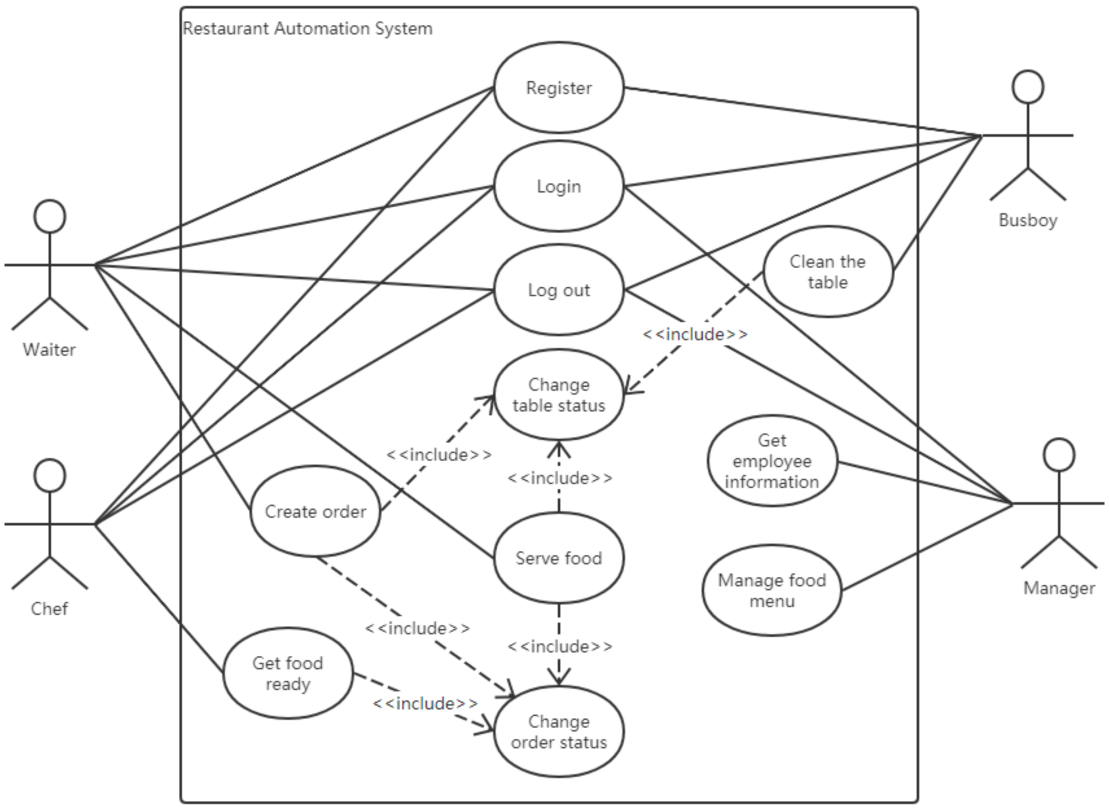
CPTS 583 Software Quality Deliverable 2-3

Restaurant Automation System

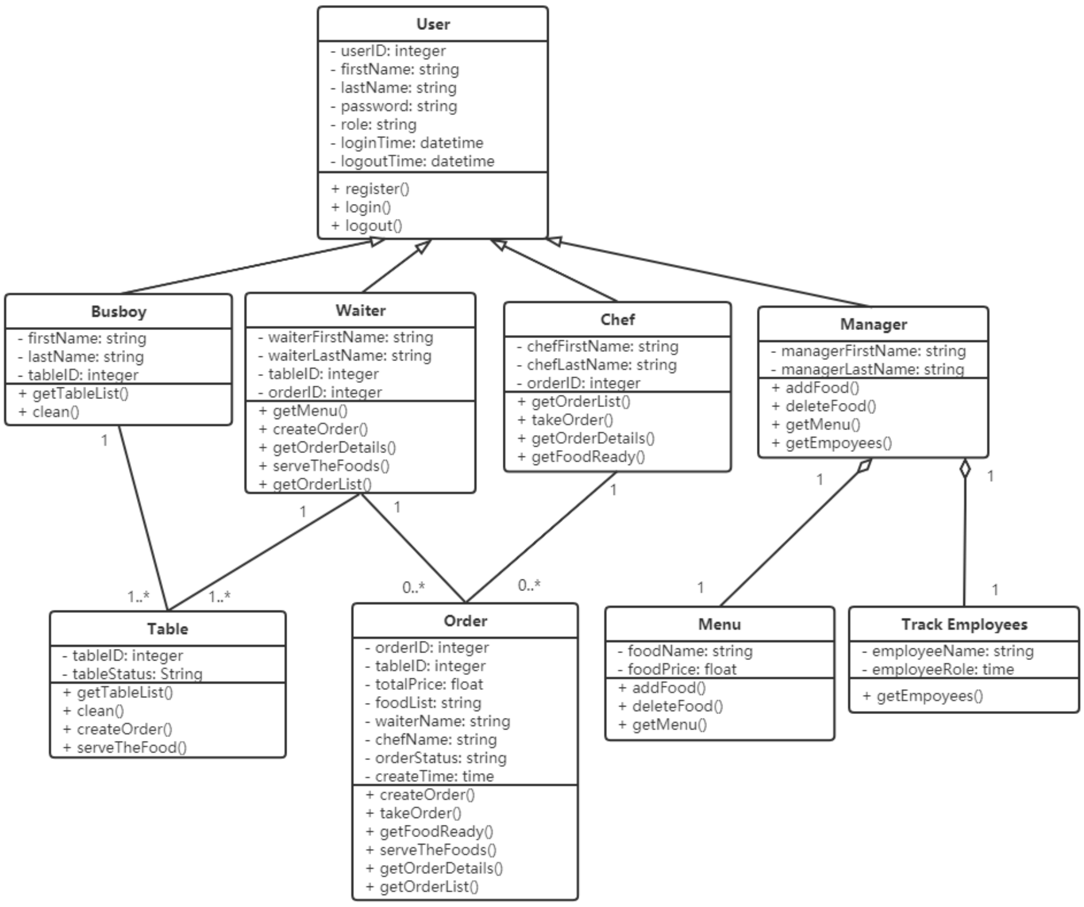
Team name: Code Cougar
Team member: Jianqiao Liu, Wen-Chih Li, Parikshit Panwar

1 Fixed/Improved analysis and design

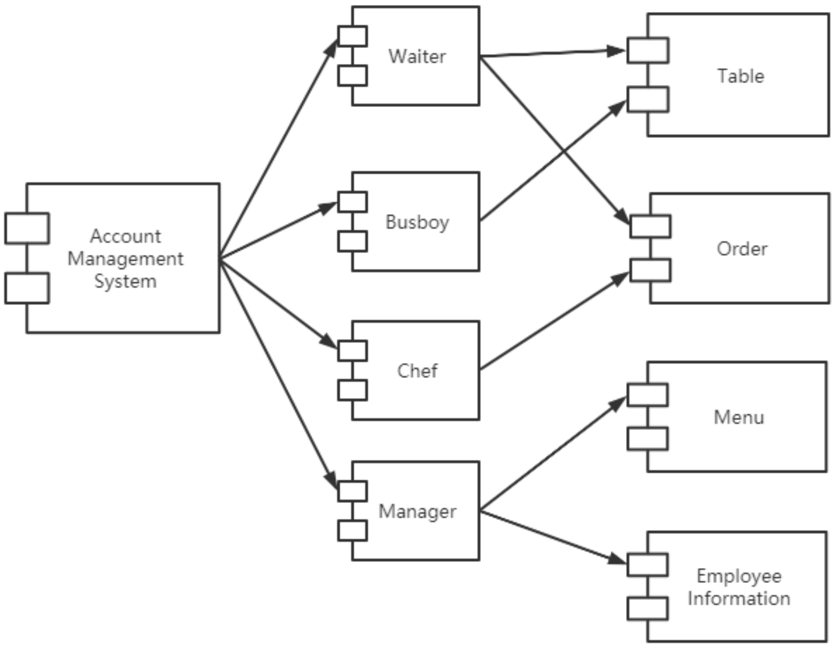
1.1 Use case diagram



1.2 Class Diagram

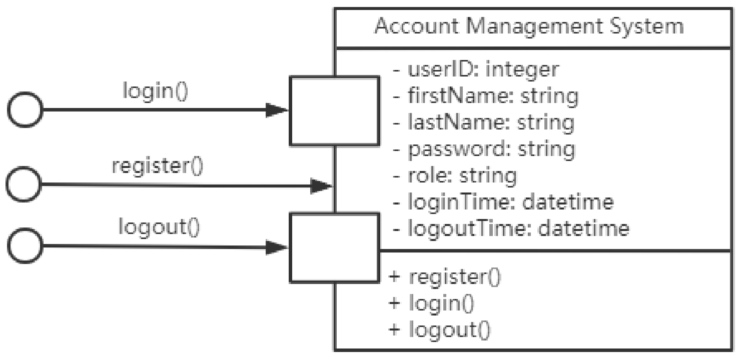


1.3 Architectural Design

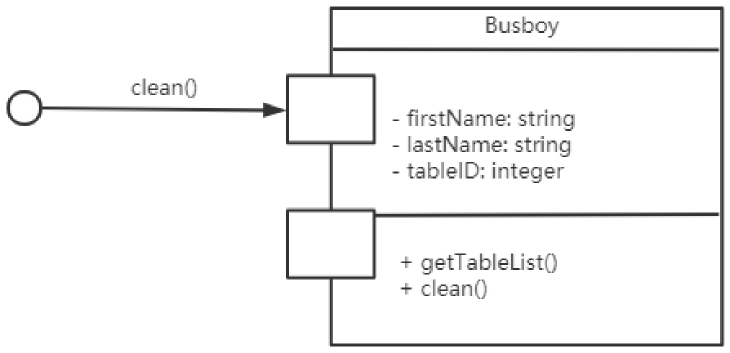


1.4 Component-level design

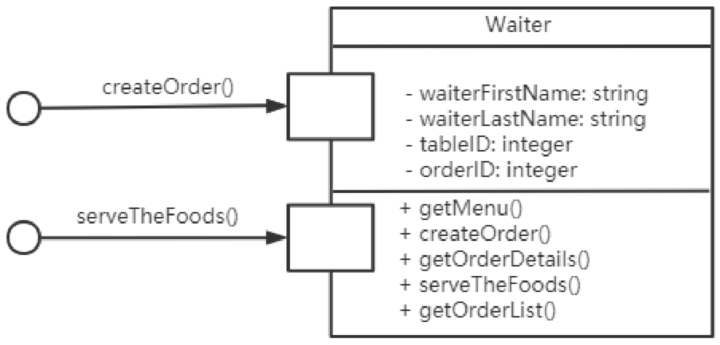
Account Management System



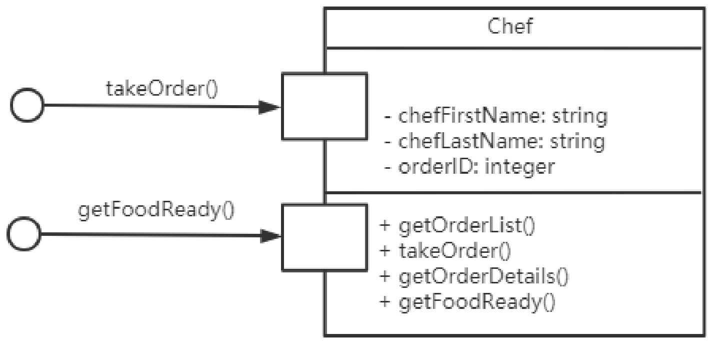
Busboy



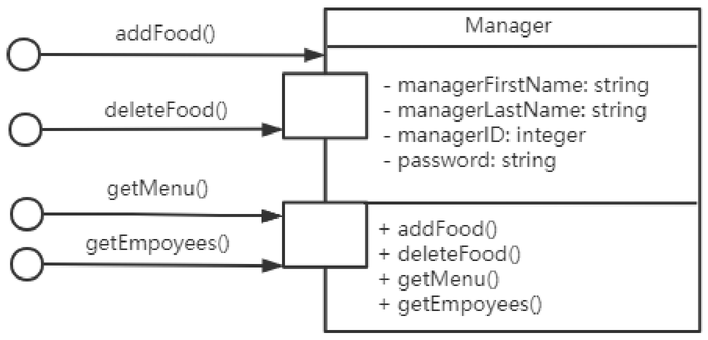
Waiter



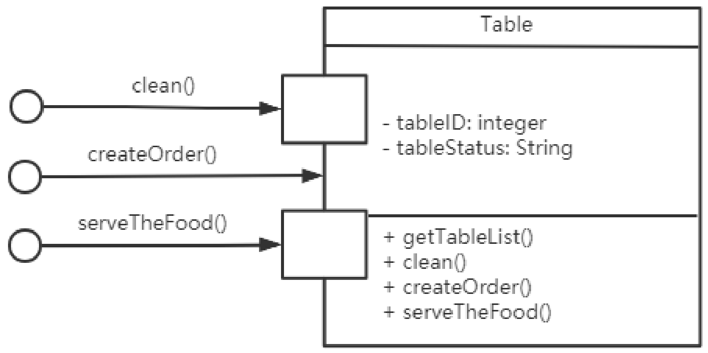
Chef



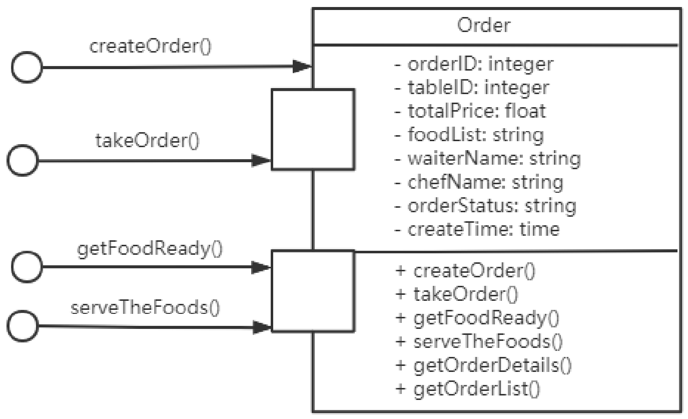
Manager



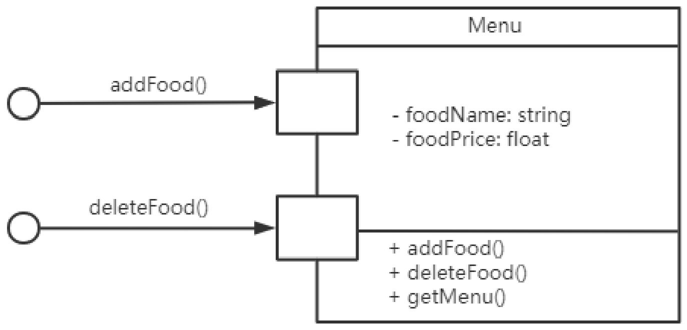
Table



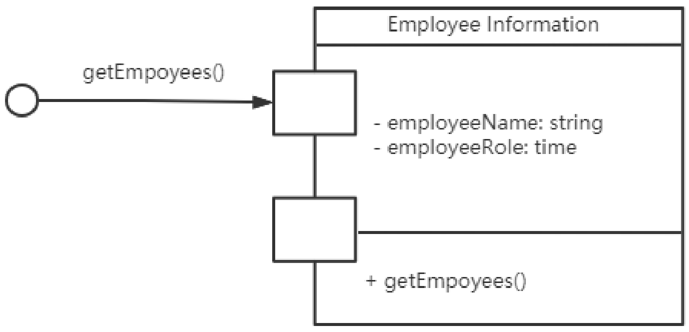
Order



Menu



Employee Information



2 Project Quality Report

2.1 Product Quality

(1) Reliability

- **Goal:**
  The system should produce correct output sequences at all times.
- **Metrics:**
  Fewer than 2% occurrence of errors should be accepted.
- **Results:**

This quality goal is satisfied. After we have done unit testing and integration testing (can be seen in the Appendix at the end of the report), all functionalities work well with no errors. In our system, we do such things to make our system reliable:

1) The system divides user roles by functional modules, each user role can only use its corresponding functions.
2) The system will verify the username and password.
3) When the user makes an operating error or the system error occurs, there are accurate and clear prompts so that the user can know the cause of the error.

(2) Usability

- **Goal:**
  The system should be easy for every user to use it.
- **Metrics:**

The system should have an understandable user-interface and it only takes less than 5 hours for a new staff to learn how to use the functions the staff needs the know.

- **Results:**

(Since the system is not deployed to the cloud server, and because of the COVID-19, it is impossible to find restaurant staffs to do the usability testing, so we do it by ourselves)

During the usability test, we found that it will only take 30 minutes for a new user to learn to use this system. Due to the short development time, the user-interfaces are quite simple and crude. But the instruction and interaction of the interfaces are not difficult to understand. However, we also found some bad interactive operations and did some improvements:

1) When users want to see the menu, list of order and order details, they have to click the corresponding buttons, such as "Get Food Menu", to get the data, this step is redundant.

Thus, we redesign the function: When the page loads, it will automatically call the function and render the data to the page, so users do not have to click the buttons to get the data.

2) Both the busboy's "table status" page and chef's "order list" page does not contain a "Back" button which might be difficult for the user to track to the previous page. Thus, we add the "Back" button to improve the usability.

(3) Efficiency

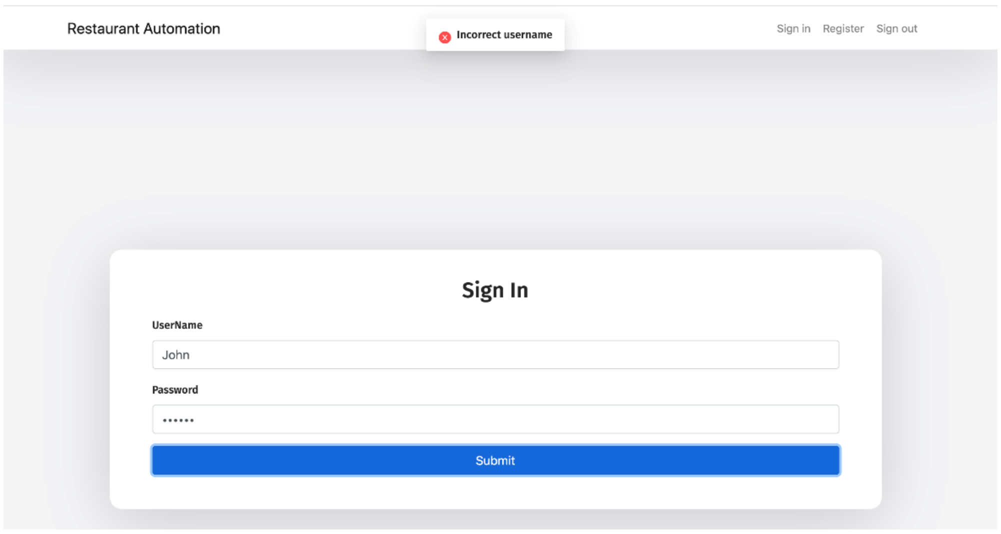
- **Goal:**
  The system should response smoothly.
- **Metrics:**
  The system should respond to the commands in less than 1 second.
- **Results:**

This quality goal is satisfied, since when we have done the testing, we found that the average response time of the system is 10ms, and the highest response time is 25ms. It shows that the system response smoothly.

(4) Integrity

- **Goal:**
  The system should be secure.
- **Metrics:**
  The system needs a registered user to log in in order to use any functionalities.
- **Results:**

This quality goal is satisfied as the system only allows authenticated user to gain access to the automation system. Therefore, the system provides 100% protection.



If the username (or password) provided is incorrect, the system will give a notification displaying "Incorrect username" (or "Incorrect password") and will not grant user access unless all the credentials are satisfied.

(5) Robustness

- **Goal:**
  The system should cope with unexpected or incorrect inputs.

- **Metrics:**
  When users input data, the system should perform correctly and there should be zero errors.
- **Results:**
  This quality goal is not satisfied, since there might be errors occur when users input data.

Thus, we made a conclusion of these errors and do some improvements:

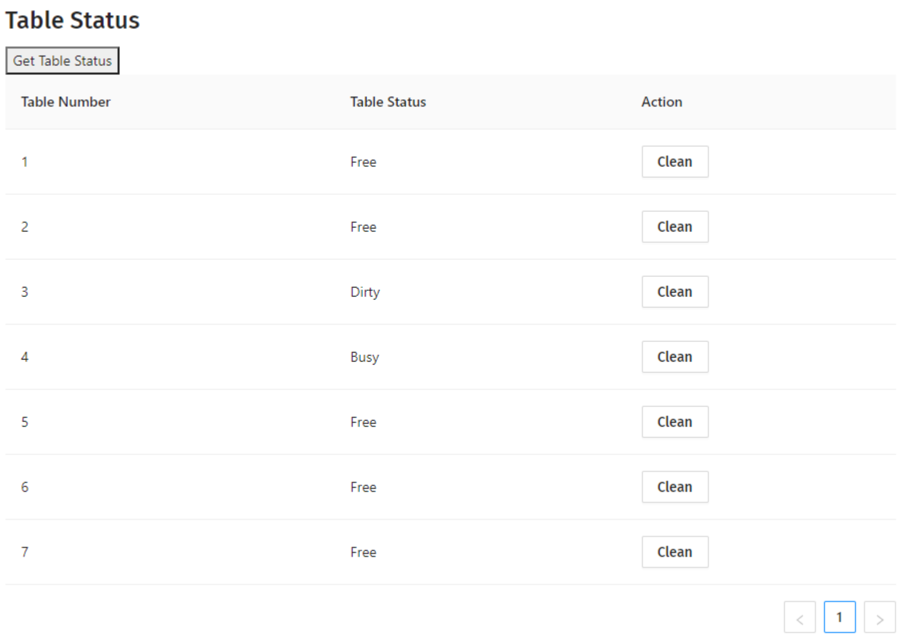


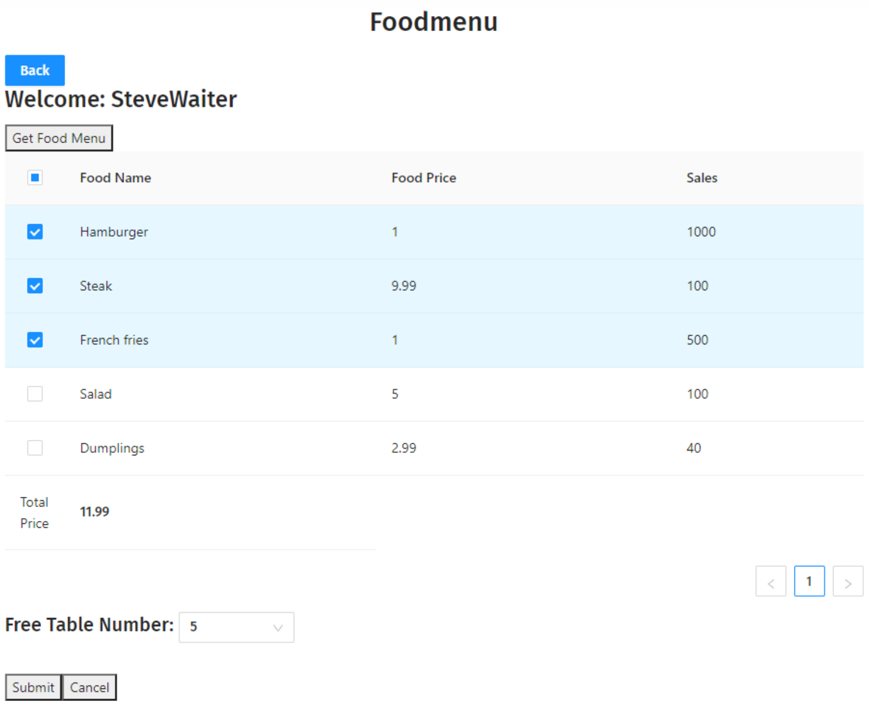
Table Status

Get Table Status

Table Number	Table Status	Action
1	Free	Clean
2	Free	Clean
3	Dirty	Clean
4	Busy	Clean
5	Free	Clean
6	Free	Clean
7	Free	Clean

1

1) For the Busboy's clean table function, only if the table status is "Dirty", busboy can click the "Clean" button to clean it. However, at first, if the busboy cleans the table which is "Busy" (means the table is occupied by customers) and "Free" (means the table is clean and not occupied), it will cause errors. Thus, we added some judgement sentences to avoid these errors, when the busboy clicks the wrong table, an error message will be displayed on the screen to tell the busboy what he/she did wrong.



Foodmenu

Back

Welcome: SteveWaiter

Get Food Menu

	Food Name	Food Price	Sales
✓	Hamburger	1	1000
✓	Steak	9.99	100
✓	French fries	1	500
	Salad	5	100
	Dumplings	2.99	40
Total Price	11.99		

1

Free Table Number: 5

Submit Cancel

2) For the Waiter's create order function, if the waiter does not select food and table and click "Submit", it will cause system to crash. Thus, we added some codes to judge if the waiter's selection is complete, otherwise, an error message will be displayed on the screen to tell the waiter what he/she did wrong.

(6) Compatibility

- **Goal:**

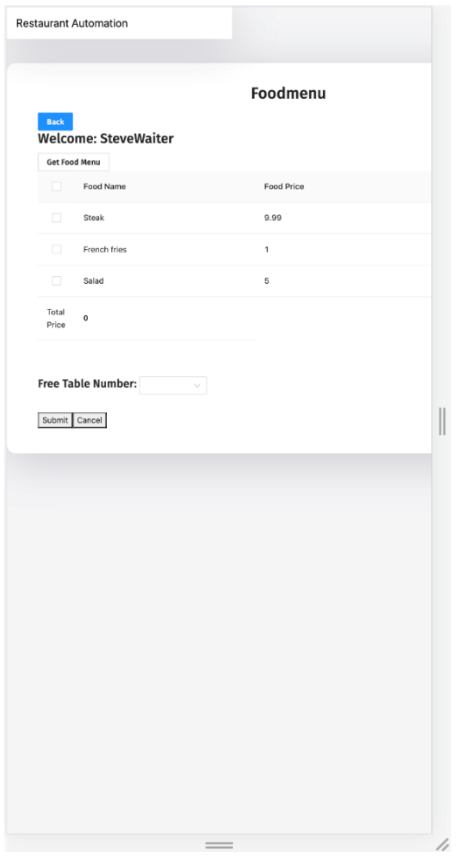
  The system should work both on computers and mobile phones.

- **Metrics:**

  All the functionality of the system should work properly on both computers and mobile devices.

- **Results:**

  This quality goal is not satisfied, the system does not seem to be compatible with mobile screens as the page becomes short in length. Also, the width of the page does not adjust to the width of the mobile screen. Therefore, it can be concluded that the web pages developed for the software are not compatible on a device having screen dimension less than that of a laptop monitor.



Restaurant Automation

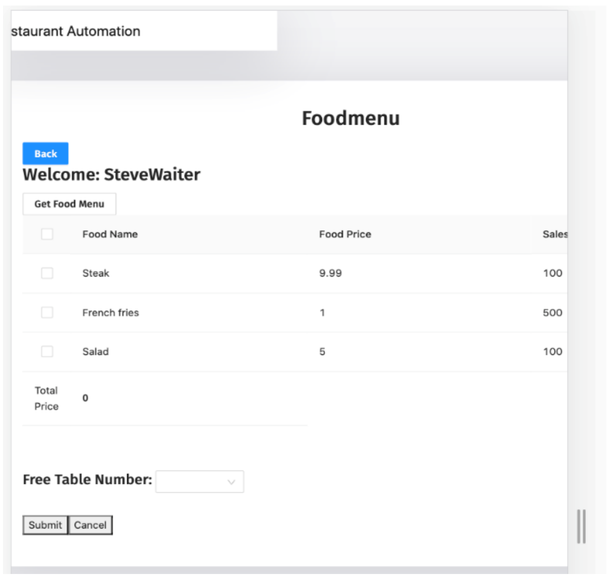
Foodmenu

Welcome: SteveWaiter

	Food Name	Food Price
☐	Steak	9.99
☐	French fries	1
☐	Salad	5

Total Price 0

Free Table Number:



staurant Automation

Foodmenu

Welcome: SteveWaiter

	Food Name	Food Price	Sales
☐	Steak	9.99	100
☐	French fries	1	500
☐	Salad	5	100

Total Price 0

Free Table Number:

2.2 Process Quality

(1) Purpose

- **Goal:** To predict the cost and function of the process and product in order to plan of it, and to evaluate the process and product in order to understand and improve it.
- **Metrics:** The total time taken to develop the product must be no more than 16 hours the total estimated effort time.
- **Results:**

By looking at the two following tables, it can be seen that the total actual effort in table 2 is within the total estimated effort in table 1. Hence, it can be said that the process for developing the software product was reliable.

The estimated cost for developing the project was as given in the following table:

Table 1.

Task Name	**Estimated Effort (hours)**	**Implementation**	**Evaluation**	**Prevention**	**Rework**
Project planning	10	9	0	0	1
Infrastructure setup	3	3	0	0	0
UI Design	20	15	2	0	3
Initial Project development and Unit Testing	55	35	5	5	10
Quality Assurance - Integration testing	25	10	5	5	5
Software Bug Correction	30	15	0	0	15
Total	143	87	12	10	34

However, the actual cost for developing the project is described in the following table:

Table 2.

Task Name	**Actual Effort (hours)**	**Implementation**	**Evaluation**	**Prevention**	**Rework**
Project planning	10	9	0	0	1
Infrastructure setup	5	5	0	0	0
UI Design	13	11	2	0	1
Initial Project development and Unit Testing	58	45	3	5	5

Quality Assurance - Integration testing	25	9	3	5	8
Software Bug Correction	19	5	4	0	10
Total	131	84	12	10	25

(2) Perspective

- **Goal:** Examine the effectiveness from the viewpoint of the restaurant employees and manager so that any new employee will be able to learn to operate the software conveniently and efficiently.
- **Metrics:** Any new user must be able to operate the software with convenience within 30 minutes.
- **Results:** Since the system is not deployed to the cloud server, and because of the COVID-19, it is impossible to find restaurant staffs to do the usability testing, so we do it by ourselves. During the usability test, we found that it will only take 30 minutes for a new user to learn to use this system.

(3) Environment

- **Goal:** The restaurant employees may have no or less experience of using the software product. The restaurant employee will have miscommunication with each other regard to the process of running the task.
- **Results:** Since the system is not deployed in the restaurant for testing because of the COVID-19, therefore, this process quality goal will not be satisfied.

Appendix:

1 API unit test (with Postman)

Postman link: https://www.getpostman.com/collections/ab5a6b7cb62596affd39

API	**Request method**	**Function**		
http://localhost:3001/signin	Post	Sign in		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Sign in successful.	Use the correct username and password to sign in.	Return the "Successful" status and the role of the user.	Return the "Successful" status and the role of the user.	12ms
Sign in with incorrect username.	Use the incorrect username to sign in.	Return the "Incorrect username" status.	Return the "Incorrect username" status.	5ms
Sign in with incorrect password.	Use the incorrect password to sign in.	Return the "Incorrect username" status.	Return the "Incorrect username" status.	6ms

API	**Request method**	**Function**		
http://localhost:3001/register	Post	Register		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Register successful.	Use username, password, role, first name and last name to register.	Return the "Successful" status.	Return the "Successful" status.	11ms
Username already exists in the database.	Use the existed username to register.	Return the "Username already exists" message.	Return the "Username already exists" message.	9ms

API	**Request method**	**Function**		
http://localhost:3001/foodmenu	Post	Food menu		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Get food menu	Send the post request to backend.	Return the "Successful" status and the list of the food menu.	Return the "Successful" status and the list of the food menu.	12ms

API	**Request method**	**Function**		
http://localhost:3001/foodadd	Post	Add food to food menu		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Add food to food menu	Use the food name and food	Return the "Successful"	Return the "Successful"	13ms

	price to add food.	status.	status.	

API	Request method	Function		
http://localhost:3001/foodDelete	Post	Delete food from the food menu		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Delete food from the food menu	Use the key in the table to delete food.	Return the "Successful" status.	Return the "Successful" status.	17ms

API	Request method	Function		
http://localhost:3001/tableClean	Post	Clean table		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Busboy cleans the table	Busboy selects table ID and clean the table.	Return the "Successful" status.	Return the "Successful" status.	18ms

API	Request method	Function		
http://localhost:3001/tableAll	Post	Get all tables		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Get all tables	Users get all tables.	Return the "Successful" status and get the list of tables.	Return the "Successful" status and get the list of tables.	18ms

API	Request method	Function		
http://localhost:3001/ table	Post	Get all free tables		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Get all free tables	Users get all free tables.	Return the "Successful" status and get the list of free tables.	Return the "Successful" status and get the list of free tables.	9ms

API	Request method	Function		
http://localhost:3001/createOrder	Post	Create order		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Waiter creates the order	Waiter selects the food, table ID and submit the order.	Return the "Successful" status.	Return the "Successful" status.	25ms

API	Request method	Function		
http://localhost:3001/orderDetails	Post	Get order details		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Users get the details of order	Select the order ID and send request.	Return the "Successful" status and the details of order.	Return the "Successful" status and the details of order.	10ms

API	Request method	Function		
http://localhost:3001/serveTheFoods	Post	Waiters serve the foods		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Waiters serve the foods	Select the order ID, table ID, and send request.	Return the "Successful" status and the details of order.	Return the "Successful" status and the details of order.	12ms

API	Request method	Function		
http://localhost:3001/updateChefOrder	Post	Chef takes the order.		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Chef takes the order.	Select the chef's name and order ID and send request.	Return the "Successful" status and the details of order.	Return the "Successful" status and the details of order.	15ms

API	Request method	Function		
http://localhost:3001/getReadyOrderList	Post	Get ready order list		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Get ready order list	Select the order ID and send request.	Return the "Successful" status and the details of order.	Return the "Successful" status and the details of order.	10ms

API	Request method	Function		
http://localhost:3001/takeOrderChef	Post	Chef takes the order		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**	**Time cost**
Chef takes the order	Select the order ID and send request.	Return the "Successful" status and the details of order.	Return the "Successful" status and the details of order.	12ms

2 JEST unit test

Function	Test case	Test Procedure	Expected Result	Actual Result	Time cost
Text from sign in page	User can view the text.	See the correct text.	All the text should show successfully.	Show successfully.	12ms

Function	Test case	Test Procedure	Expected Result	Actual Result	Time cost
Username	the username displays correct.	The format of username is correct and show it on the page	Username is correct.	show successfully.	12ms

Function	Test case	Test Procedure	Expected Result	Actual Result	Time cost
Button	Users click the button.	to check the button can be click.	Button can be click.	click successfully.	10ms

3 Integration Tests

Test case	Test Procedure	Expected Result	Actual Result
Waiter	1. Waiter can register an account. 2. Sign in with Waiter's account. 3. Waiter can view the menu and select foods and a "Free" (clean and unoccupied) table to create an order. 4. Waiter can view the order detail and serve the food. 5. Waiter can get the order list and serve the food.	All the attributes are working correctly.	All the functions actually working well.
Test case	**Test Procedure**	**Expected Result**	**Actual Result**
Busboy	1. Busboy can register an account. 2. Sign in with Busboy's account. 3. Busboy can view table status. 4. Busboy can clean the "Dirty" table.	All the attributes are working correctly.	All the functions actually working well.
Test case	**Test Procedure**	**Expected Result**	**Actual Result**
Chef	1. Chef can register an account. 2. Sign in with Chef's account. 3. Chef can get the order list which the orders	All the attributes are working correctly.	All the functions actually working well.

	are not taken by other chefs and take the order. 4. Chef can view the order detail. 5. Chef can change the order status to "Ready" when the food ready.		
Test case	**Test Procedure**	**Expected Result**	**Actual Result**
Manager	1. Manager can register an account. 2. Manager can sign in with Manager's account. 3. Manager can view the employees' information. 5. Manager can get the food menu and add or delete the food.	All the attributes are working correctly.	All the functions actually working well.