MBJ Co.

ChefNODE
Software Design Report
Web Application

Version 1.0

Revision History

Date	Version	Description	Author
24/3/2017	1.0	Preliminary stages	Mohamed Sondo Jake Biller Benjamin Yi

Table of Contents

Software Requirements Specification

1. Introduction

1.1 Purpose

This documentation provides a detailed description of the ChefNODE restaurant ordering system and analyzes the different subsystems within this web application. As it is a web application, this document explains the features of the interface available to the users, how the users will interact with the application, and how the application will respond to these interactions. This document will also describe the reliability of its dependencies such as APIs and libraries published by third parties.

1.2 Scope

ChefNODE is a food delivery system which allows individuals to order food from a chosen restaurant. Customers, and potential customers, are given a window view to restaurants, allowing them to view menus of a restaurant as well as the chef(s) preparing the menus. Upon receiving the meal, customers can review their experience and rate their food.

Moreover, ChefNODE gives certain responsibilities to users besides customers – the employees of the restaurant: the manager, chefs, and delivery men. These responsibilities include such things as changing the items on the menu (chefs), the hiring and firing of employees (managers), and choosing a delivery route (delivery).

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Manager (MG)	A restaurant personnel – has access to a control panel which handles customer registrations and work status of its employees (hire/fire/promote).
Chef (CH)	A restaurant personnel – can set up a menu.
Delivery Personnel (DP)	A user who can access and view orders placed, and plan a route of delivery of the order.
Registered Customer (RC)	A registered customer who can also browse/search for restaurants and view the restaurant's details. Furthermore, RC can place an order and provide feedback for the restaurant.
Non-registered Customer (NC)	A visiting user who can only browse/search for restaurants and view the restaurant's details and menu.
System (SY)	The system provides checkpoints for specific actions of the system's users.
Manager Control Panel (MG-CP)	A control panel specific to a manager user.
Chef Control Panel (CH-CP)	A control panel specific to a chef user.
Node.js	An asynchronous, event-driven JavaScript runtime built to build scalable network applications.
Express.js	A web application framework for Node.js designed to build web applications and APIs.
React.js	A JavaScript framework for building and manipulating user interfaces.
MongoDB	A NoSQL database program designed to use a JSON-like format (or a hash format) to store data.
Passport	An authentication middleware for Node.js which allows applications to have easy registration through social media applications such as Facebook and Google+.

Materialize	A CSS framework utilizing Material Design.
MERN	A scaffolding tool allowing for easy-to-build applications using Mongo, Express, React, and Node.
Google Maps API	An API that allows developers to embed maps and locate specific places such as restaurants.
Big Oven API	An API that allows calls to a very large database of food items and recipes.

1.4 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. *IEEE Computer Society, 1998*.
"Node.Js V7.7.4 Documentation". Nodejs.org. N.p., 2017.

1.5 Overview

The remainder of this specification will detail the functionality of this web application

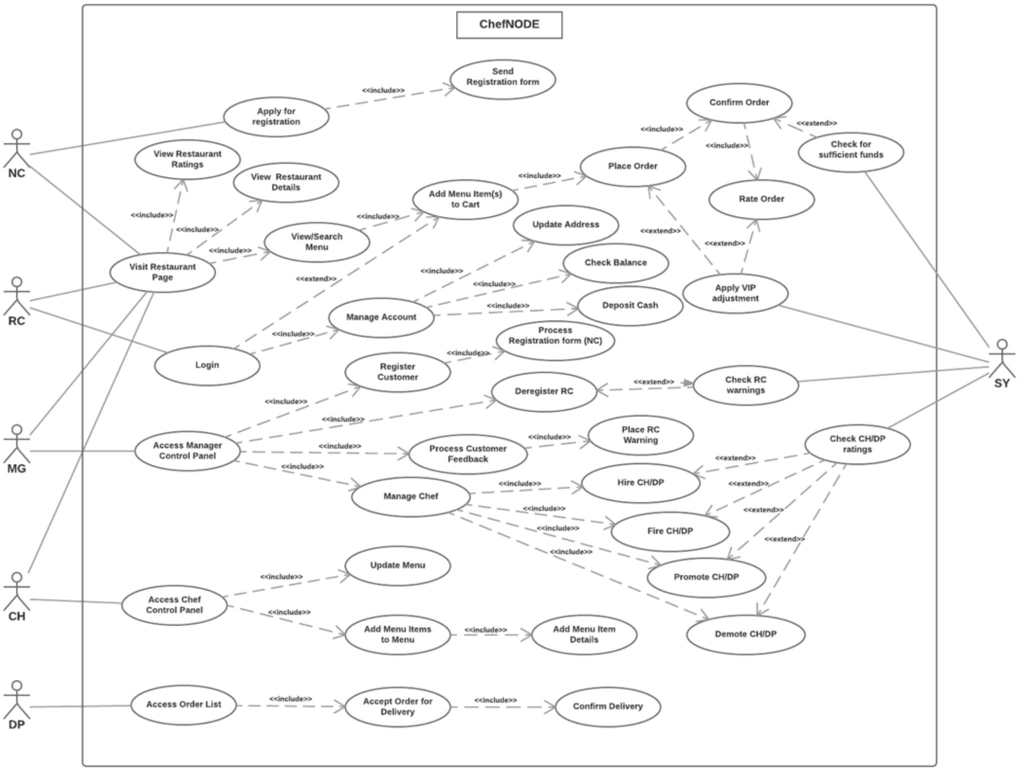
The second, following section will provide a brief description of the application by reviewing use-cases and dependencies.

The third section provides some better details of the applications not included in it's prior sections and uses more specific definitions to describe different use-cases and dependencies of this application.

2. Overall Description

2.1 Use-Case Model Survey

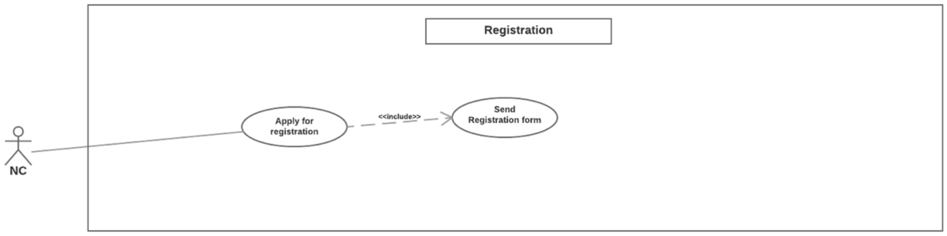
The following diagram provides a visual to available features of the application unique to each user.



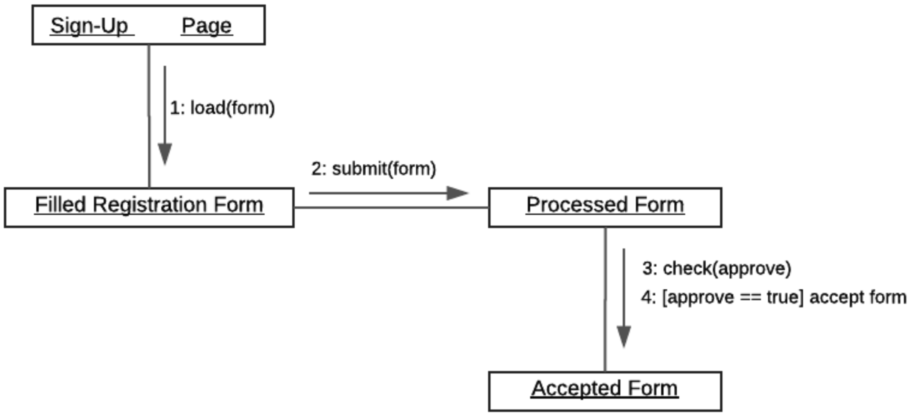
2.2 Use-Case Reports

2.2.1 Registration

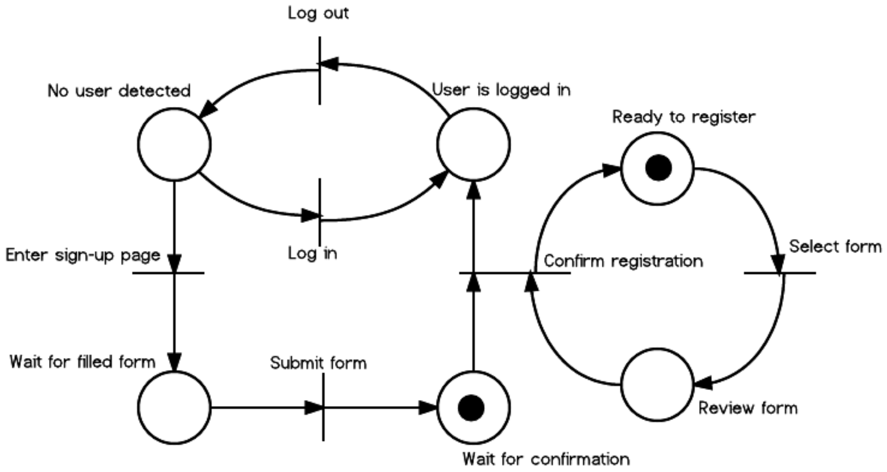
Use case diagram:



Collaboration diagram:



Petri-net:



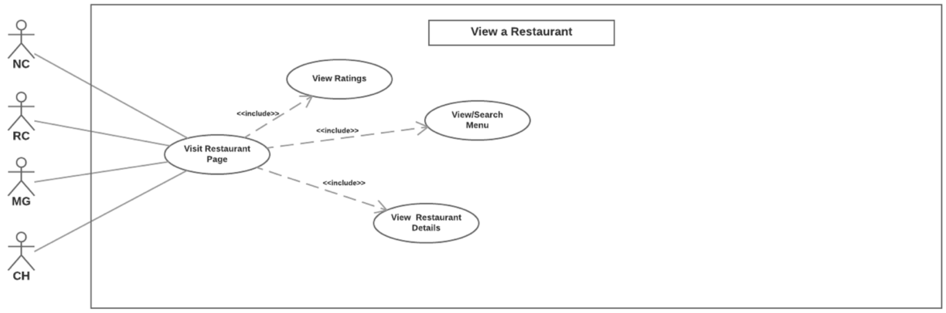
Description: Assuming the system is displayed properly to a visiting user, a non-registered customer (NC) can view a restaurant and its details. But, to place an order, the NC must first submit a registration form to become a registered customer (RC). Once an RC, the user gains multiple access controls for the system.

Details:

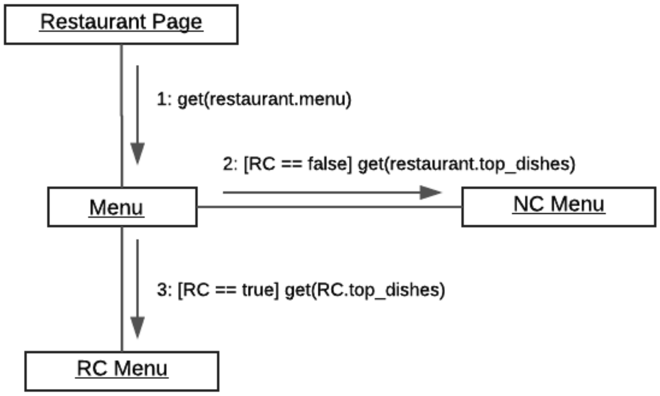
1. NC enters the “sign-up” page
2. NC enters valid information on the sign-up form
3. NC submits/sends the form and waits for approval
4. MG views the form and approves, thus registering the customer
5. SY creates an account for NC, which then becomes RC

2.2.2 Viewing a restaurant page

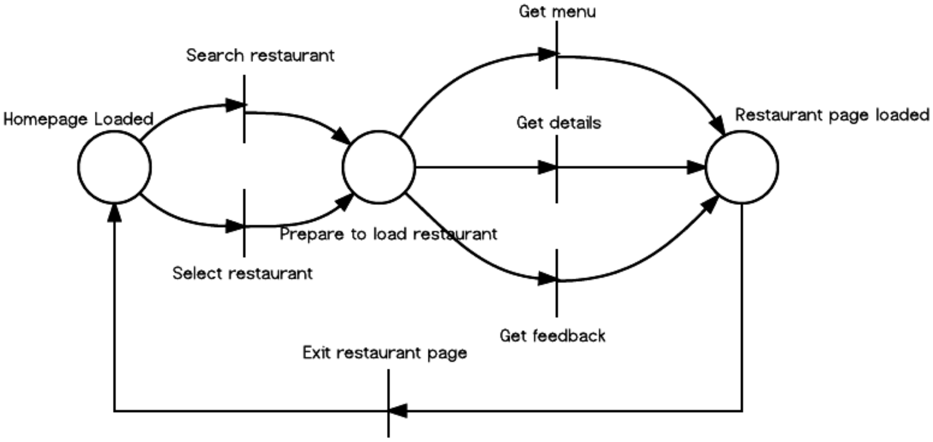
Use case diagram:



Collaboration diagram:


Restaurant Page
1: get(restaurant.menu)
2: [RC == false] get(restaurant.top_dishes)
Menu
NC Menu
3: [RC == true] get(RC.top_dishes)
RC Menu

Petri-net:



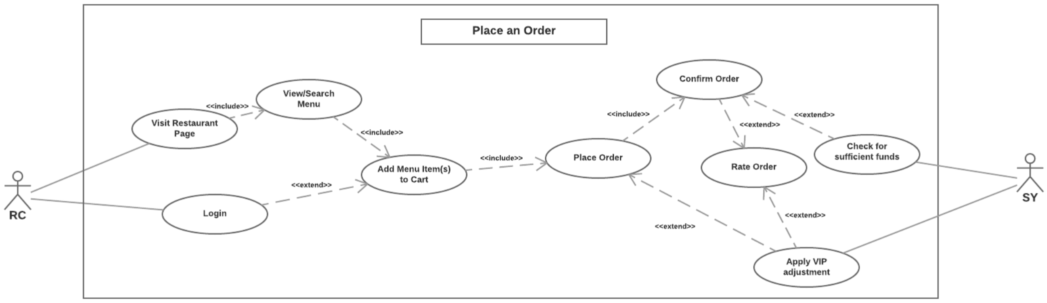
Description: All users will be allowed to view a restaurant page, which includes the ratings, menus, and additional details of the restaurant.

Details:

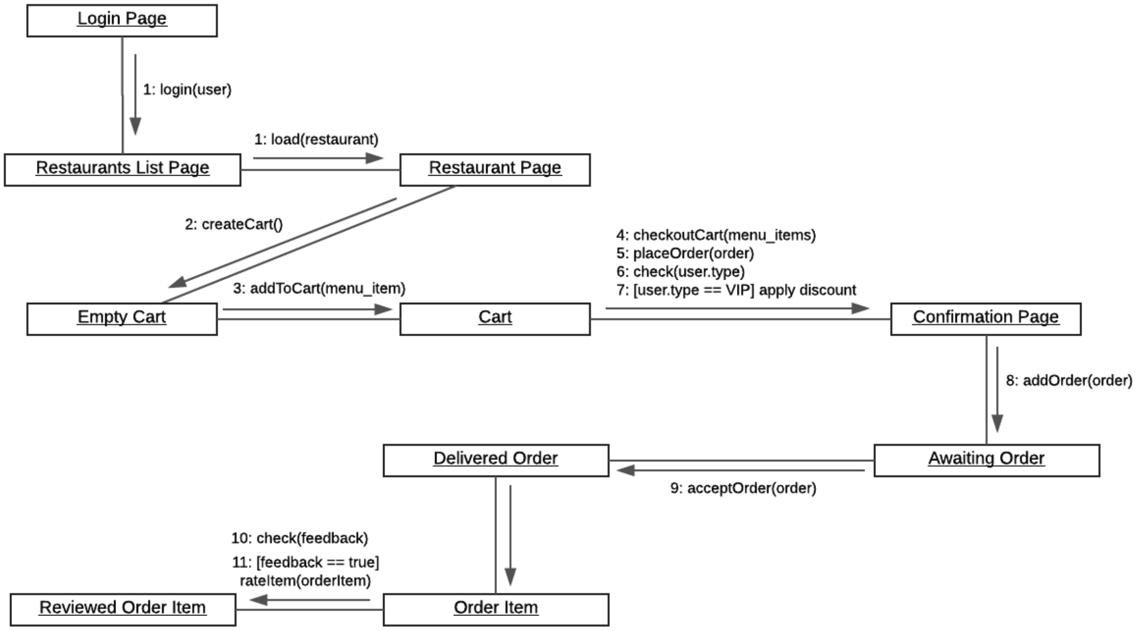
1. User connects to the website
2. User browses restaurants and chooses a restaurant
3. User views the restaurant's page
4. User views ratings
    a. Ratings will be an average of feedback received by RCs who ordered from current restaurant.
5. User views menu
    a. The menu will present the top five most popular dishes at the top of the menu
6. User views details
    a. View vicinity and location of current restaurant
    b. View hours of operation of current restaurant
    c. View CH operating at the current restaurant

2.2.3 Placing an order

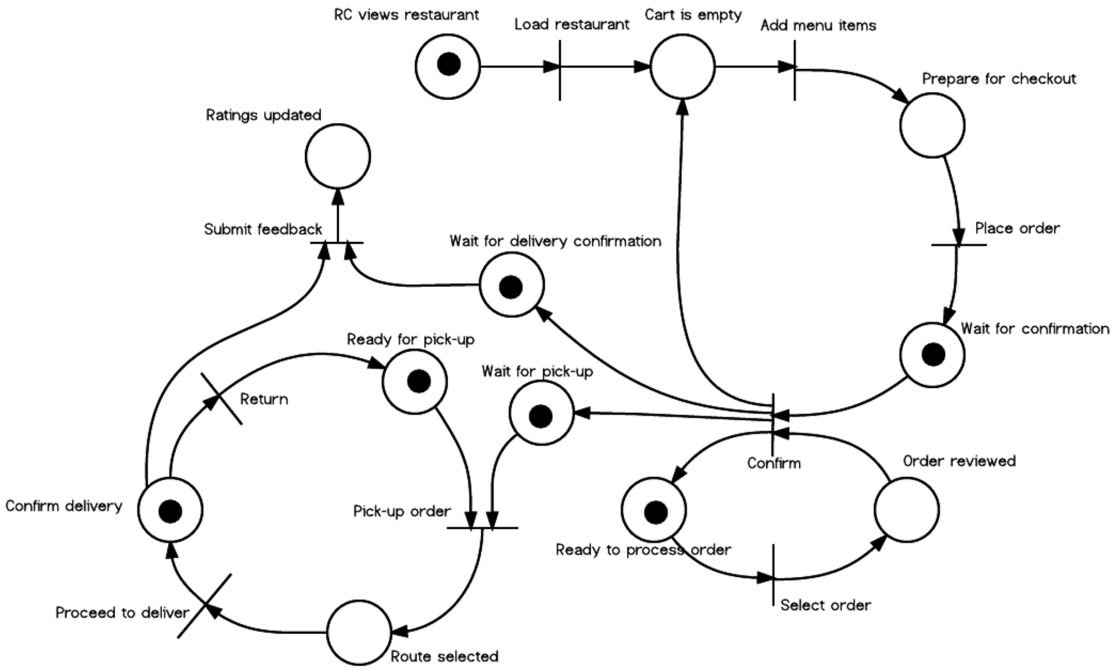
Use case diagram:



Collaboration diagram:



Petri-net:



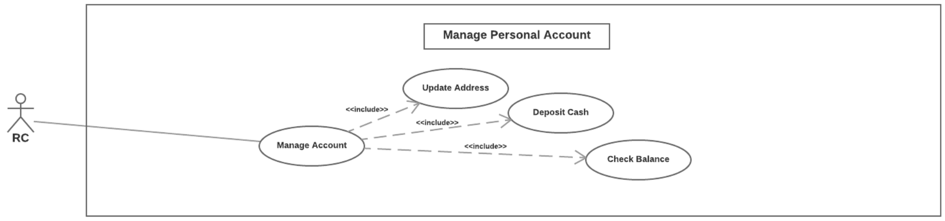
Description: Once registered, a user gains an online cart to add menu items from a menu of a restaurant. Once the user has desirable items in the cart, the user can place the order.

Details:

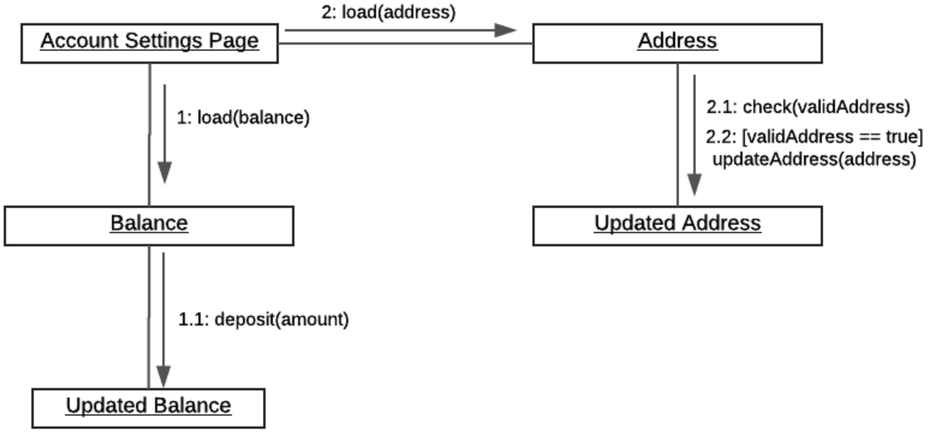
1. User finds a restaurant and entered the restaurant's page
2. User views and searches the menu of the restaurant
3. User cannot proceed further unless the user is logged in as RC
4. RC adds menu items to the cart
5. RC places the order
6. SY checks the RC for sufficient funds, and adjusts total
7. Order is confirmed
8. RC options to rate the order, and SY applies an adjustment to the rating

2.2.4 Managing personal accounts

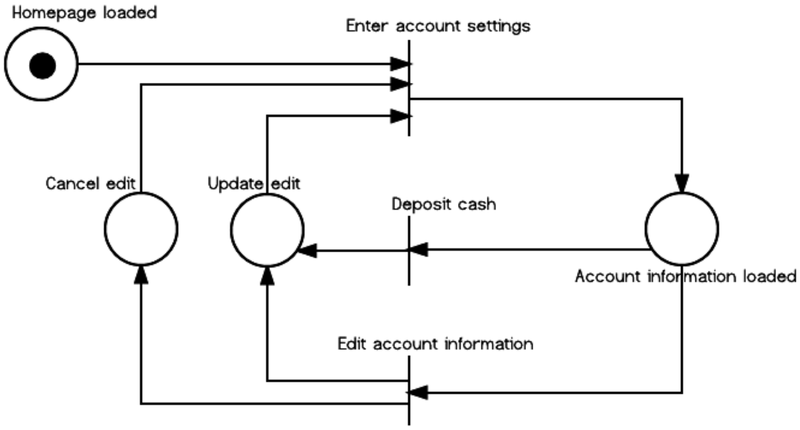
Use case diagram



Collaboration diagram:



Petri-net:



Description: A RC must manage the RC account to have a valid address and enough balance to place an order.

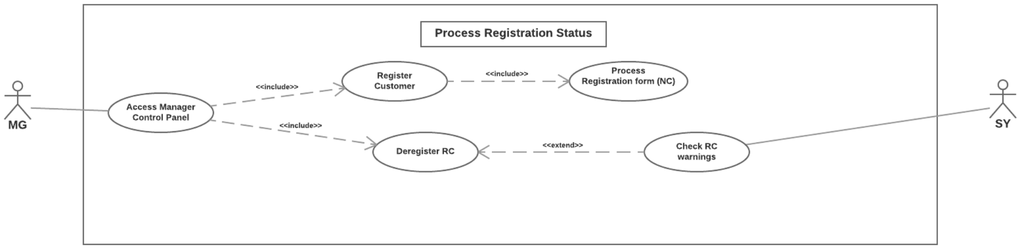
Details:

1. RC visits account settings page
2. RC edits and saves address details
3. RC transfers or deposits cash into the account

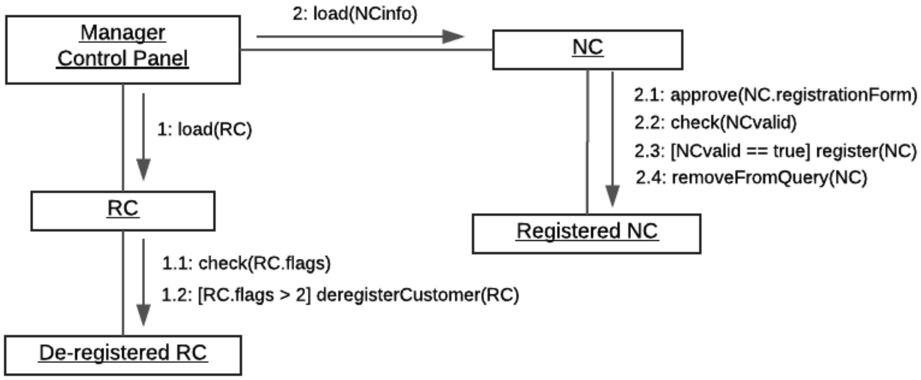
RC can check the current balance

2.2.5 Processing registration status

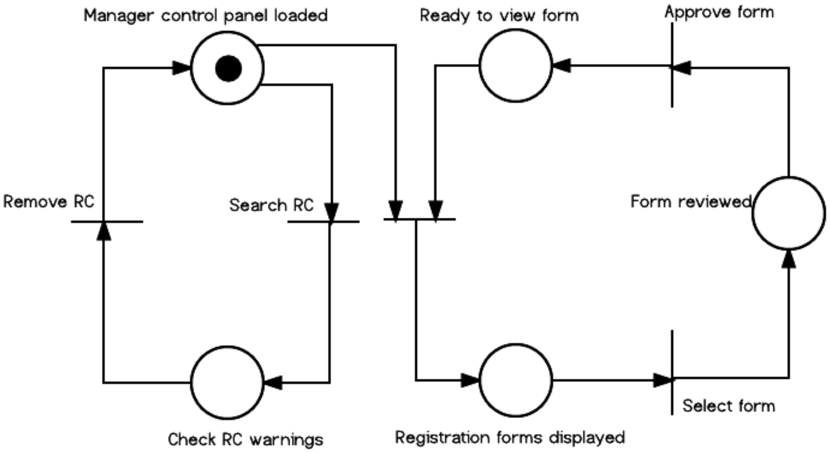
Use case diagram:



Collaboration diagram:



Petri-net:



Description: MG is granted special access through the MG-CP (manager control panel). This control panel allows MG to manage the system’s users.

Details:

Registration process:

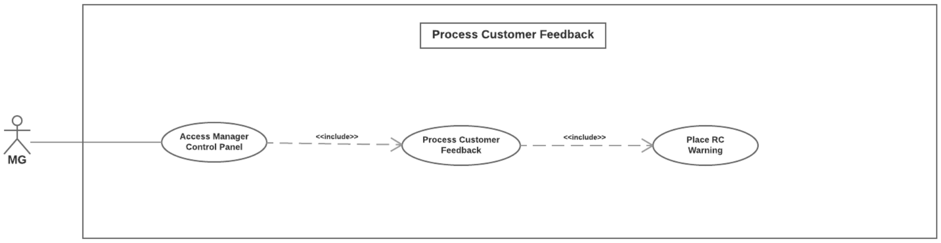
1. User logs in as MG
2. MG views submitted forms
3. MG reviews a specific form
4. MG accepts form and a RC account is created

De-registration process:

1. User logs in as MG
2. MG views RC and prepared to de-register
3. SY checks RC account for a specific number of warnings
4. SY allows de-registration, and RC account is destroyed

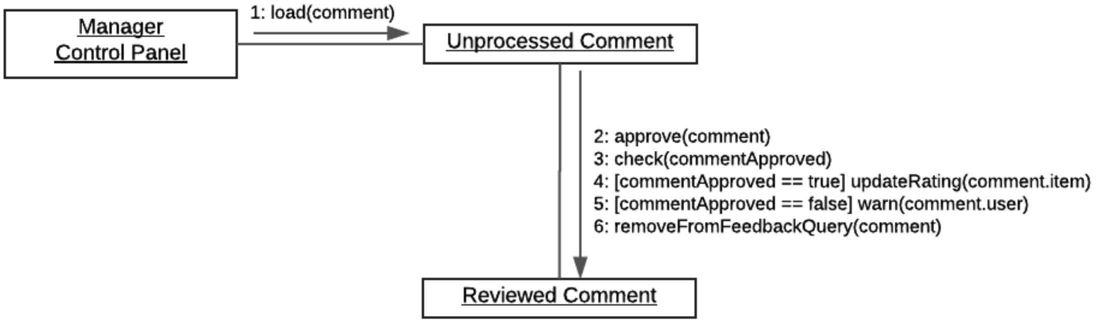
2.2.6 Processing customer feedback

Use case diagram:



Process Customer Feedback
MG
Access Manager Control Panel
<<include>>
Process Customer Feedback
<<include>>
Place RC Warning

Collaboration diagram:


Manager Control Panel
1: load(comment)
Unprocessed Comment
2: approve(comment)
3: check(commentApproved)
4: [commentApproved == true] updateRating(comment.item)
5: [commentApproved == false] warn(comment.user)
6: removeFromFeedbackQuery(comment)
Reviewed Comment

Petri-net: **(See 2.2.3)**

Description: MG is granted special access through the MG-CP (manager control panel). This control panel allows MG to process feedback given by RC.

Details:

Accepting a comment

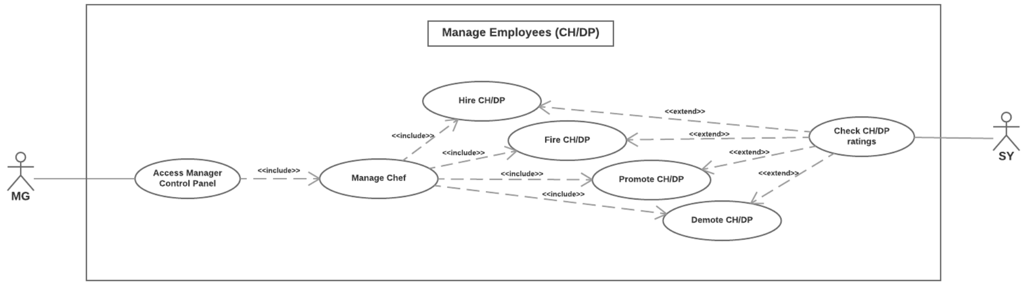
1. User logs in as MG
2. MG accesses MG-CP
3. MG views unprocessed feedback provided by RC
4. MG confirms and accepts feedback
5. SY updates the average rating of the restaurant or menu item

Placing a warning

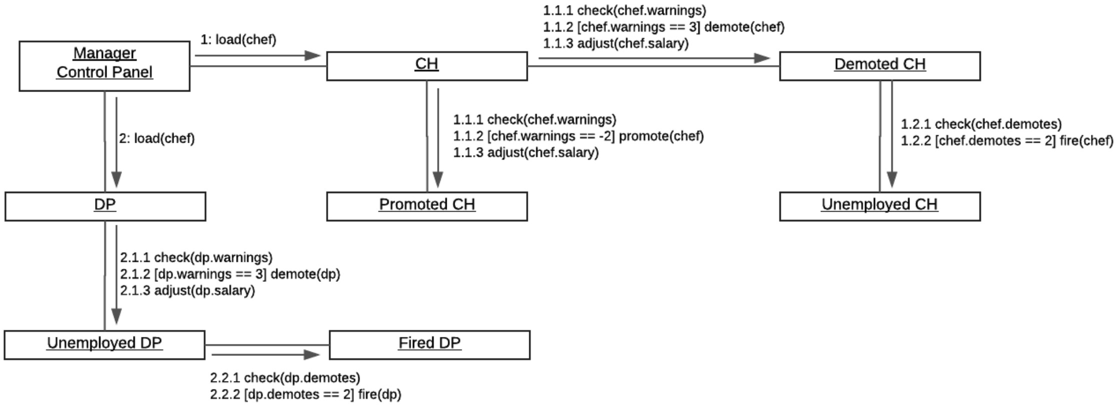
1. User logs in as MG
2. MG accesses MG-CP
3. MG views unprocessed feedback provided by RC
4. MG flags RC for inappropriate feedback

2.2.7 Managing employees

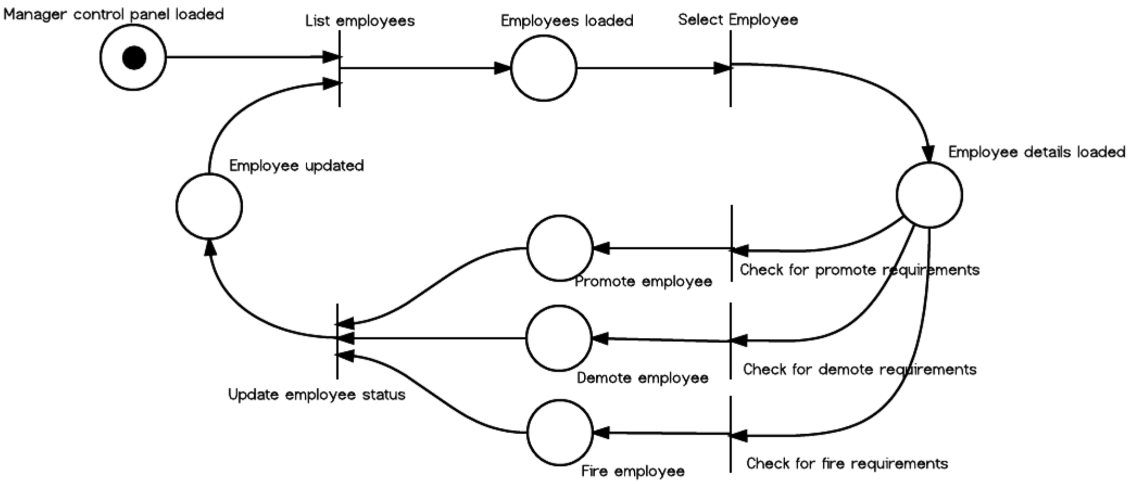
Use case diagram:



Collaboration diagram:


Manager Control Panel
1: load(chef)
CH
1.1.1 check(chef.warnings)
1.1.2 [chef.warnings == 3] demote(chef)
1.1.3 adjust(chef.salary)
Demoted CH
2: load(chef)
1.1.1 check(chef.warnings)
1.1.2 [chef.warnings == -2] promote(chef)
1.1.3 adjust(chef.salary)
1.2.1 check(chef.demotes)
1.2.2 [chef.demotes == 2] fire(chef)
DP
Promoted CH
Unemployed CH
2.1.1 check(dp.warnings)
2.1.2 [dp.warnings == 3] demote(dp)
2.1.3 adjust(dp.salary)
Unemployed DP
Fired DP
2.2.1 check(dp.demotes)
2.2.2 [dp.demotes == 2] fire(dp)

Petri-net:



Description: MG is granted special access through the MG-CP (manager control panel). This control panel allows MG to manage the employees of the restaurant (CH and DP).

Details:

Hiring

1. User logs in as MG
2. MG accesses MG-CP
3. MG views CH or DP operating at current restaurant
4. MG hires CH or DP

Firing

1. Users logs in as MG
2. MG accesses MG-CP
3. MG views a CH or a DP
4. MG prepares to fire CH or DP
5. SY checks for ratings of CH or DP and approves of fire
6. CH or DP is de-registered from current restaurant

Promoting

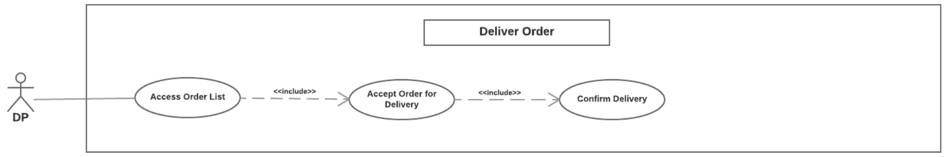
1. User logs in as MG
2. MG accesses MG-CP
3. MG views a CH or a DP
4. MG prepares to promote CH or DP
5. SY checks for ratings of CH or DP and approves the promotion
6. CH or DP is promoted
7. SY adjusts salary of selected CH or DP

Promoting

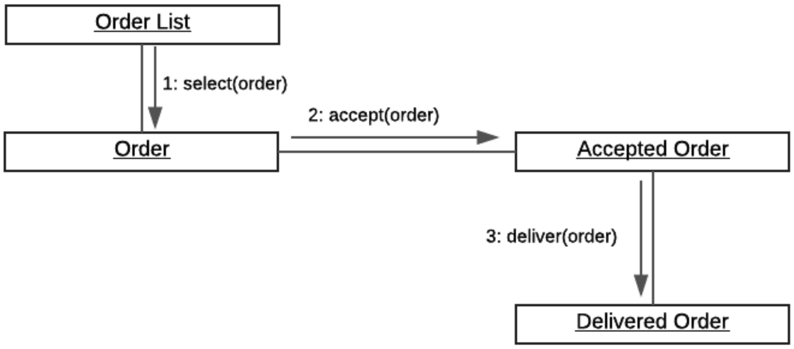
1. User logs in as MG
2. MG accesses MG-CP
3. MG views a CH or a DP
4. MG prepares to promote CH or DP
5. SY checks for ratings of CH or DP and approves the demotion
6. CH or DP is demoted
7. SY adjusts salary of selected CH or DP

2.2.8 Delivering orders

Use case diagram:



Collaboration diagram:



Petri-net: **(See 2.2.3)**

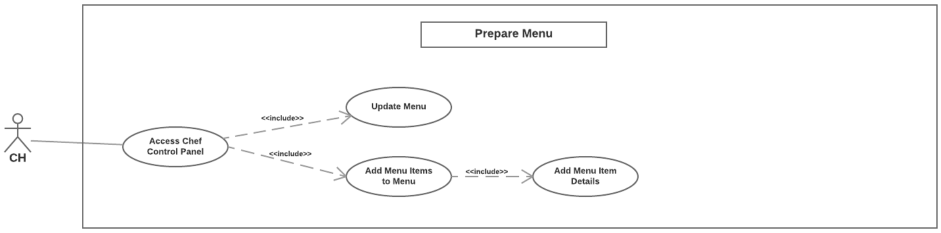
Description: Delivery personnel (DP) has access to a list of orders placed by RC and proceeds to delivery to the RC location.

Details:

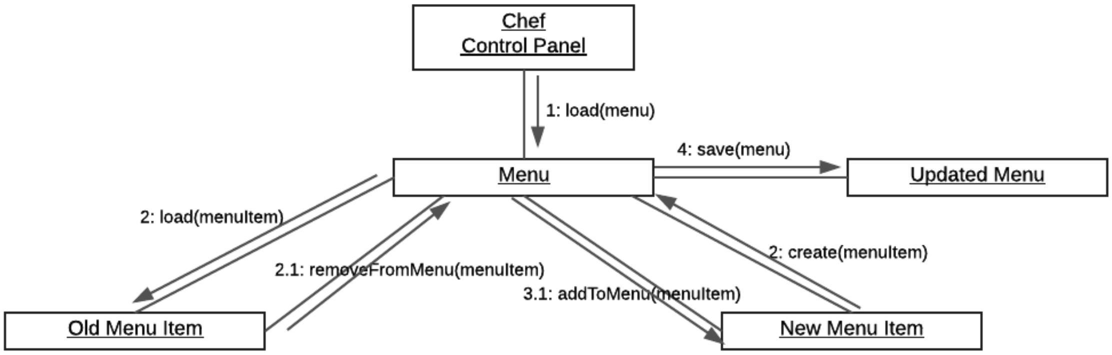
1. User logs in as DP
2. DP views list of orders for current restaurant
3. DP accepts an order to deliver
4. DP delivers and confirms that the order has been delivered

2.2.9 Preparing the menu

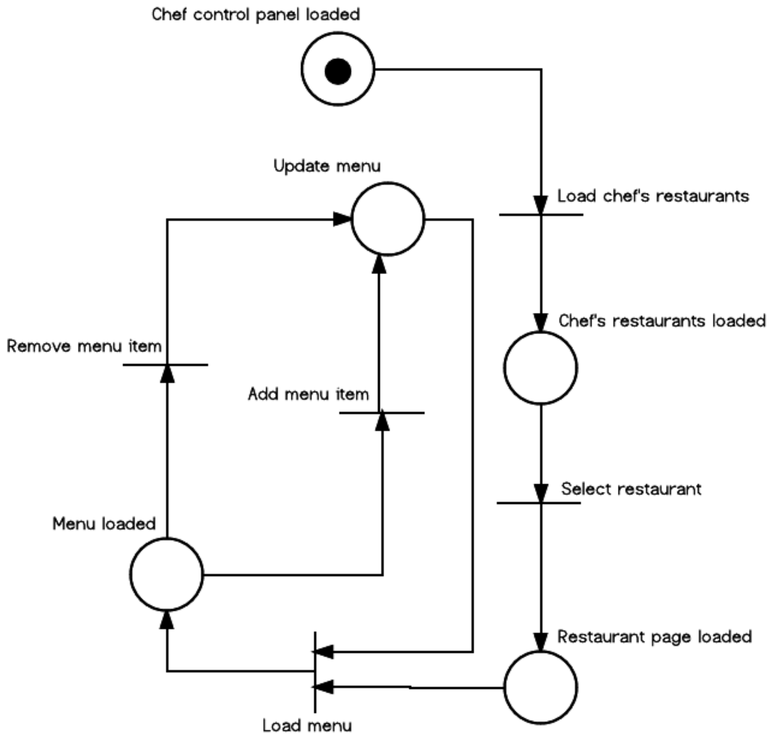
Use case diagram:



Collaboration diagram:



Petri-net:

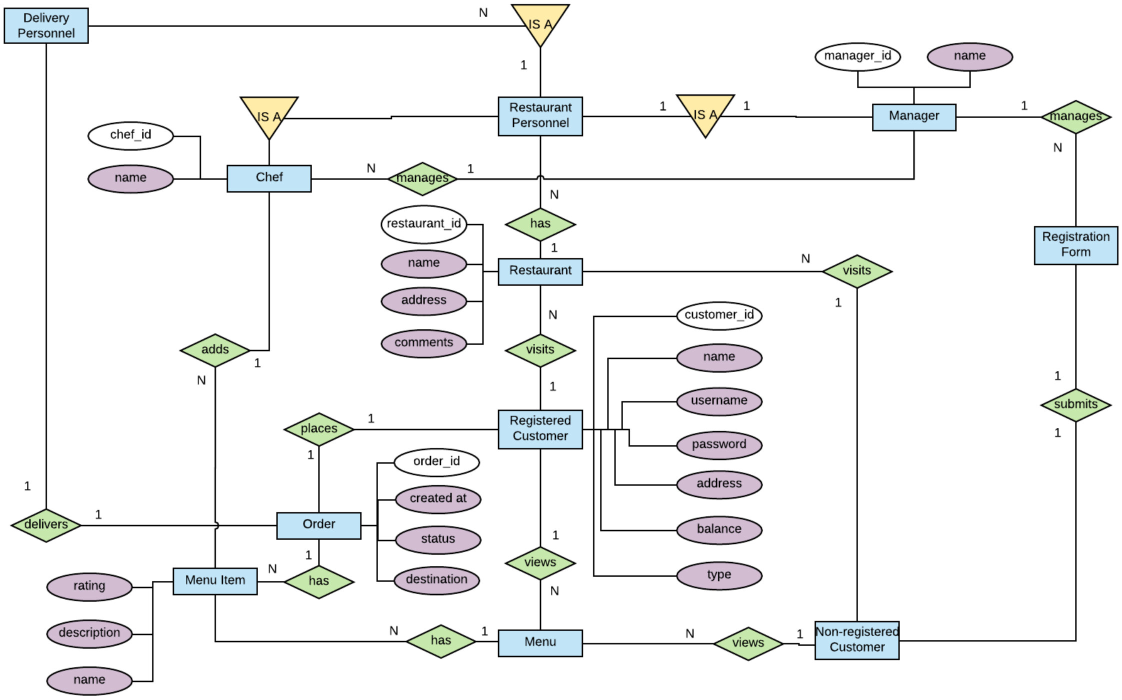


Description: Chefs (CH) have access to a special chef control panel (CH-CP), which allows the CH to alter the menu.
Details:

1. User logs in as CH
2. CH accesses CH-CP
3. CH views menu of current restaurant
4. CH adds or removes menu items
5. CH updates or saves the new menu

3. Entity Relationships

3.1 E-R Diagram



4. Detailed Design

4.1 Pseudo-code

This section provides pseudo-code for major components of the ChefNODE system. These modules are categorized by general processes such as registrations (and de-registrations), ordering, and basic UI sequences done by the system. The functions are represented by underscore-separated phrases followed by parameters within parenthesis (inputs) followed by a set of procedures within brackets {outputs}.

4.1.1 Registration Processing

```
submit_register_form {
  Initialize user_info hash (name_field:, password_field:)
  Initialize name, password
  Input valid name
  Input valid password
  Add user_info to RC-waitlist database
}

register_customer(user_info) {
  If user_info is denied by MG
    Return ERROR_INVALID_USER
  Initialize empty RC
  RC.name = user_info.name
  RC.password = user_info.password
  Add RC to RC-entity database
}
deregister_customer(user) - Remove user from RC database
```

4.1.2 Order processing

```
place_order(cart_items[], user, restaurant {
  Initialize order_sum to zero
  Initialize empty order
  For each item in cart_items
    Add item.price to order_sum
  If user is a VIP
```

```
    Reduce order_sum to 90%
  Update UI

  Add cart_items and user_id to order

  If user balance <= order_sum
    Add order to restaurant.orders
    Go to confirm order page
}

confirm_order(order, restaurant) - Add order to restaurant.orders

receive_order(order, restaurant) - Remove order from restaurant.orders

update_ratings(order, restaurant) {
  For each item in order.items
    Initialize chef to item.chef
    Update item rating
    update_employee_rating(chef, item.rating)
}
```

4.1.3 Account management

```
deposit(user) {
  Initialize balance to user.balance
  Initialize credit card information
  Input credit card information
  Input deposit amount
  If valid credit card
    Transfer deposit amount
}

update_address(user) - Update user address
```

4.1.4 Restaurant management

```
place_rc_warninig(feedback) {
  Feedback.user.warnings++
  If warnings > 3
    deregister_user(feedback.user)
}

update_menu(restaurant, menu_items) {
  restaurant.menu_items is menu_items
  Update restaurant menu_items database

update_employee_rating(employee, rating) {
  Let employee be CH or DP
  If rating is < 3
    employee.flags--
  If rating is > 3
    employees.flags++
  If employees.flags < 3
    employees.rating--
  If employees.flags > 3
    employees.rating++
  If employees.rating = 5
    employees.salary++
  If employees.rating = 0
    employees.salary--
    employees.demotes++
    If employees.demotes = 2
      Remove employees from restaurant.employees
}
```

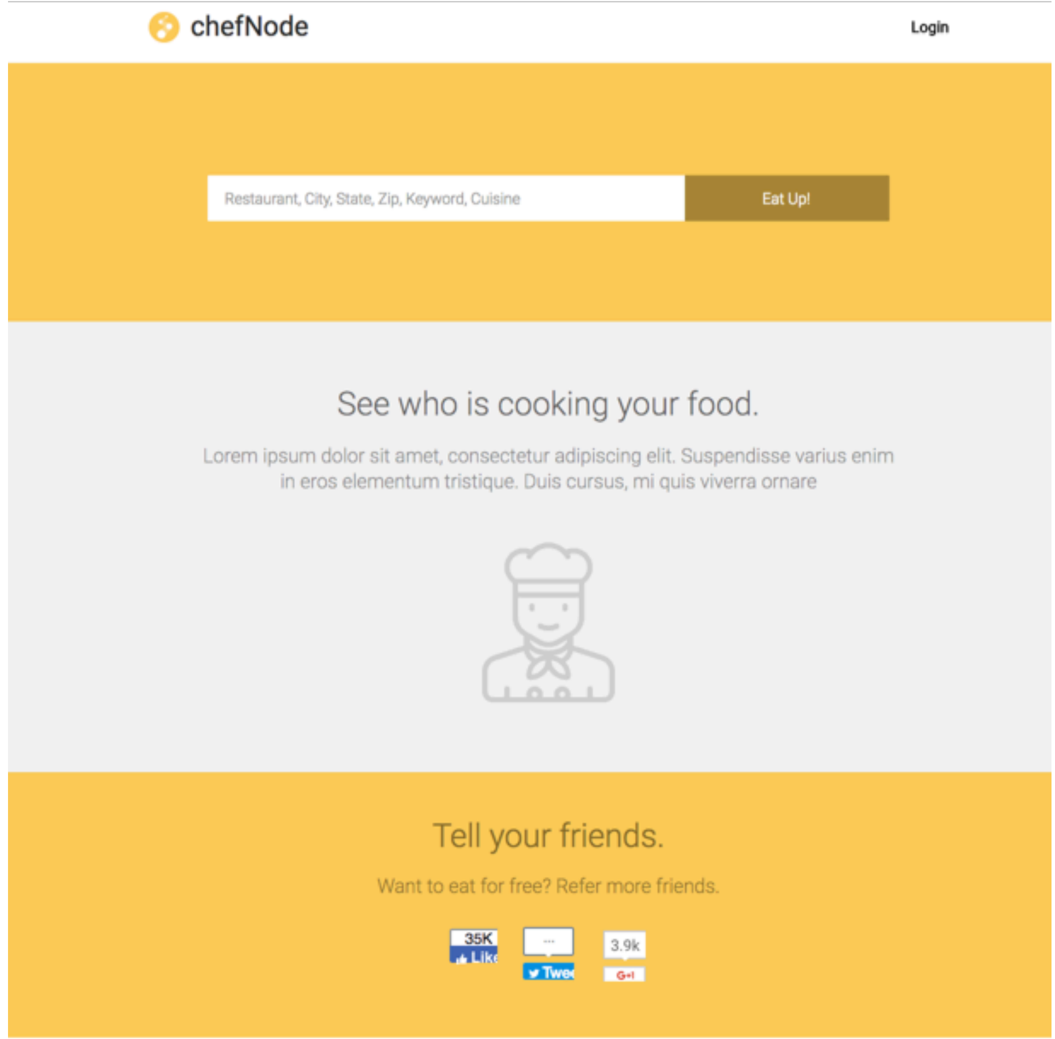
4.1.5 GUI Sequences

```
load_restaurant_page(restaurant) {
  Load html w
  Get restaurant from database
  load_menu()
  load_details()
}

load_menu(restaurant) - Load menu items of restaurant
load_details(restaurant) - Load location, reviews, description of restaurant

load_menu(user) {
  If user is NC
    Show top_dishes[5]
  Else if user is RC
    Show user.top_dishes[5]
  Show rest of menu
}
```

5. System Screens

5.1 Homepage screens

5.2.1 Homepage (NC)



See who is cooking your food.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse varius enim in eros elementum tristique. Duis cursus, mi quis viverra ornare

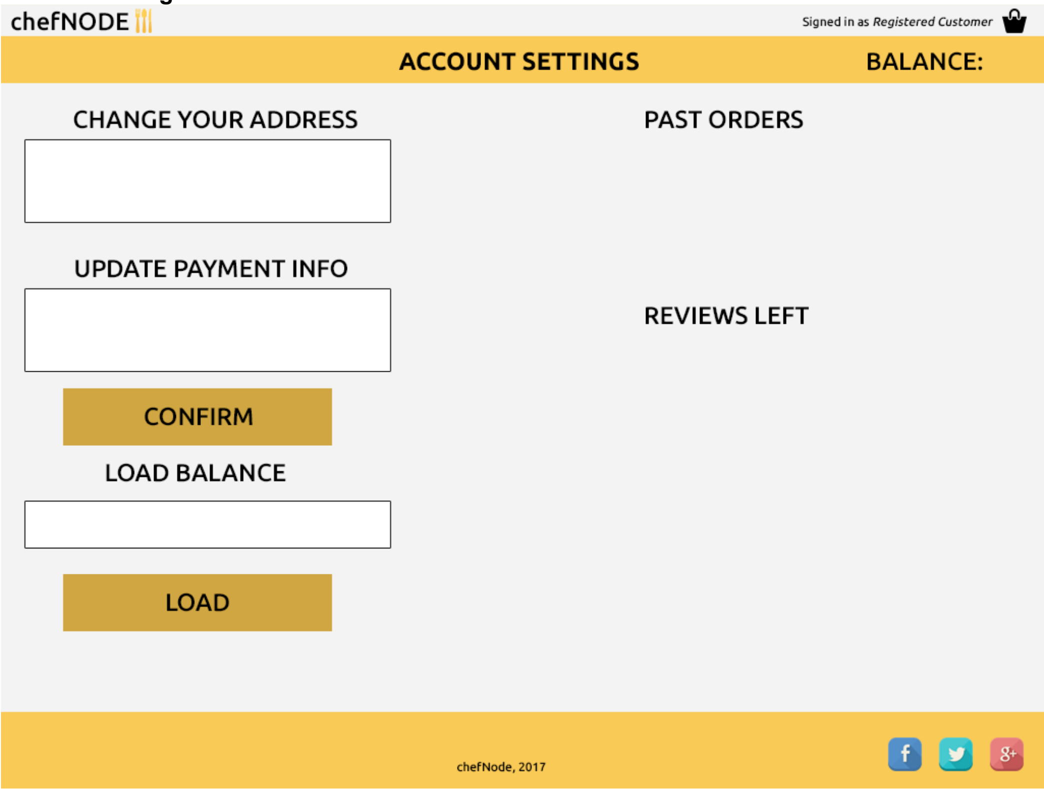
Tell your friends.

Want to eat for free? Refer more friends.

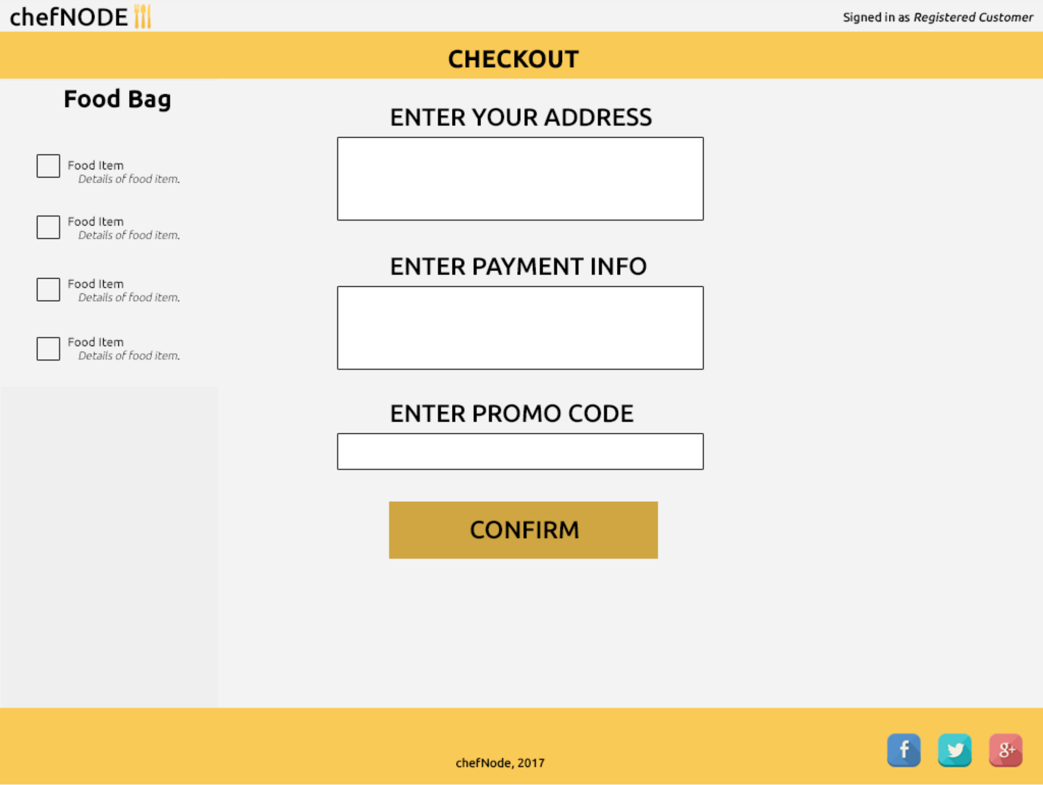
5.2.1 Homepage (RC)

5.2 Registered users

5.2.1 Account settings



5.2.2 Checkout



chefNODE

Signed in as *Registered Customer*

CHECKOUT

Food Bag

□ Food Item
Details of food item.

□ Food Item
Details of food item.

□ Food Item
Details of food item.

□ Food Item
Details of food item.

ENTER YOUR ADDRESS

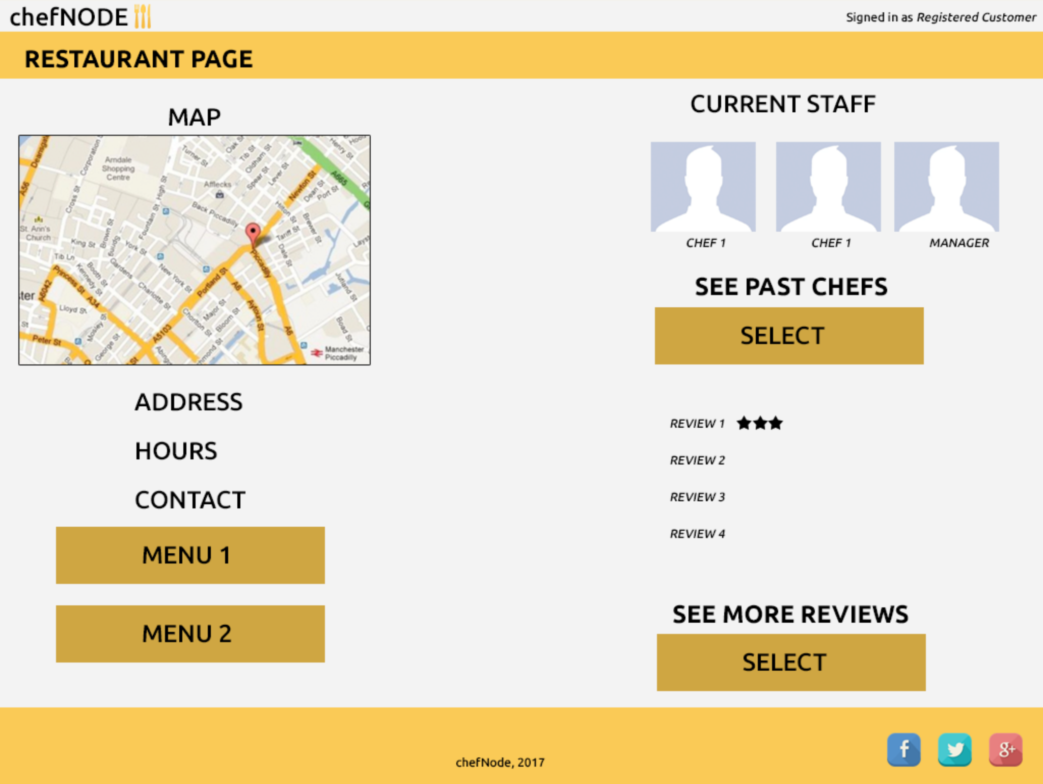
ENTER PAYMENT INFO

ENTER PROMO CODE

chefNode, 2017

5.3 Restaurant

5.3.1 Restaurant page



chefNODE

Signed in as *Registered Customer*

RESTAURANT PAGE

MAP

CURRENT STAFF

CHEF 1 *CHEF 1* *MANAGER*

SEE PAST CHEFS

ADDRESS

HOURS

CONTACT

REVIEW 1 ★★★

REVIEW 2

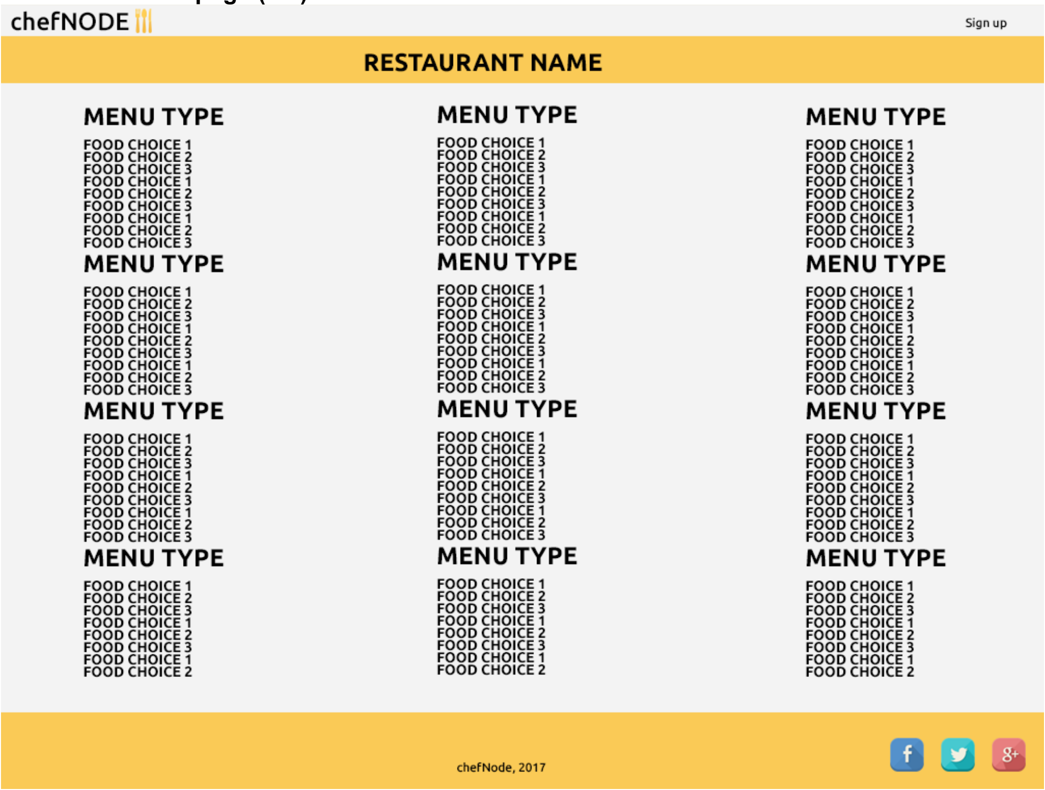
REVIEW 3

REVIEW 4

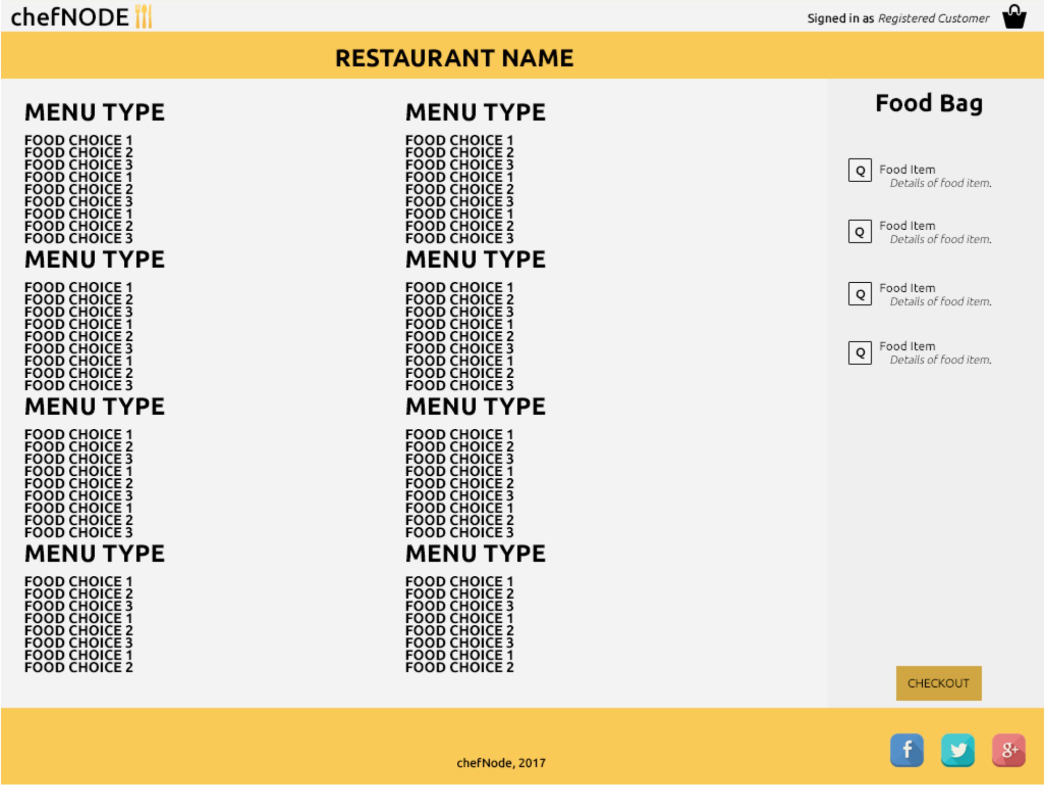
SEE MORE REVIEWS

chefNode, 2017

5.3.2 Restaurant menu page (NC)

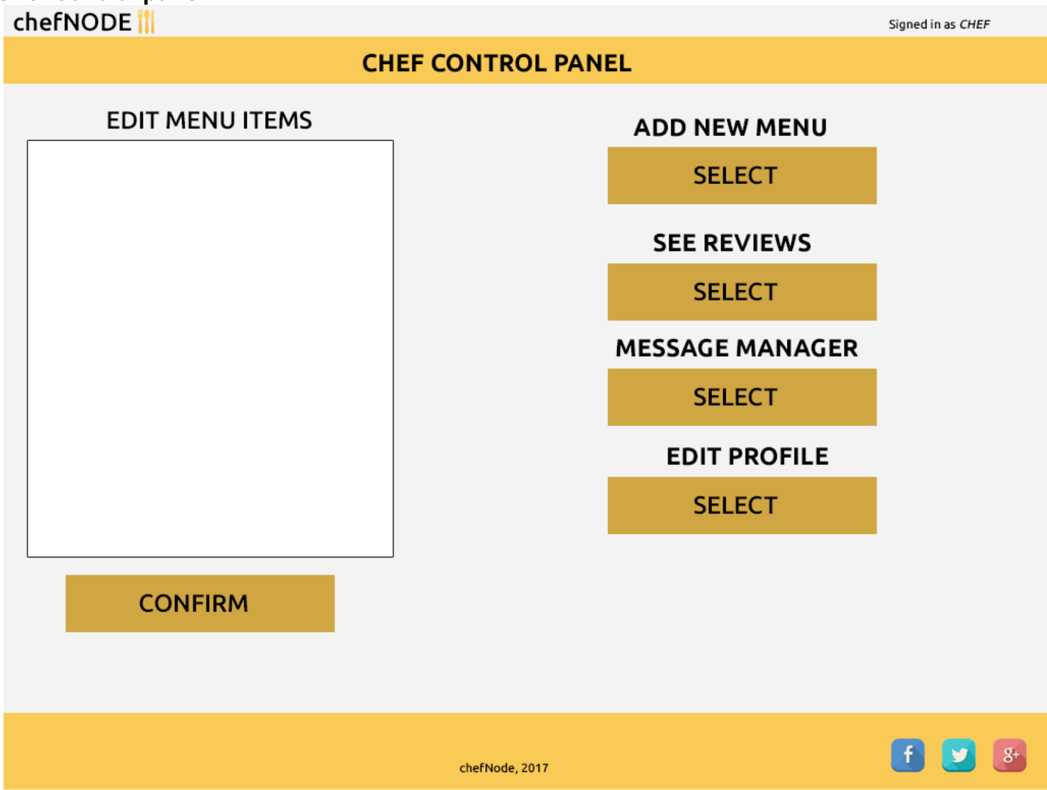


5.3.2 Restaurant menu page (RC)

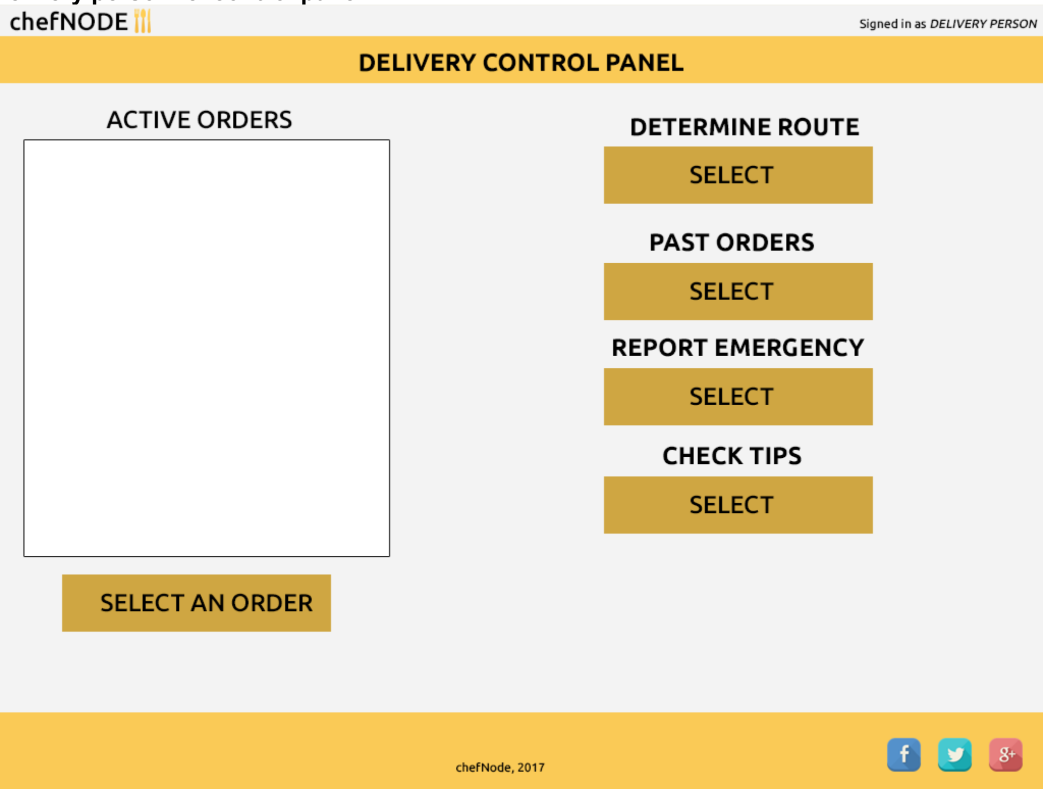


5.4 Restaurant personnel

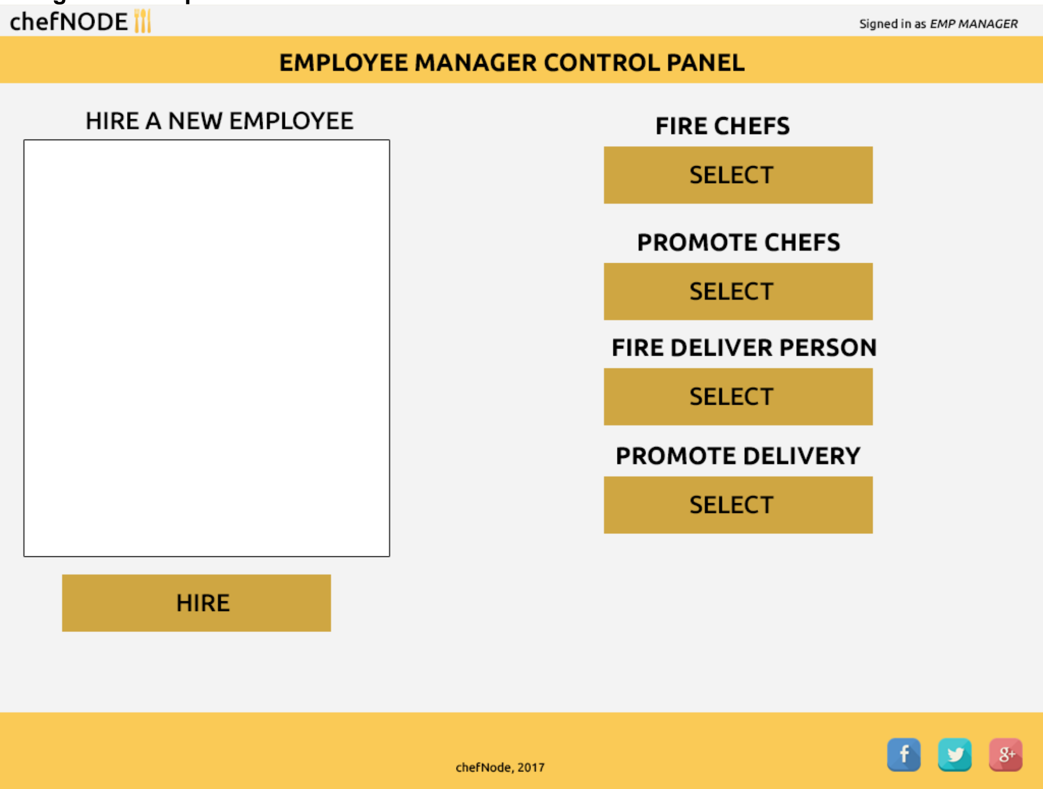
5.4.1 Chef control panel



5.4.2 Delivery personnel control panel

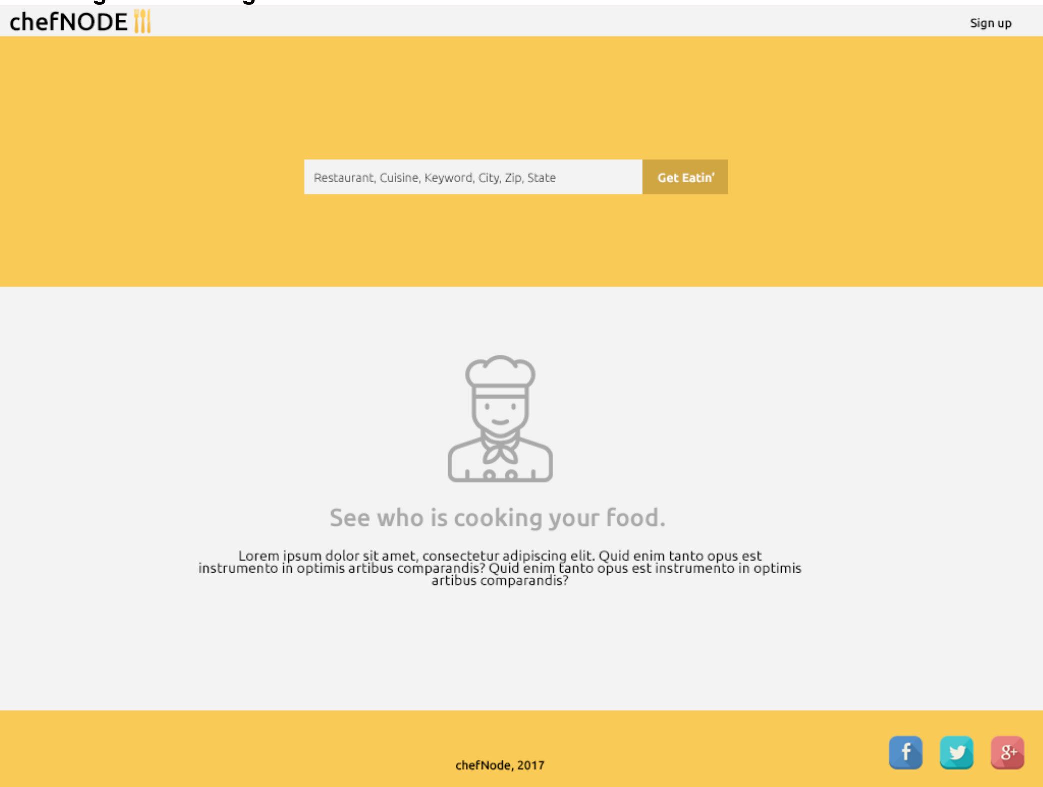


5.4.3 Manager control panel



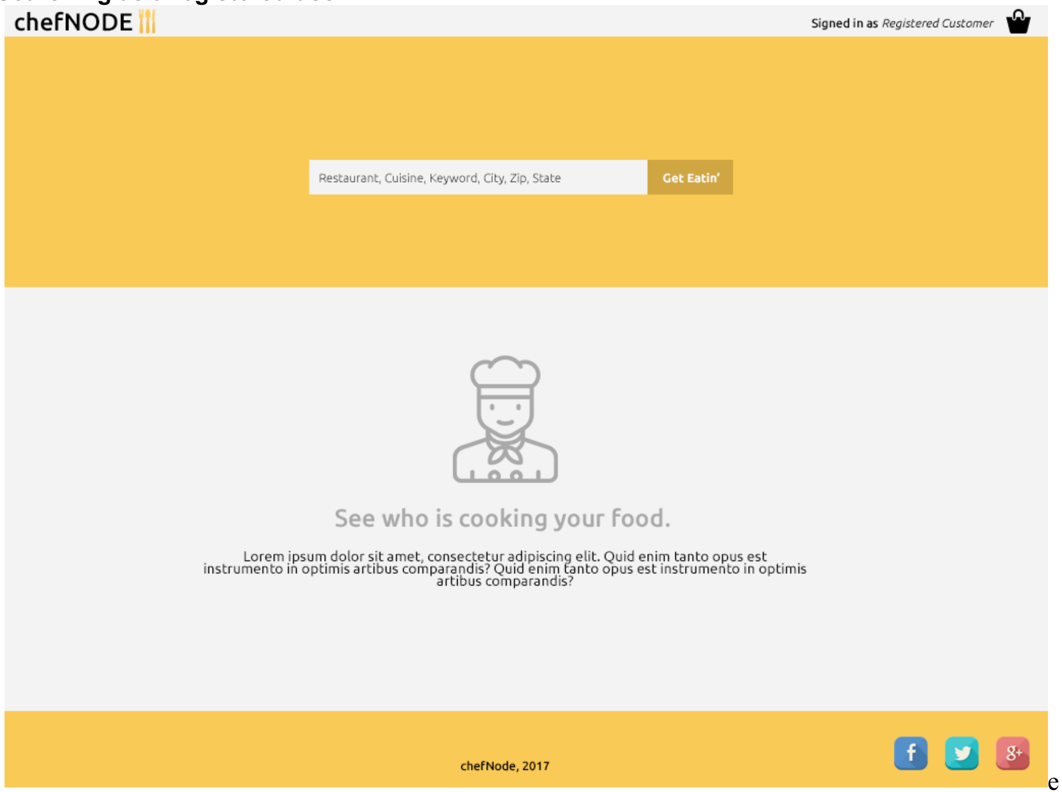
5.5 Searching and browsing

5.5.1 Searching as a non-registered user


chefNODE
Sign up
Restaurant, Cuisine, Keyword, City, Zip, State
Get Eatin'
See who is cooking your food.
chefNode, 2017

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quid enim tanto opus est instrumento in optimis artibus comparandis? Quid enim tanto opus est instrumento in optimis artibus comparandis?

5.5.2 Searching as a registered user



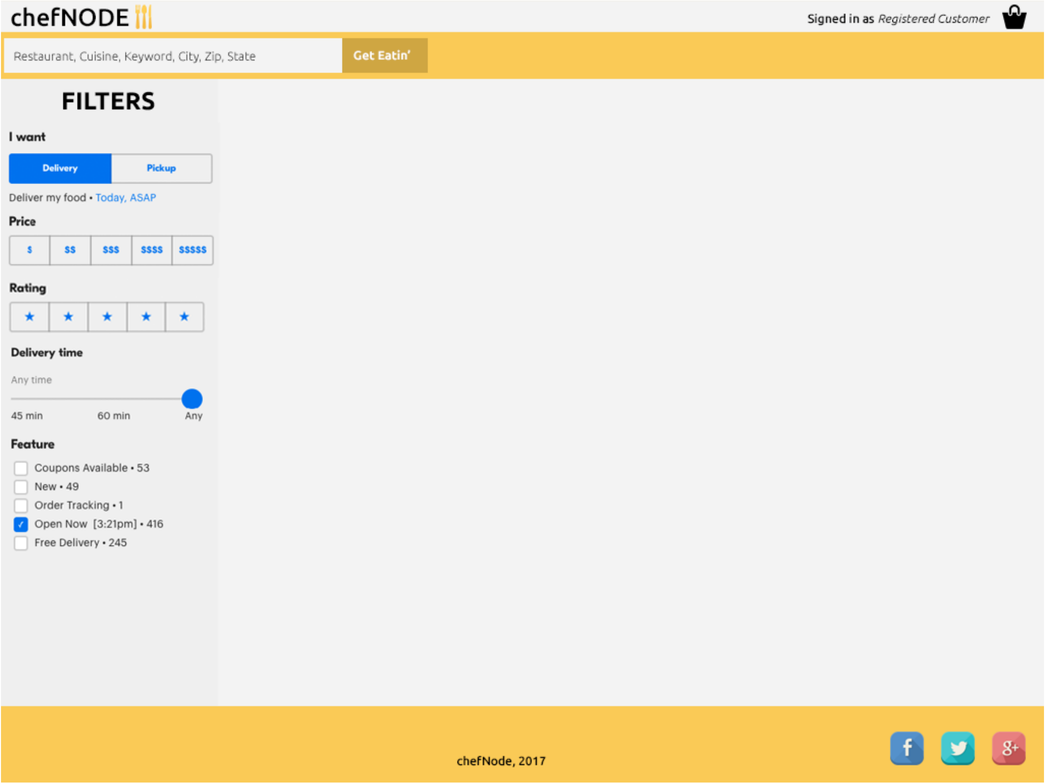
chefNODE
Signed in as *Registered Customer*
Restaurant, Cuisine, Keyword, City, Zip, State
Get Eatin'

See who is cooking your food.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quid enim tanto opus est instrumento in optimis artibus comparandis? Quid enim tanto opus est instrumento in optimis artibus comparandis?

e

5.5.3 Search results



chefNODE
Signed in as *Registered Customer*
Restaurant, Cuisine, Keyword, City, Zip, State
Get Eatin'
FILTERS

I want

● Delivery ○ Pickup

Deliver my food • Today, ASAP

Price

$ | $$ | $$$ | $$$$ | $$$$$

Rating

★ ★ ★ ★ ★

Delivery time

Any time

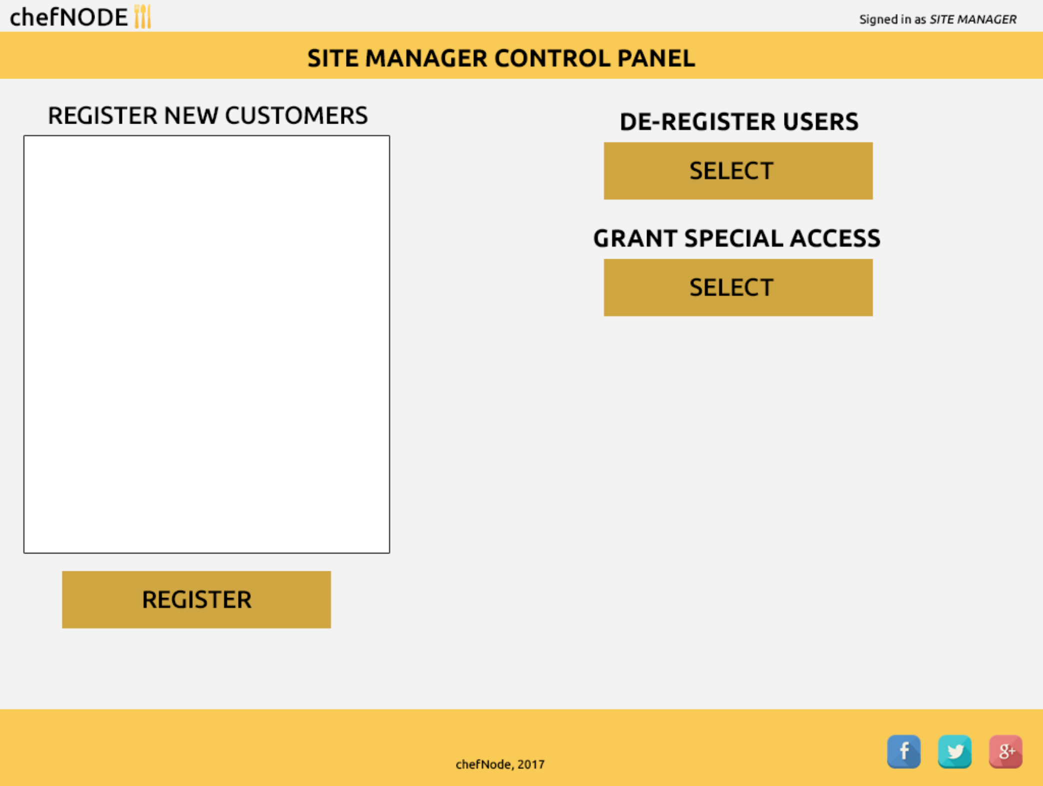
45 min | 60 min | Any ●

Feature

□ Coupons Available • 53
□ New • 49
□ Order Tracking • 1
☑ Open Now [3:21pm] • 416
□ Free Delivery • 245

5.6 System

5.6.1 Administrator control panel


chefNODE
Signed in as SITE MANAGER
SITE MANAGER CONTROL PANEL
REGISTER NEW CUSTOMERS
DE-REGISTER USERS
SELECT
GRANT SPECIAL ACCESS
SELECT
REGISTER
chefNode, 2017