:00.000Z	12:23:11	18	Delivered	
5	2017-04-24T04:00:00.000Z	12:07:05	10	Delivered	
6	2017-04-23T04:00:00.000Z	12:15:31	22	Finished	
20	2017-04-30T04:00:00.000Z	20:05:53	54	Cancelled	

4. EVALUATION

4.1 Completion of Intent/Challenge

Completion of intent:

We basically realized all the main functions mentioned in the project proposal which includes functions for customers and those for restaurant. We also set up a database (using mysql), and connected it with our web app. Beautification (using CSS) is also worth-mentioning since it's way time-consuming than we expected, even though the logic behind is pretty straightforward.

Challenge:

- Webapp part (painful)

As expected, we don't have much trouble setting up the database, but it was painful in the web app part, especially the constructing the structures, the routers, and decided designing the components. It might be a piece of cake for graduates with CS background, but our team is stronger in statistics, and our CS skills are more related to machine learning. So this part consumed most of our energy.

- New trade-offs during the real world application

As we've been taught in the class, we already have many trade-offs in the database part, like how to design schema and queries considering both efficiency and storage costs.

However, when we designed both database and the web app, more trade-offs showed up. For example, we had some very trivial trade-offs like whether generating primary key in the front-end (web app) or in the database when user insert a new row.

We also had some big trade-off which could greatly the efficiency. For example, suppose user updates an attribute in his/her view, and the updated <key, value> pair is sent to the database. After database is updated, we have 2 alternatives to refresh user's view:

(1) Query out the view, and refresh the view
Using this alternative, we don't need much coding in the web app part. We could just query out new view and refresh whenever update happens. This is convenient and safe (what user see is always exactly what the corresponding data are in the database. Consistent!). But this alternative is apparently consuming.

(2) Refresh the view in the web app immediately after update.
Using this alternative, we are still sending updated <key, value> pairs to database. But instead of querying out the updated view and refreshing, we refresh the view in the front-end(web app). This is super fast (because this saves some time of commuting between web app and database.) However, this is risky (What we see in the web app might not be consistent with database), and much more coding in the webapp part.

To obtain the efficiency of alternative 2, and consistency of alternative 1, we coded constraints in the front-end (web app) corresponding to constraints in database. For example, we have constraints like the type of *price* is xxxx.xx, then we also have a javascript version of this constraint on *price* so that the web app is consistent with database.

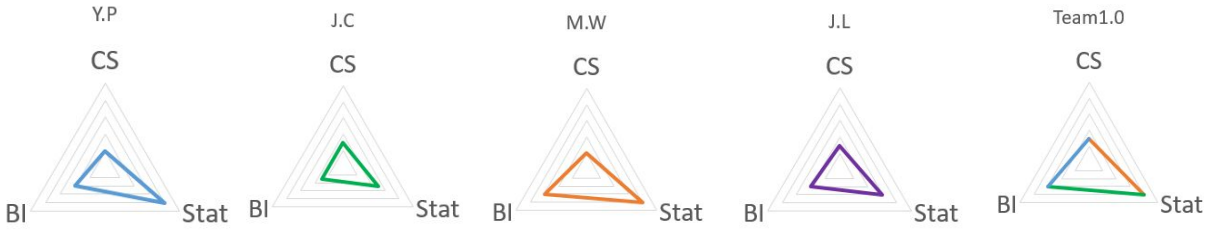
- beautification

The logic of CSS is really straight forward. But when it comes to practice, CSS is just too "art". This surprised all especially near deadline. So we also put it into the challenge list.

4.2 Lesson learned

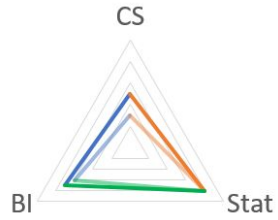
If we describe the capabilities by CS, BI and Statistics, each team member and team overall

capabilities are visualized as below:



So statistics is our strong point, then is BI. CS is a weak point for us. And our CS skills are more related to machine learning like python, R, SAS, Matlab, and Hadoop. So we are really novices with respect to web app.

After this project, we have a basic idea and basic skills to design a web app including connecting with database. And all work was done in a very real world context. Understanding the logic is important, but implementation in a specific real-world problem is not less important and easier. So both CS and BI skills were developed to some extent.



4.3 Contribution

Unlike the work distribution of the proposal and progress report, we met way more frequently as planned and was trying to figure out the hardest coding part (web app) part together, even it seems our personal contributions are disjoint. So we use “host” and “assistant” to describe the contribution sorted on project step rather than on team member.

Planned	**Actual**	**Host**	**Assistant**	**Status**
Conceptual	Conceptual	Yalei Peng,	Jianjun Luo,	Closed on 3/15

design	design	Meng Wang	Jiaoyan Chen	
Logical design	Logical design	Meng Wang, Jianjun Luo,	Jiaoyan Chen, Yalei Peng	Closed on 3/19
Physical schema design	Physical schema design	Yalei Peng, Meng Wang	Jianjun Luo, Jiaoyan Chen	Closed on 3/26
HTML and CSS	Webpage framework	We study and figure it out together. Y.P: Restaurant Component J.L: Customer Component M.W: Some Features in different components J.C: Connect app with database		Closed on 4/12
User interaction	Webpage component			Closed on 4/21
Test	CSS & Test & adjustment	All		Closed on 4/23
Report	Report	All		Closed on 4/30

5. CONCLUSION

Our project aims at implementing an online food ordering application for both diners and restaurants. We successfully build the online ordering platform with main features by building the backend database using MySQL, using React to develop dynamic web page as user interface and Node.js to handle message passing. In the future, more functions, like comment and personal recommendation, can be added into the platform.