Software Requirements Specification

For

Software Specification and Design

01219243

Sahadporn Charnlertlakha
6210545611

Introduction

Purpose

This specification is a starting point for the design for project FoodNow 1.0. As I started working on the project, there will be more obstructions, detours and more. Thus the project will have changes, improvements, refinements, and adjustments in the future. There will be an attempt to keep this specification up to date as possible. By no means should this specification be the final version.

Document conventions

This specification uses standard Times New Roman fonts.
The asterisks is used for remarks and important information.

Intended audience

This specification is meant to be read by developers, project managers, documentation writers, and any interested individual.

Project scope

The scope of the first version is to meet minimum viable requirements for basic food delivering service platform. The primary goal is to release a fully functional version of FoodNow in 3 months to match the business goal; which is to capitalize on the growing demand for food delivery service during this time of pandemic.

Contact information

Contact the team through FoodNow community , FoodNow@gmail.com.

References

- This specification took influence from the IEEE standard specification.
- Best Practices for Designing Amazon API Gateway Private APIs and Private Integration - AWS Whitepapers
- AWS EC2
- AWS Relational Database Service
- AWS Virtual Private Cloud
- Postman | The Collaboration Platform for API Development for API Builder.
- https://www.figma.com for interface design tool.

Overall Description

Product perspective

Project “FoodNow” is a project developed by the FoodNow community during this time of pandemic. FoodNow will allow potential customers to have food delivered to their doorstep in a short period of time. By placing an order through this service and notify local restaurants. This service will then assign the order to one of the employees to do the delivery. Since it will cover only the local area for each customer the waiting time will be short.

Note:
*This project is intended for a small local area scale of service, not country wide service.
*The FoodNow delivery related functions such as assign orders to employees , employees updated status of delivery e.g. pick up from kitchen , delivered..etc. These functions are performed in the other project i.e. DeliverNow project which will not be part of the FoodNow project. However, we have to refer to DeliverNow in this document for end to end workflow completion.

Product functions

FoodNow Platform supports two types of user. Functionalities will be listed by user type as below.

Normal Customer

- The customer is a potential customer who wants to order food on FoodNow platform.
- The customer will register to the service and provide information such as their name and address.
  *Online payment and security improvement will be provided in the future version.
- The customer will choose a restaurant, select the desired menu and place an order.
- FoodNow will notify the target restaurant to prepare the order.
- FoodNow will notify the target restaurant on the incoming order with full details.
- The target restaurant can notify FoodNow back when the order is finished.
- Employee will send confirmation (through a separate application, DeliverNow) back to the service after their successful delivery.

Restaurant

- The interested restaurant could register to be part of FoodNow platform.
- FoodNow will provide functionalities for restaurants to post/update information about restaurants e.g. address ,type of food ,operating time , images and logo...etc
- The interested restaurant is able to add/edit menu both description , price and dish images.
- The interested restaurant is able to view and reject orders.
- The FoodNow will provide business performance analytic reports to each restaurant.

User classes and Characteristics

Customer

A customer is a potential client who uses this service to place food orders.

Restaurant

A restaurant is a merchant who uses this service as a platform to provide their food and catering services.

Operating environment

- The web UI will use React technology.
- The backend services are written in python.
- Database is MySQL.
- No software needs to be permanently installed on the client side.
- Version 1.0 is available for Windows, IOS, and android.
- The backend service will be deployed on AWS cloud using standard internet port.
- It will be deployed on EC2 instances which are installed on Ubuntu.
- AWS API Gateway is used to manage FoodNow REST API traffic from FoodNow UI.

User Documentation

N/A

Design and implementation constraints

All design and engineering decisions will follow a principle ‘Simple is better than complicated’. So the team aims to keep choices that customers must make to the minimum to give customers the smoothest experience.

This project is written in an object-oriented language with strong GUI links via a simple REST API called. The UI should be able to run on PC or any mobile devices.

The FoodNow backend is hosted on AWS so this will help to achieve seamless recovery, and cannot allow any data loss. This also enables the FoodNow backend to scale easier if demands are increasing.

Assumptions and Dependencies

FoodNow platform version 1.0 must be available for Windows, IOS, Android. In the future it may be improved to make it compatible with other systems.

It is assumed that none of the constituent system components will be implemented as embedded applications.

Assumptions were made that the environment for deployment must be capable of accepting internet networks. The target client side hardware is assumed to access FoodNow via the internet so this will consume it’s processing capability,battery life and Internet access expense.

External interface requirements

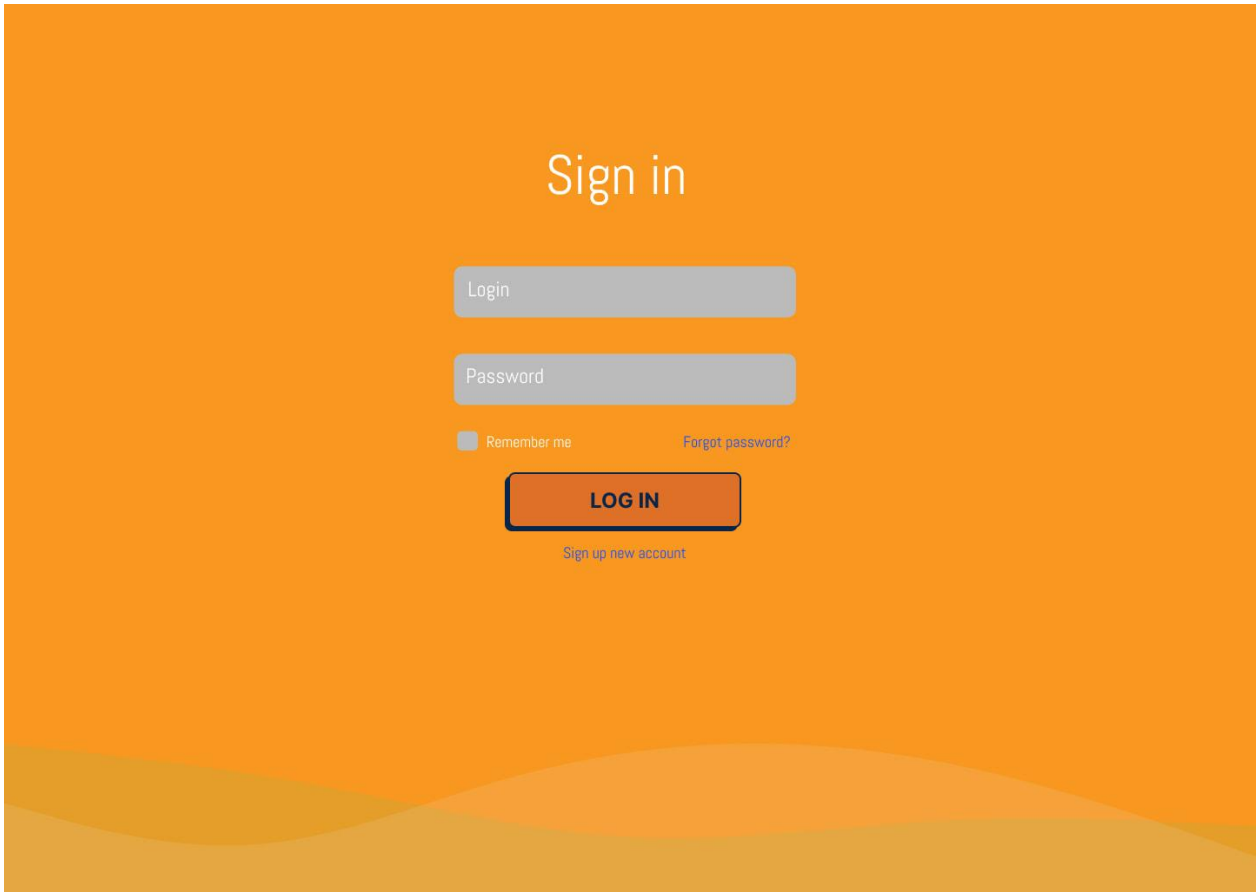
User interfaces

The user interfaces can be listed as follow:

*The first version of interface design is heavily based on the usage on the computer before mobile phones and tablets.

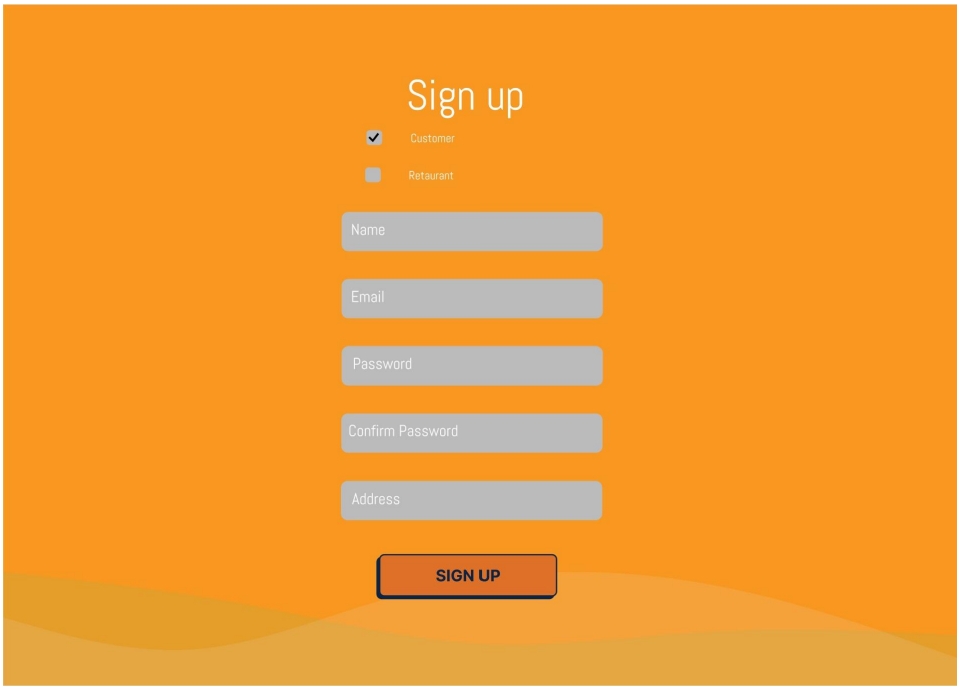
Customer UI

The Customer UI is the interface used by customers. Customers can interact with the system by using a cursor on the screen display for Desktop. For mobile, navigation tab and swipe gestures will be supported. Customers can interact with components such as items of food, menu lists, customer information setup, etc. Menu list will display similar to a list view, customers can click to access the food information inside the menu, which will also display as a list view. Additionally customers can click to evoke order settings pop up to set and reset the food information, quantity, promotion, etc.



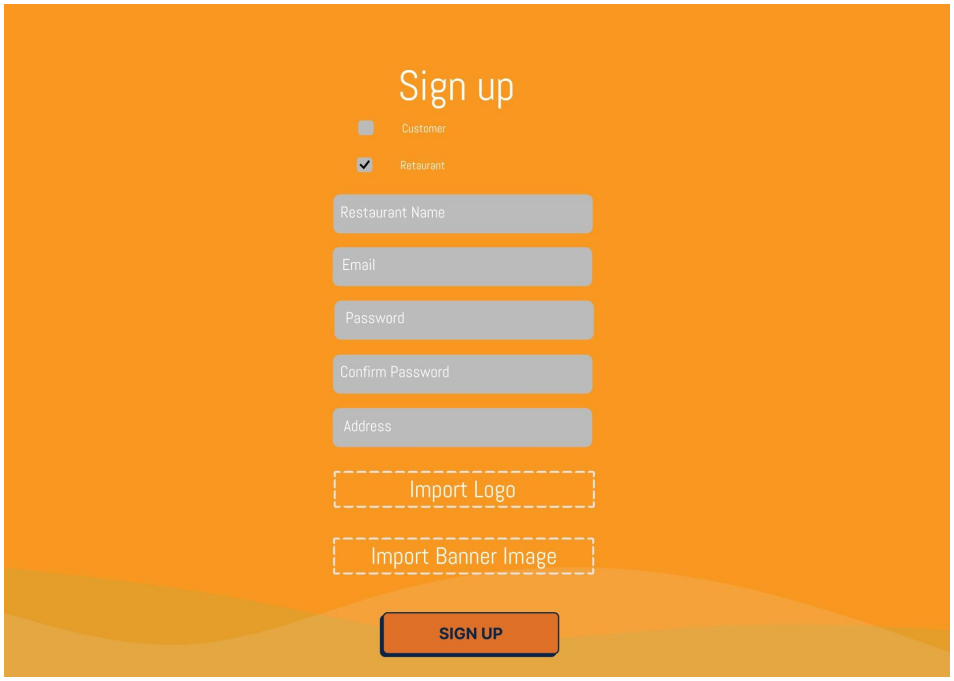
1-1: Login page

For Login on the current version there is only email available. Users must log in with email and password. Users can click ‘remember me’ to remember their account.



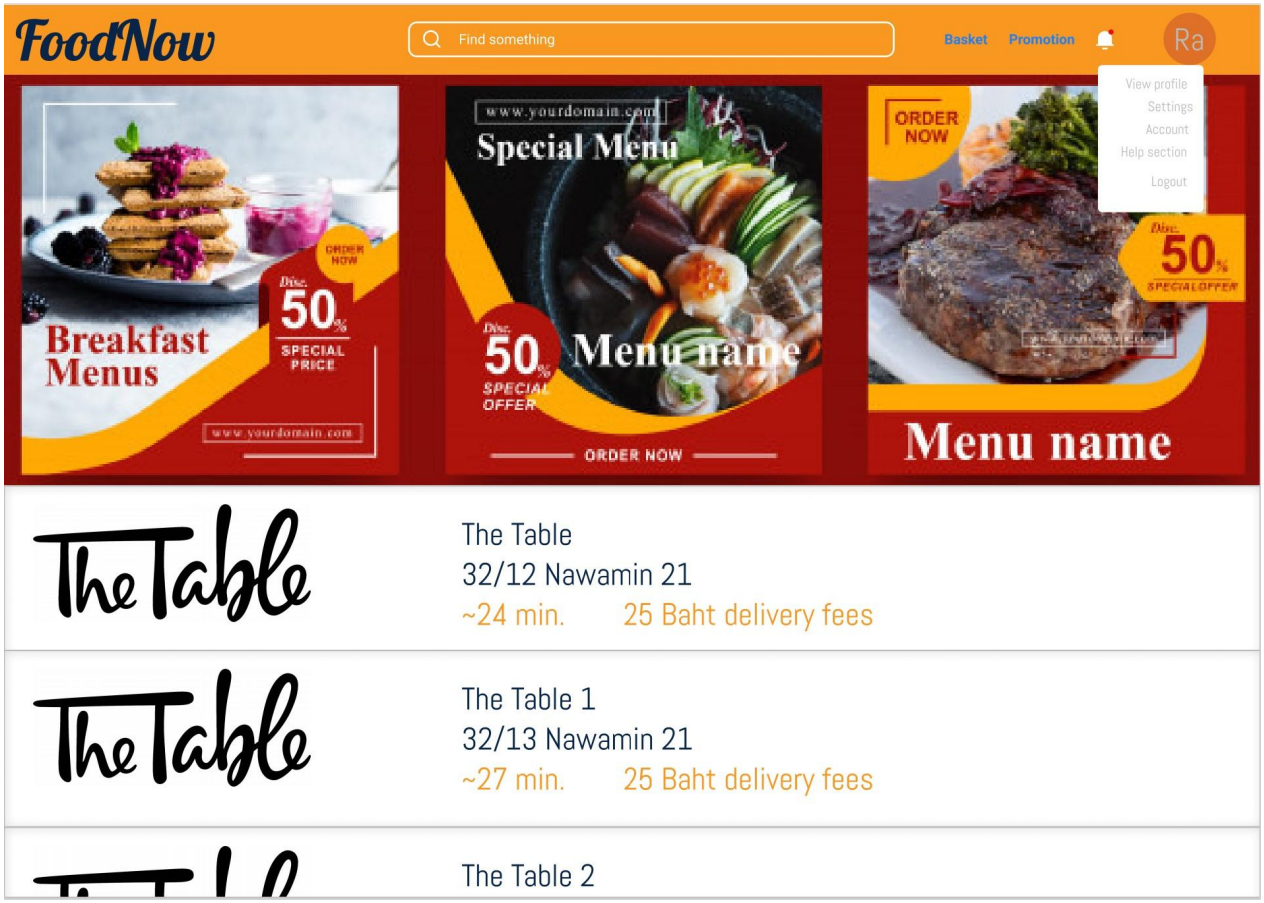
1-1-2: Sign up page

On the current version sign up, only email is available. For Customer: name, address, email, and password are required.



1-1-3 Sign up page restaurant

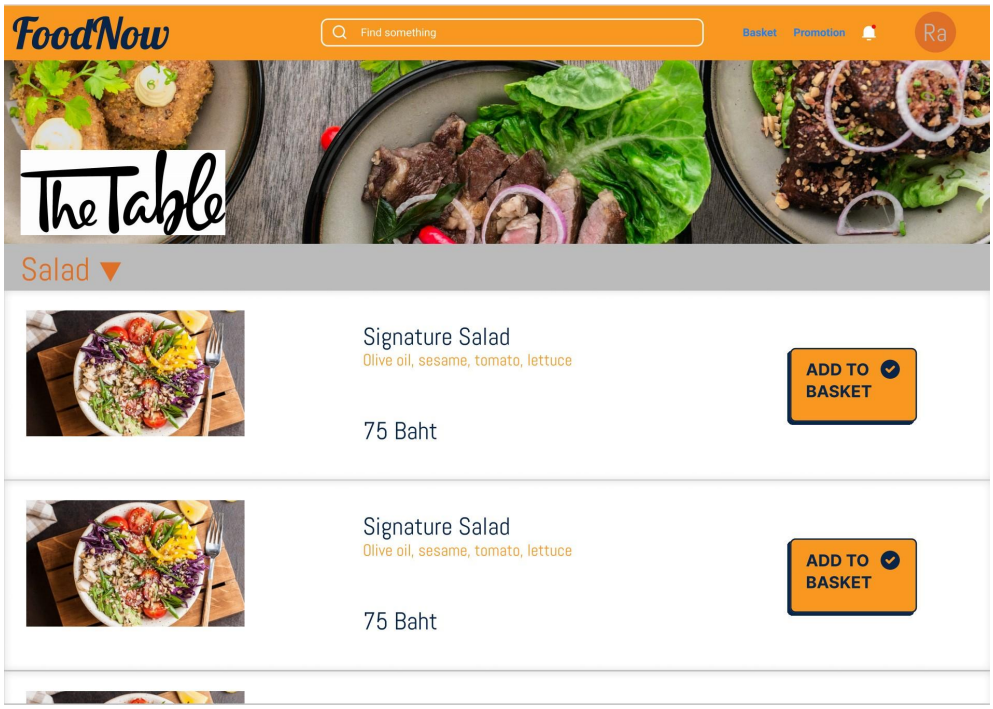
For restaurant, accounts, restaurant's address, logo and banner image are also required.



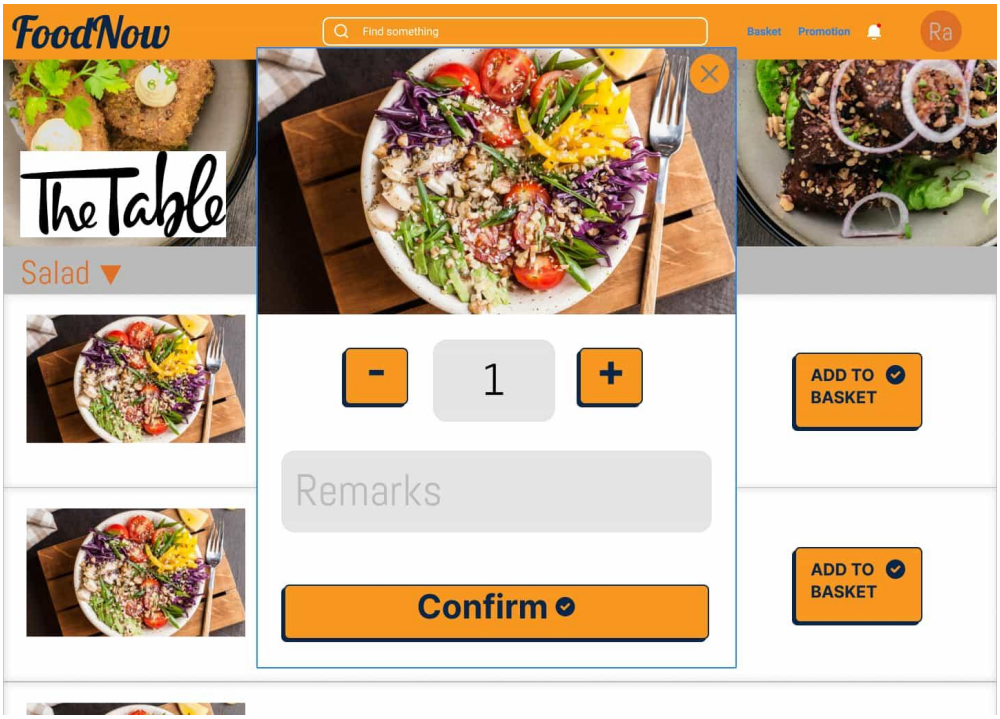
1-2: home page

Once a customer login successfully, a first UI will display all restaurants to the customer. Customers can scroll up-down and click (tab gesture on mobile) on each restaurant to view the restaurant's menu. Customers can search for restaurants by the search box on the navigation bar.

For food ordering, Customers will click (tab gesture on mobile) on the desired food item's 'Add to basket' button. A pop up will appear so customers can choose it's quantity and add any special remarks.

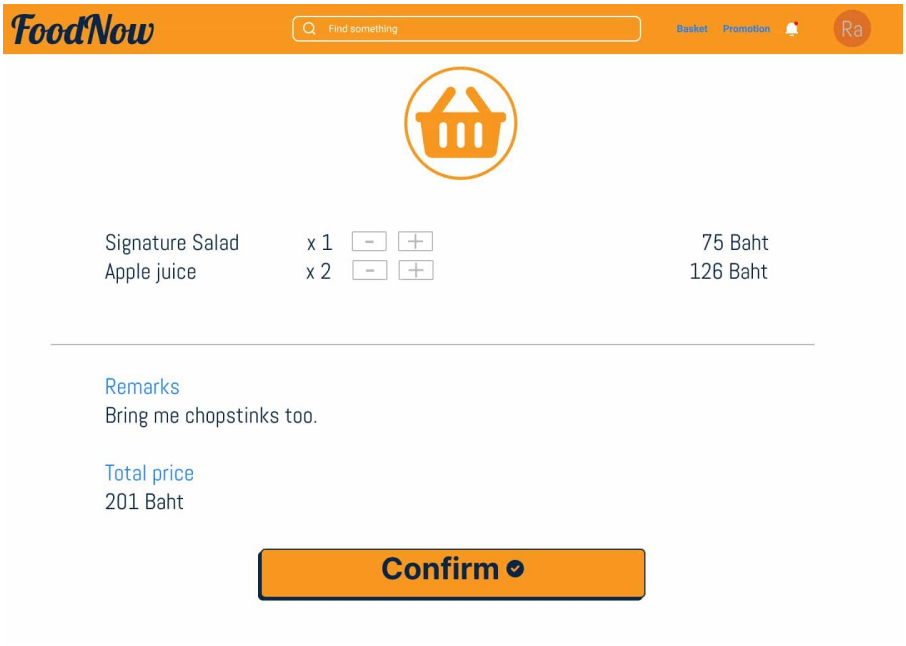


1-3: food items list view by restaurant page

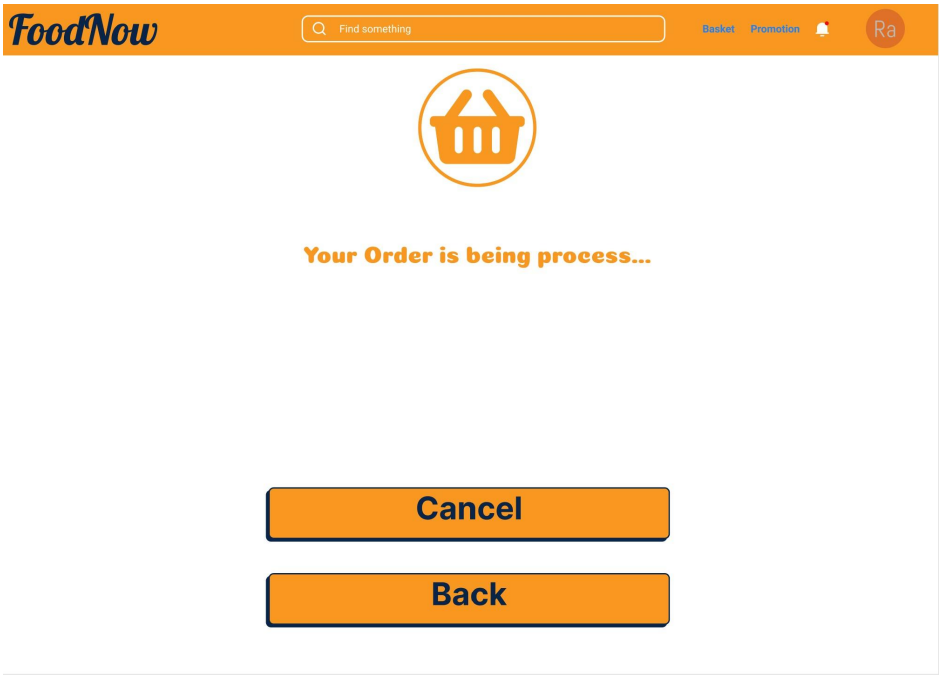


1-4: food item pop up

Customers can check their current selection by clicking on the ‘Basket’ link on the navigation bar. Basket will store the food selection that is not yet in any orders. Customers then can add or delete the food quantity again. Both price and total price are computed dynamically. After that press the ‘Confirm’ button order will be sent to the restaurant. Basket page will change for customers to track the status of their order. Customers can press cancel button to cancel their order.



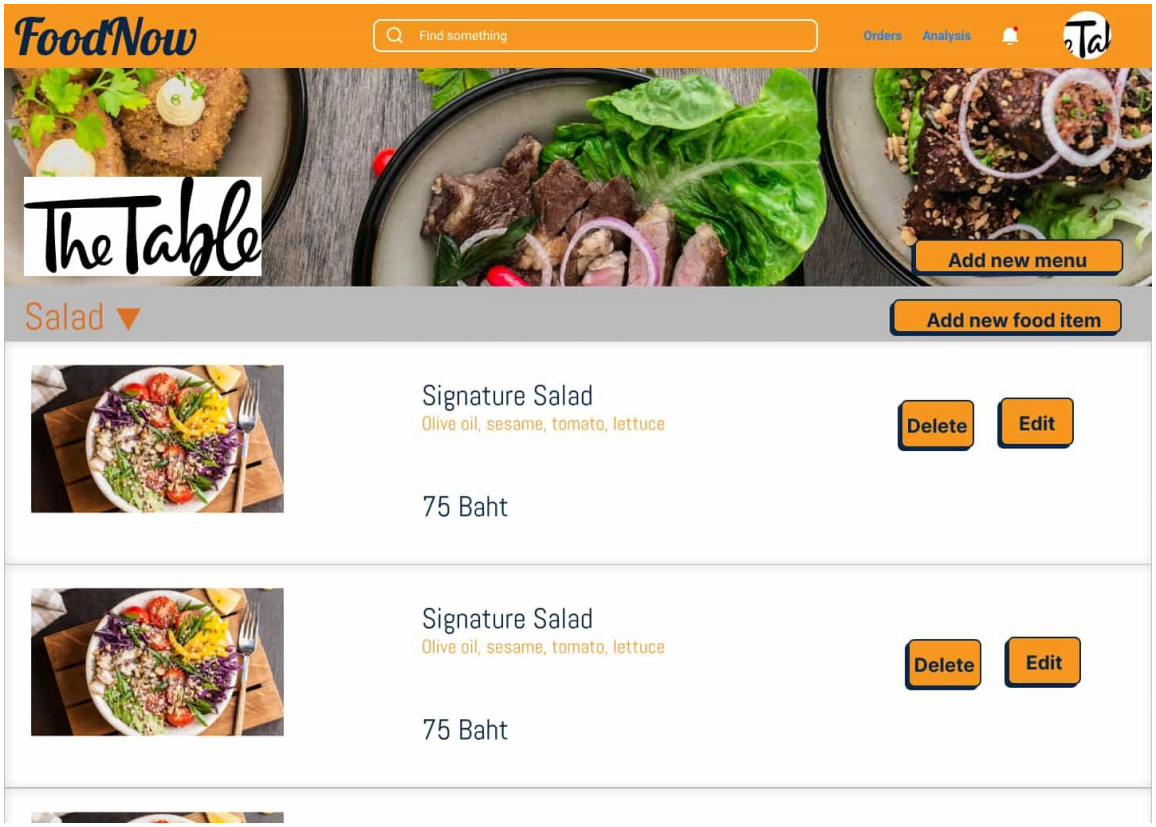
1-5: Basket page



1-6: Basket status

Restaurant UI

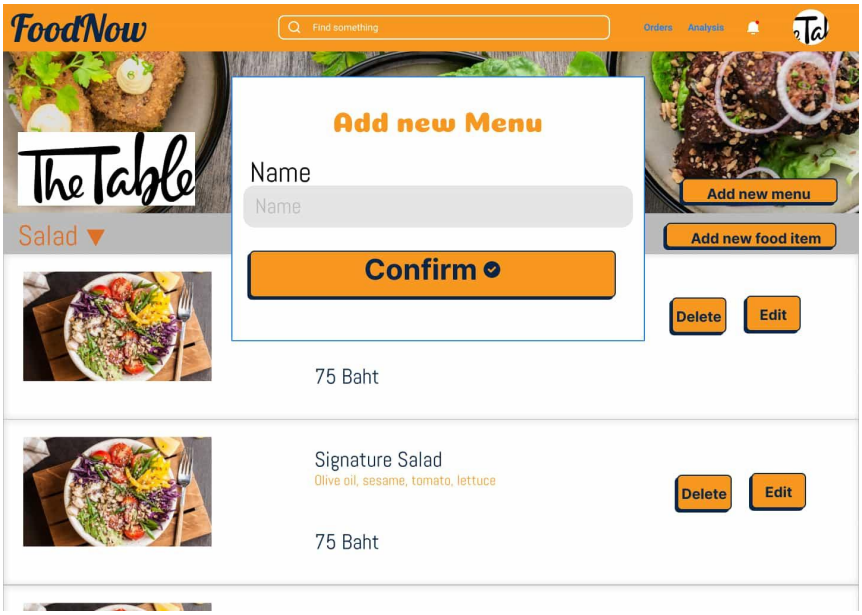
The Restaurant UI is the interface used by the restaurant administrator. The UI will display a list of menu items owned by this restaurant. Restaurants can add or remove their menus and items of food by a single click on the left button provided on the menu. A small customization on the menu style and listing maybe possible e.g. food images , price , description..etc



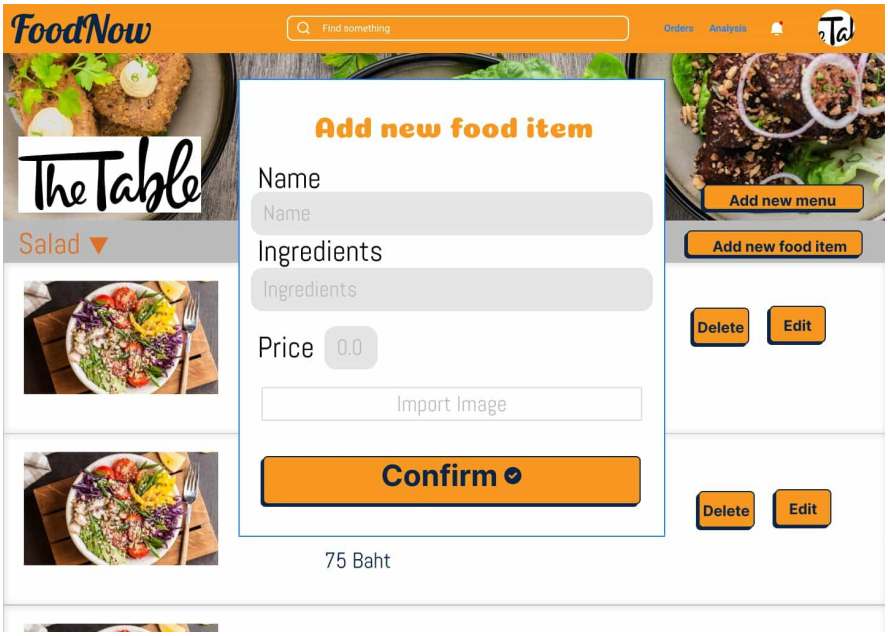
2-1: restaurant home page

Similar to the customer's UI. Users can scroll around, add menu, add food, edit food, and delete food.

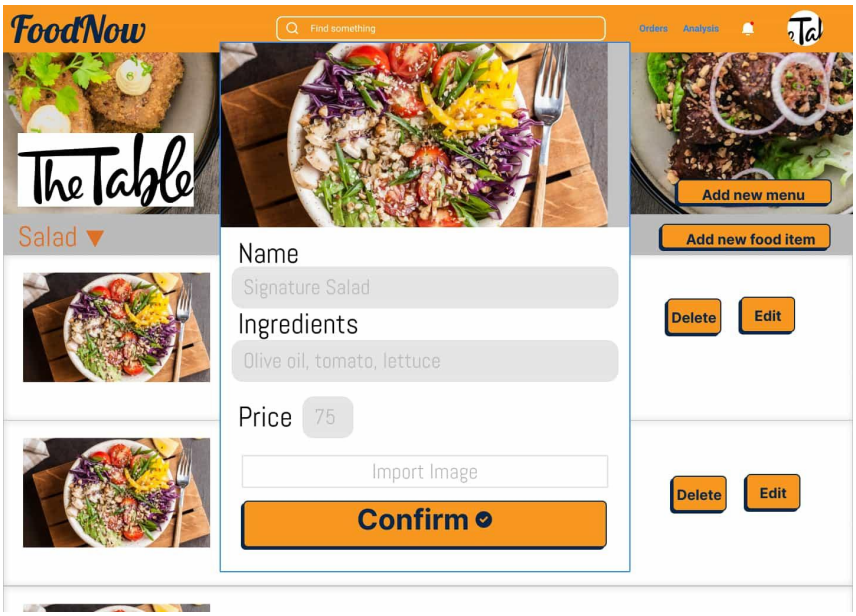
To add a new menu click ‘Add new menu’ Button on the bottom right of the banner. Pop up will appear for users to type a new menu name. Users can add food items by clicking the ‘Add new food item’ button on the right side of every menu tab. Pop up will appear for the user to type the name, ingredients, price, and image for the new food item.
Users can also edit the existing food item by clicking the ‘Edit’ button on the right side of every food item. Pop up will appear for users to edit the food item’s name, ingredients, price, and image. For deleting the food item simply click on the ‘Delete’ button on the right side of every food item. Promotion of this food item will appear here if there are any.



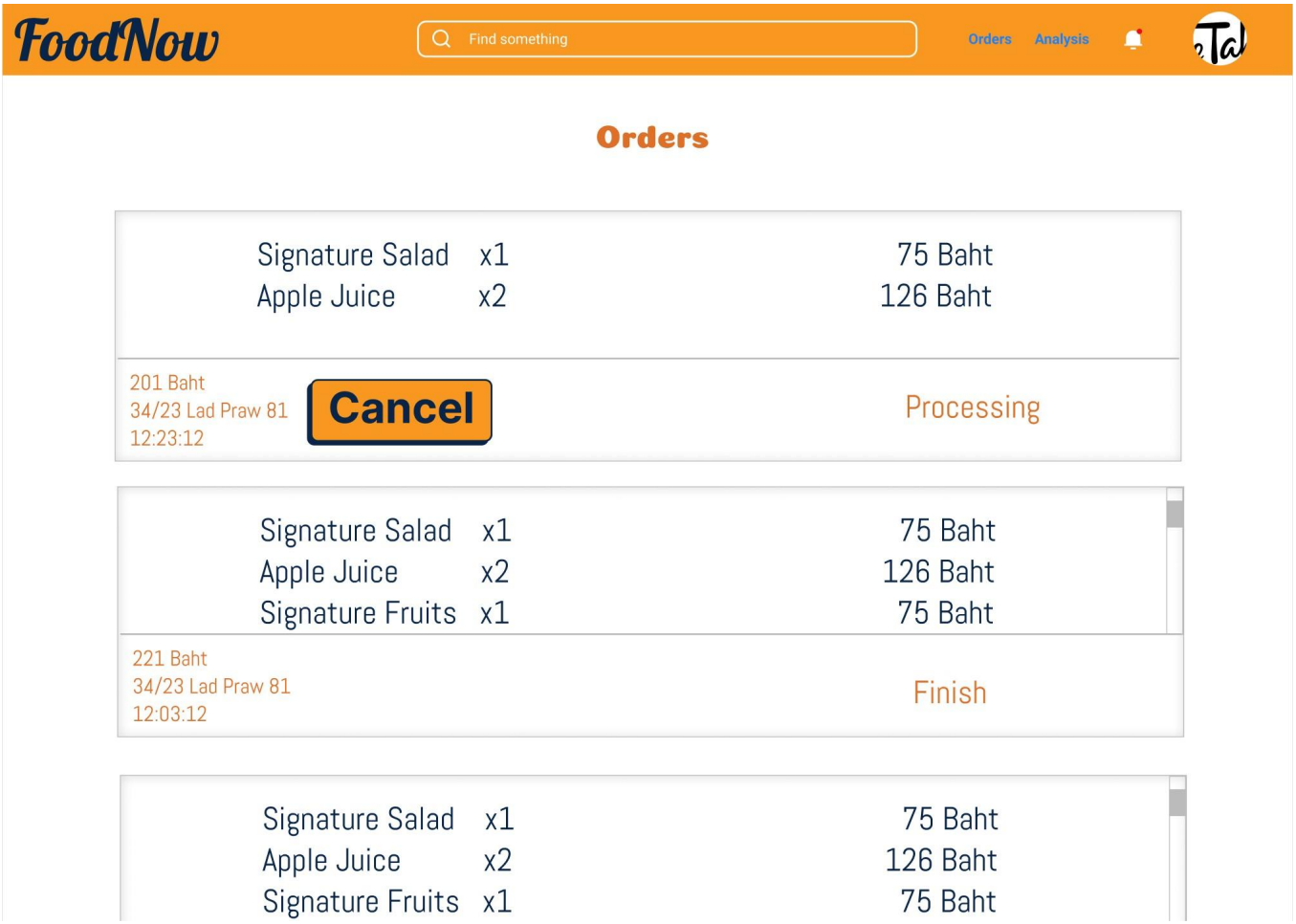
2-2: Add menu item pop up



2-3: Add new food item pop up



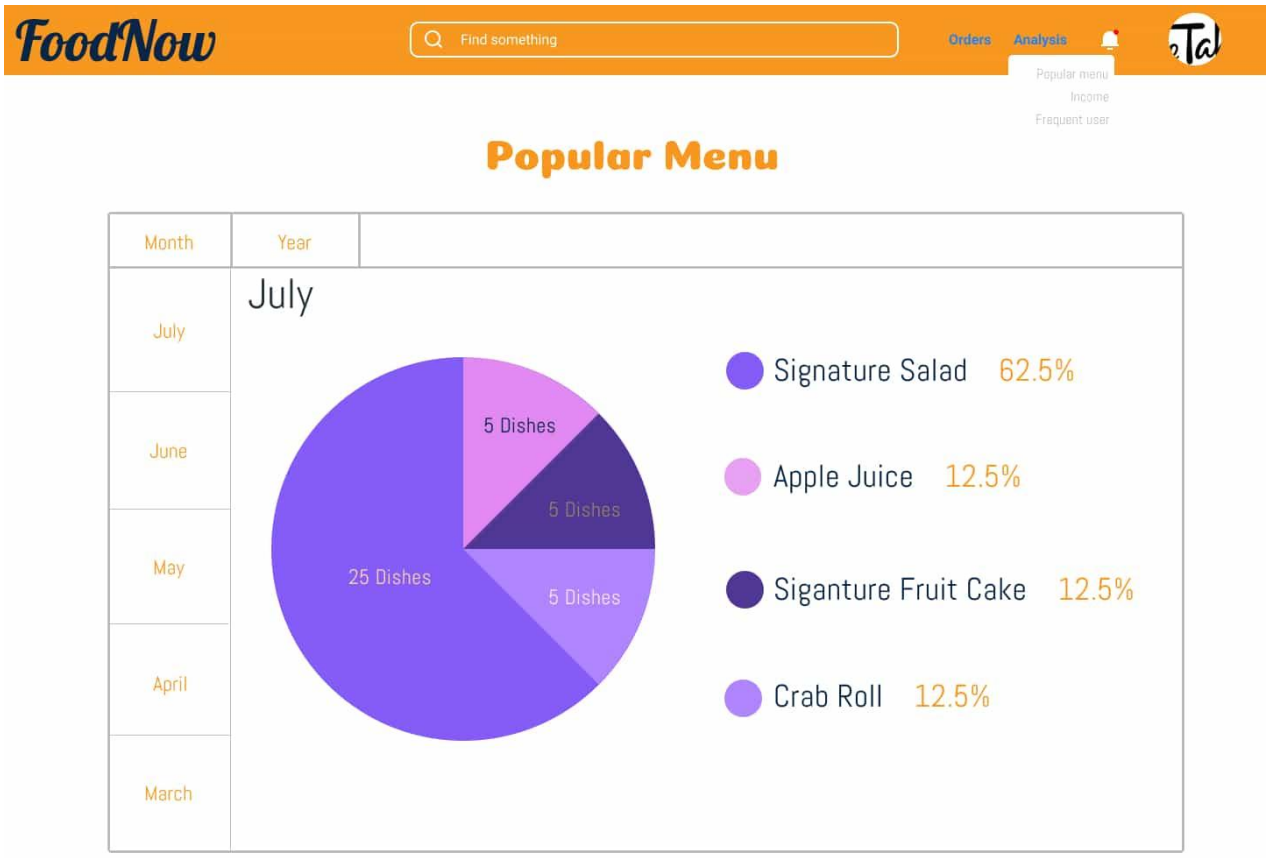
2-4: Edit food item pop up



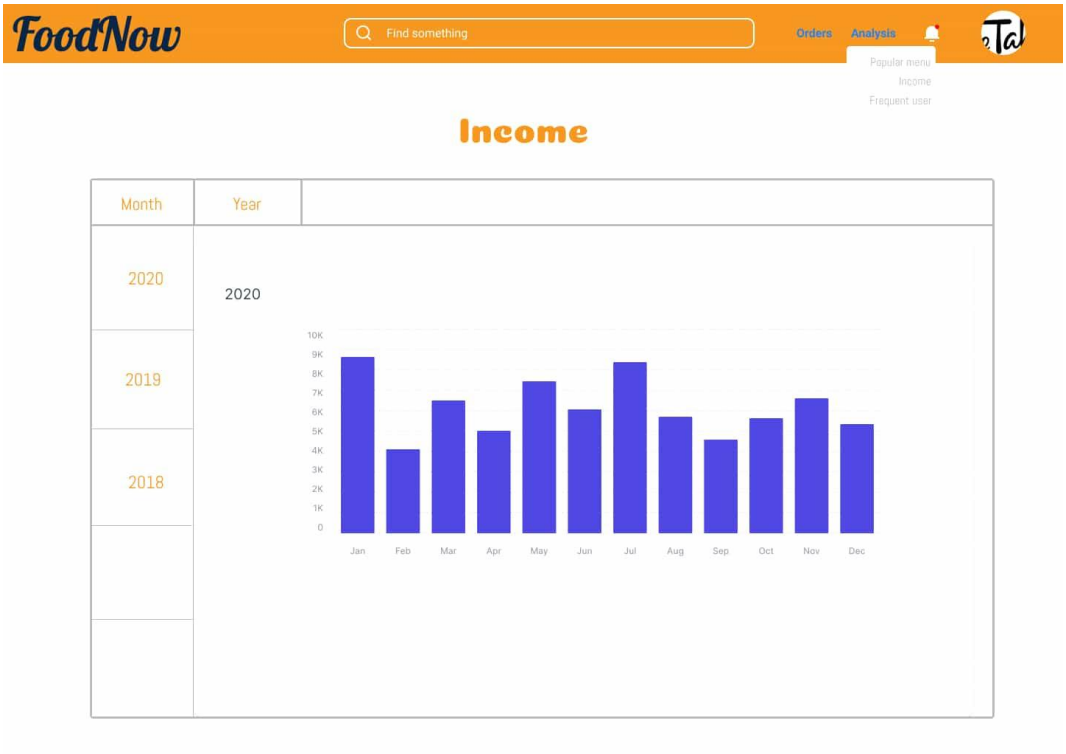
2-5: Order listing page

Restaurants can see all active orders and historical orders by clicking the 'Orders' link on the navigation bar.
Users can scroll on the mini view to see all the orders or click on the order to see the full listing pop up.
Status of order will change after the deliverer sends a confirmation on successful delivery via DeliverNow application (refer to the other app this team had developed for the delivery employee).
Restaurants can press cancel to reject orders.
Restaurants will also receive notification for new orders with the bell icon on the navigation bar.

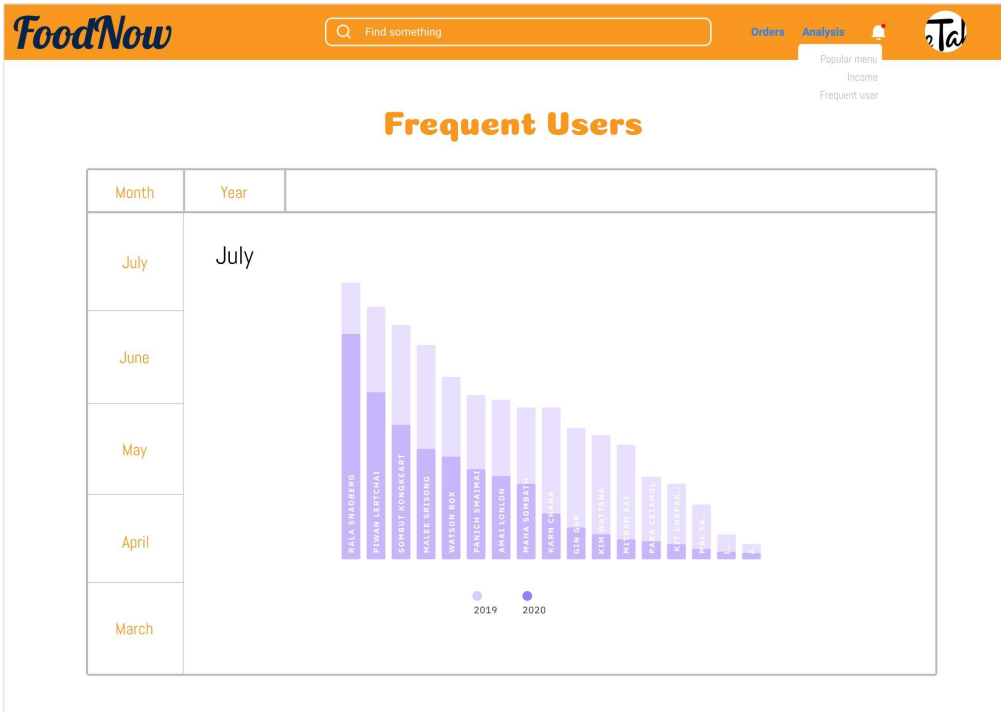
For the analysis function click the ‘Analysis’ link on the navigation bar. Drop down will appear with 3 choices: Popular menu, Income, and Frequent Users
Popular menu will show the most ordered menu from this restaurant based on month or year.
Income will display the raw total price the customer pays for each order calculated based on each month, year.
Frequent Users will display customers that have ordered from this restaurant 5 times or more based on each month, year.



2-6: popular menu analysis

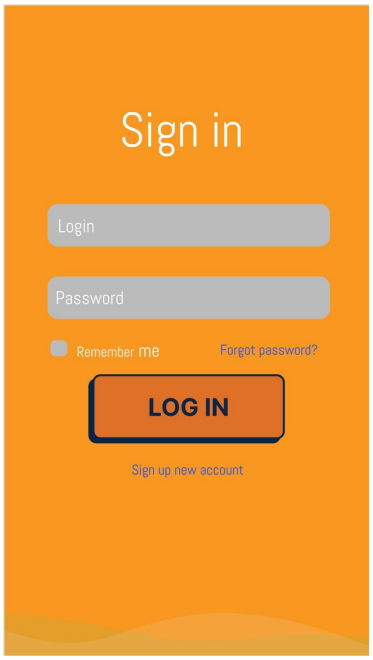


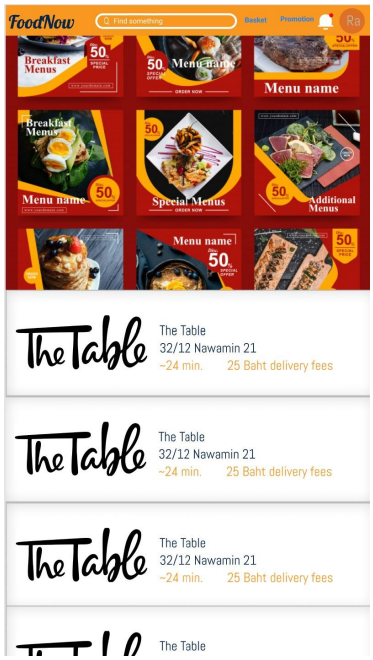
2-7: Income analysis

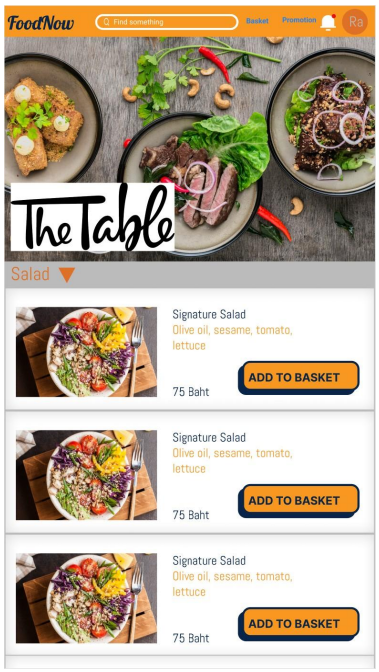


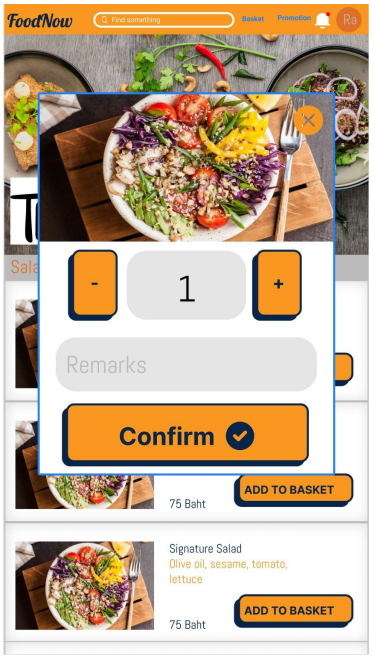
2-8: Frequent Users analysis

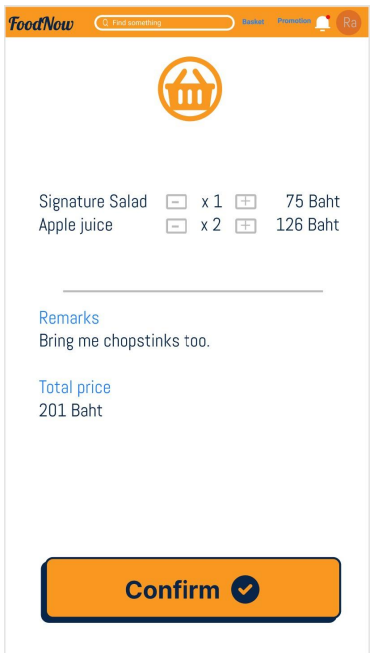
UI for mobile phone

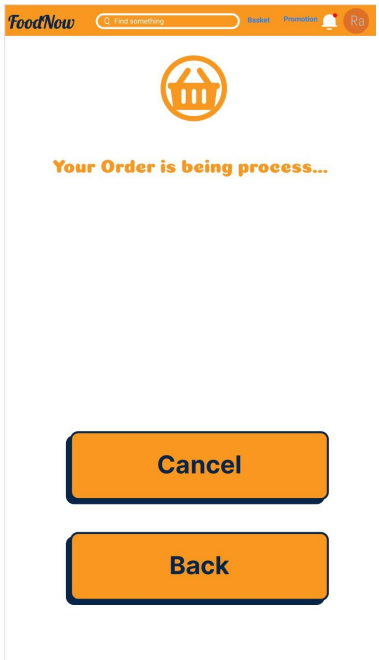






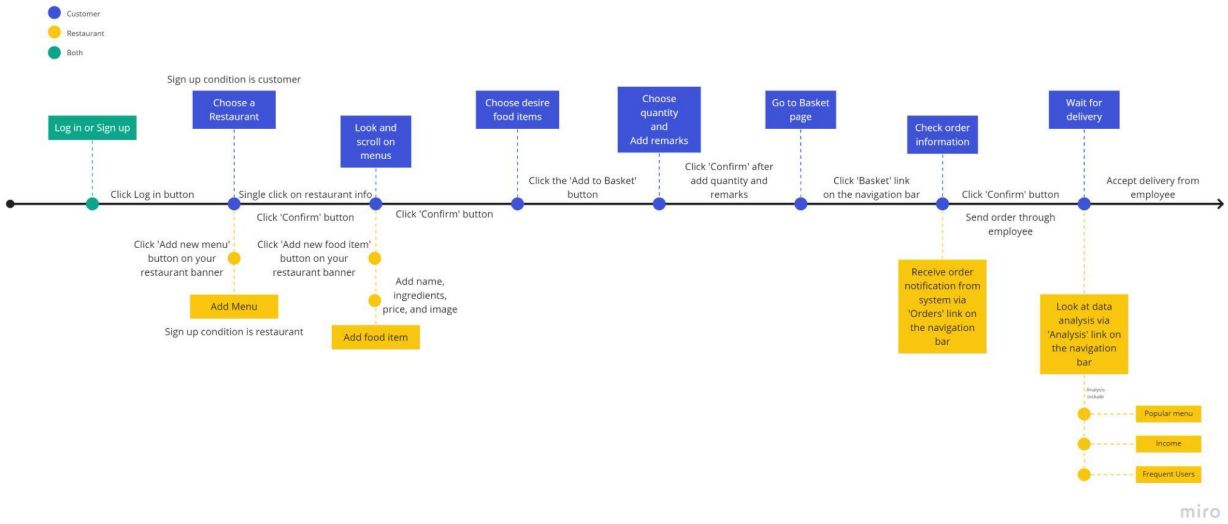






Link for all UI images

User journey



Link for better quality image

Customer journey shows customer's activity from logging in to finishing the order.
Restaurant journey shows restaurant's activity from creating new menu and product to accepting orders from customers. Additionally, check data analysis after successful delivery.

Hardware interfaces

Server side

Backend services will be hosted on AWS cloud. This will allow us to deploy systems quicker and easier to scale in the future. Moreover, the cost will be charged per usage which will be fit for a new community like "FoodNow".

It will consist of AWS Virtual Private Cloud (VPC) containing AWS EC2 which is running FoodNow services and Database. These will be accessible in a private subnet. AWS API Gateway will be used to handle and manage API traffic and API security.

EC2 will be installed on the latest Ubuntu with a proper python environment and MySQL.

Client side

UI is a Web frontend based on React so there is no software installation on the client side so any normal Desktop or Laptop or mobile devices with internet access will do.

Software interfaces

This project will interface with a Database Management System (DBMS) that stores the information related to the service.

Since this project is a web application, in order to be runned Windows must be at version 7 or higher, IOS must be at version 10.12 or higher, and Android must be at SDK 9.0 or higher.

Class Diagrams

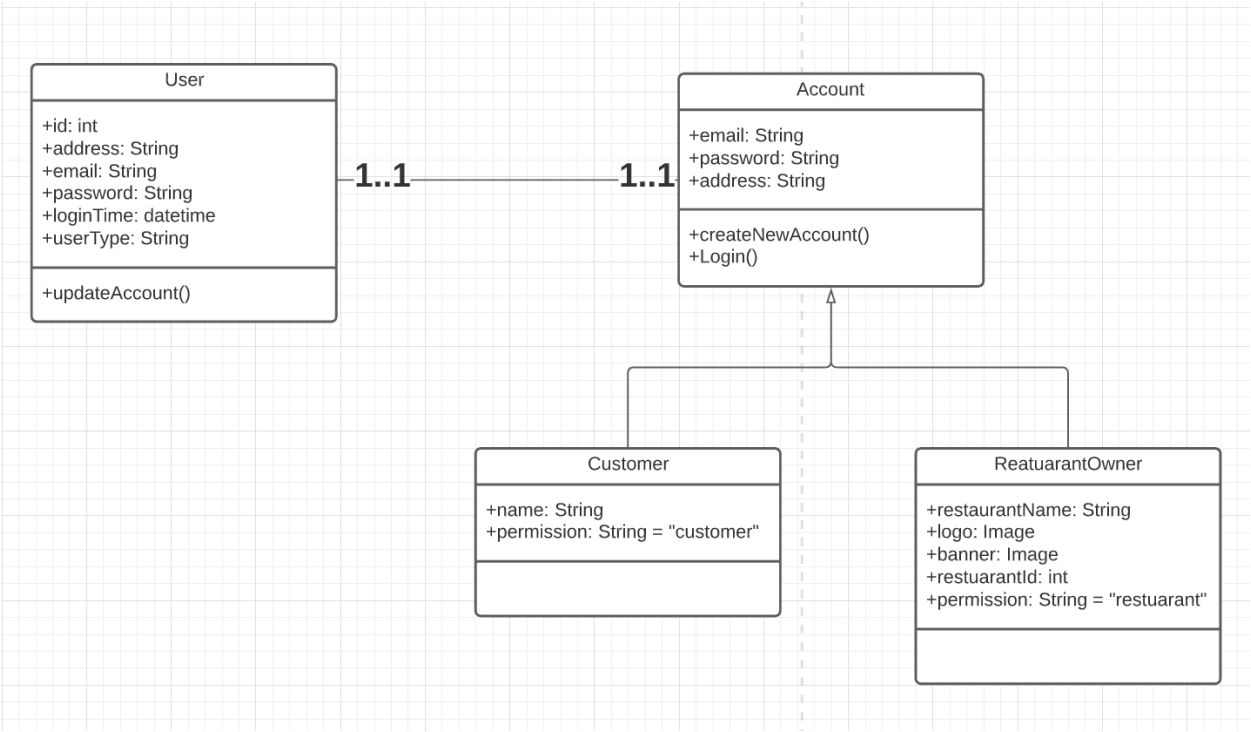


Diagram for user's account with thay sign up or login.

User sign up, log in, and information input is handled by 'Account' with 2 permission types: customer, restaurant. Then after successful login the user data will transfer to keep in 'User' which connects to the database.

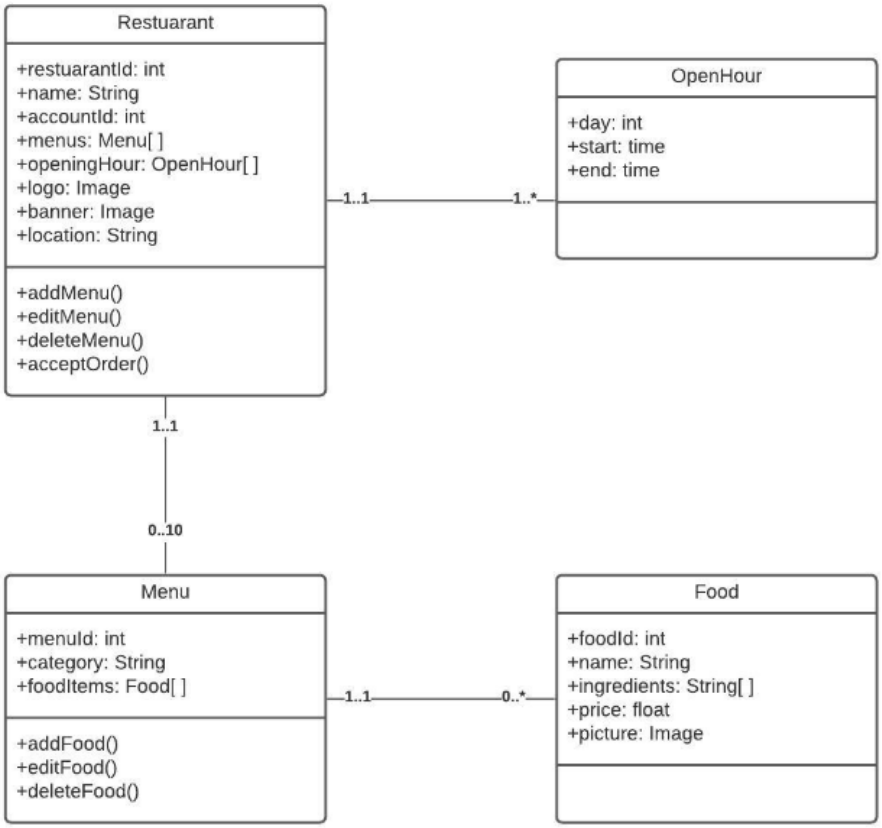


Diagram for restaurant, menu, and food item.

The ‘Restaurant’ class will mainly be associated with 3 classes. ‘OpenHour’ contains the days and duration of opening time. ‘Menu’ contains category and Food items in that category, Restaurant can have multiple menus. ‘Food’ contains information about each food item.

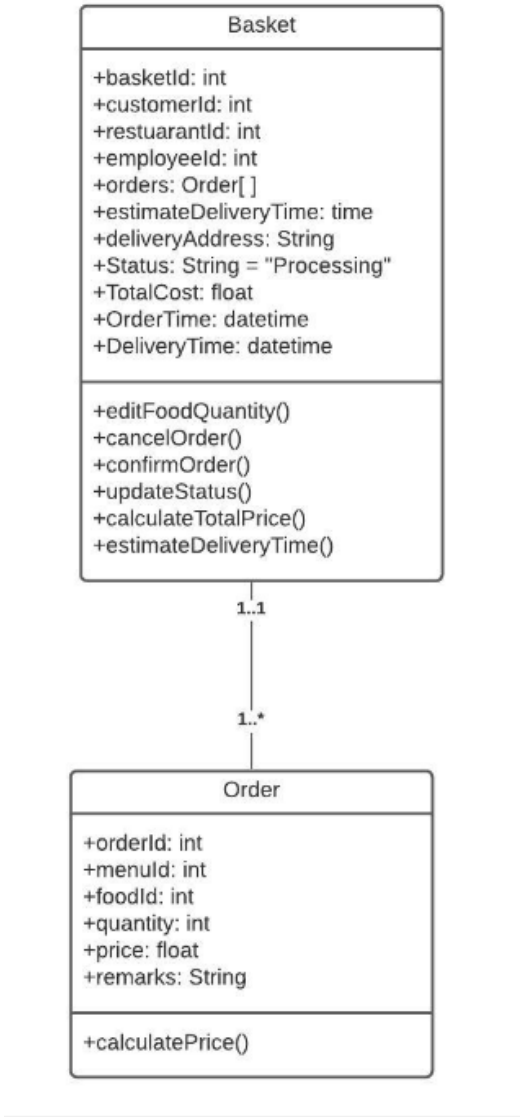


Diagram for Basket Order

'Basket' will contain the IDs of customer, restaurant, employee so that it can connect to all parties involved. One 'Basket' can contain multiple 'Order'. Each 'Order' contains information about each food item and it's information it is to be ordered.

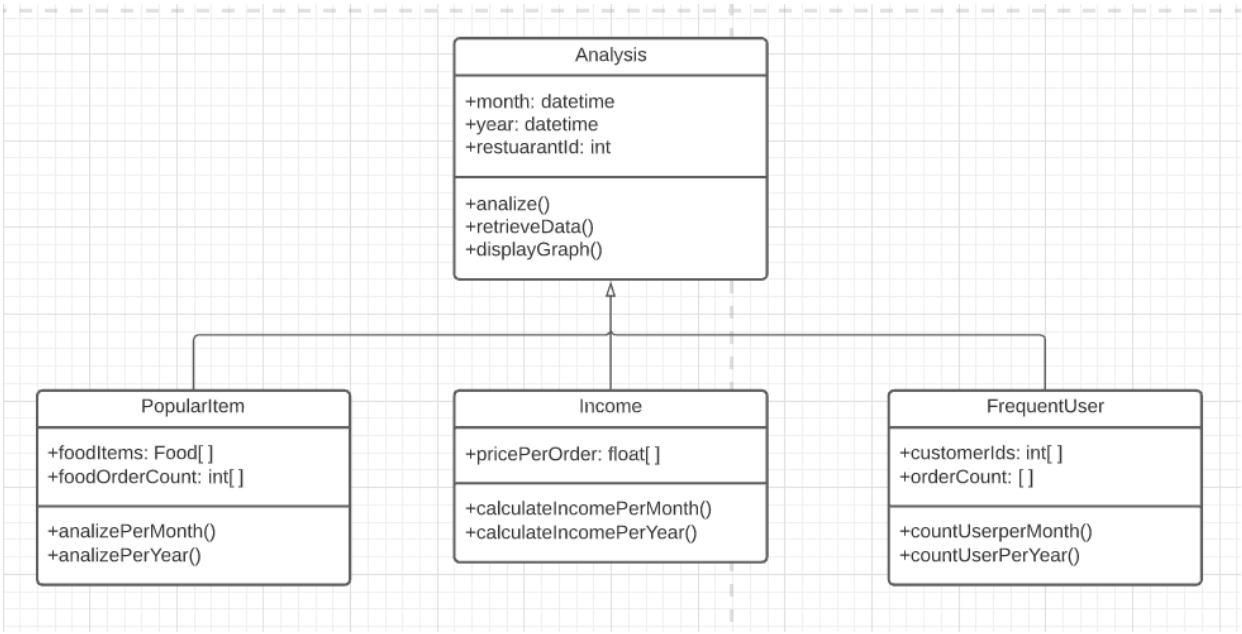
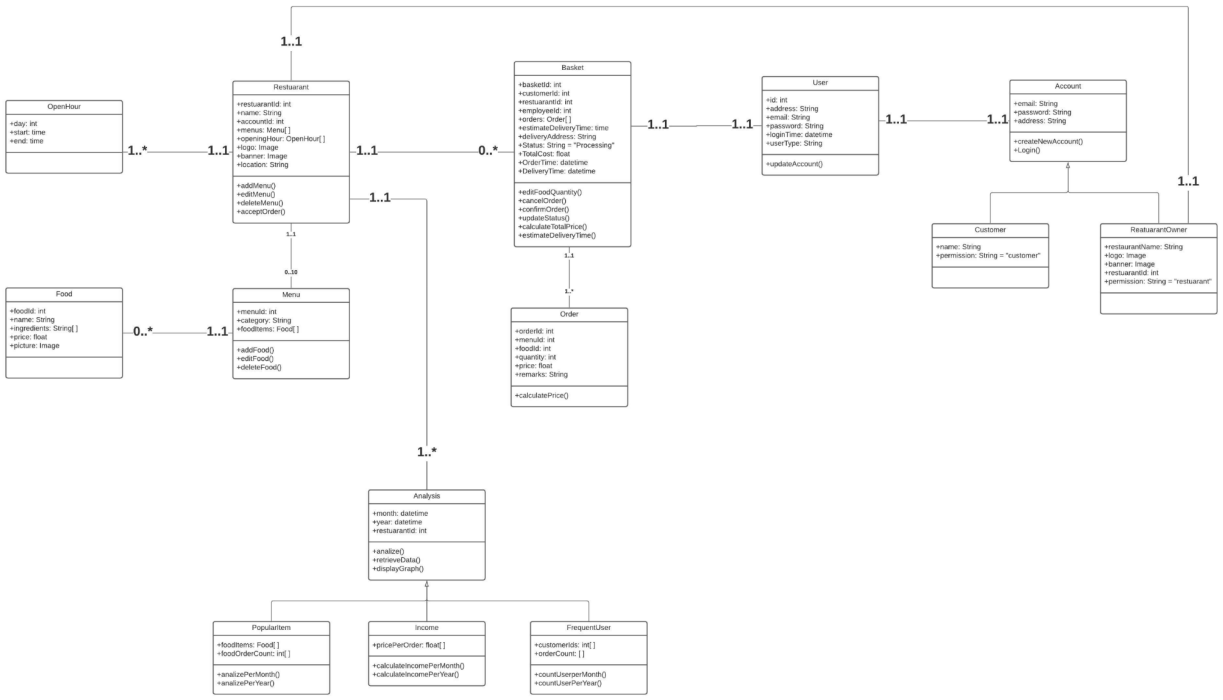


Diagram for analysis function

‘Analysis’ retrieves and processes information. It contains 3 subclasses for each analysis function.

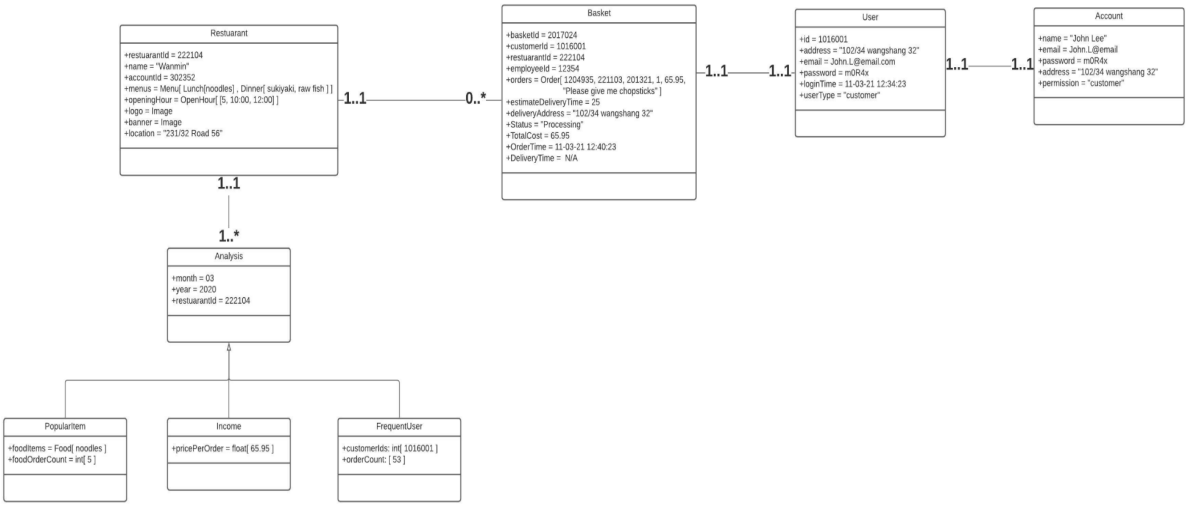
Link for each component class diagram better quality images

Overall software class diagram



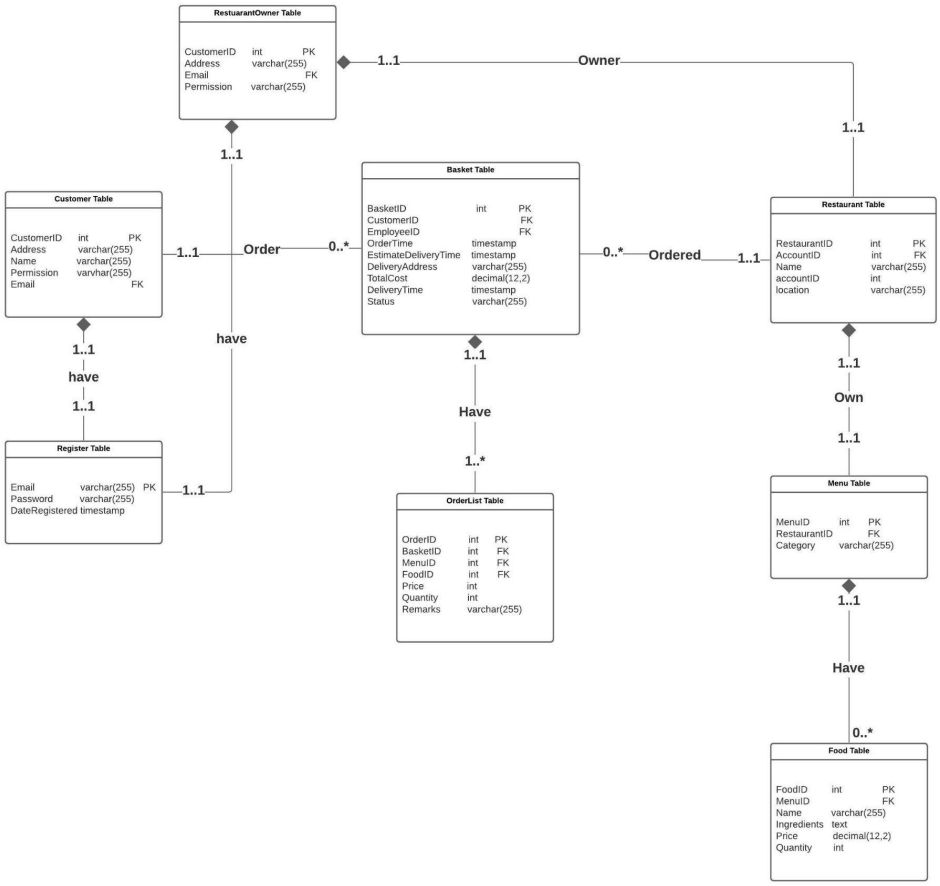
Link for better quality image

Overall software object diagram



Link for better quality image

Database UML Diagram



Link for better quality image

Communications interfaces

HTTP protocol and methods to communicate through the internet because FoodNow platform is hosted on AWS. All devices connected should have Internet access.

Application programing interfaces

The FoodNow API is the API between FoodNow UI and FoodNow backend services.
The API is designed around REST. It has predictable resource-oriented URLs, returns JSON responses, and uses standard HTTP response codes.

Errors
The FoodNow API uses standard HTTP response codes to indicate the success or failure of an API request.

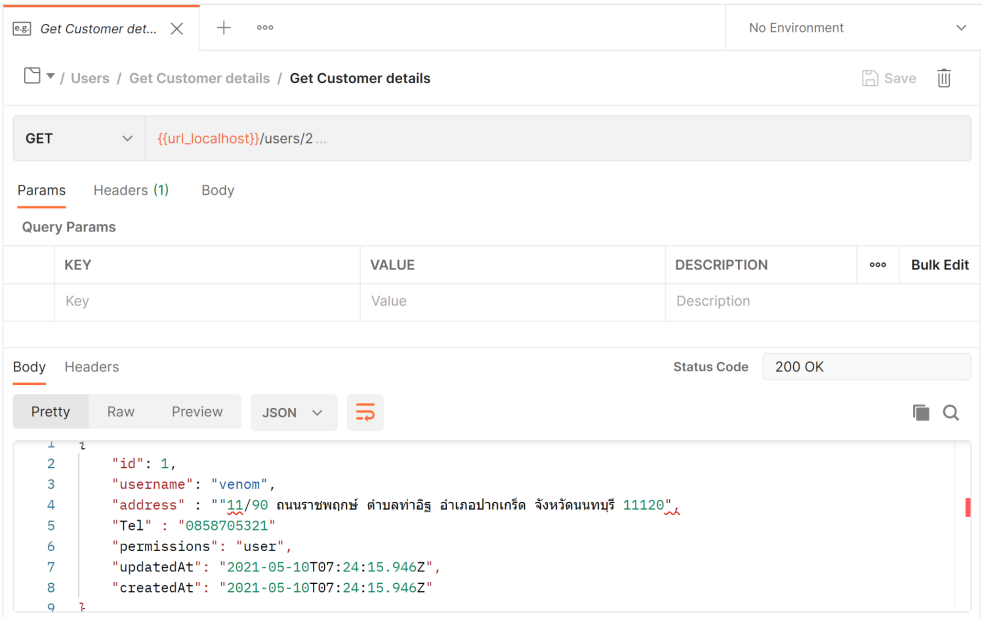
HTTP status code summary
200 - OK. Everything worked as expected.
201 - Created. Create success.
400 - Bad Request. The request was unacceptable, often due to missing a required parameter.

User API

The FoodNow implement 2 roles with different permission levels:
Regular Customer : Can see all restaurants and place orders from them
Restaurant Admin: Can create/read/update/delete (CRUD) its restaurant details ,menu and dish.

GET : customer details
Using Get method to retrieve customer information.

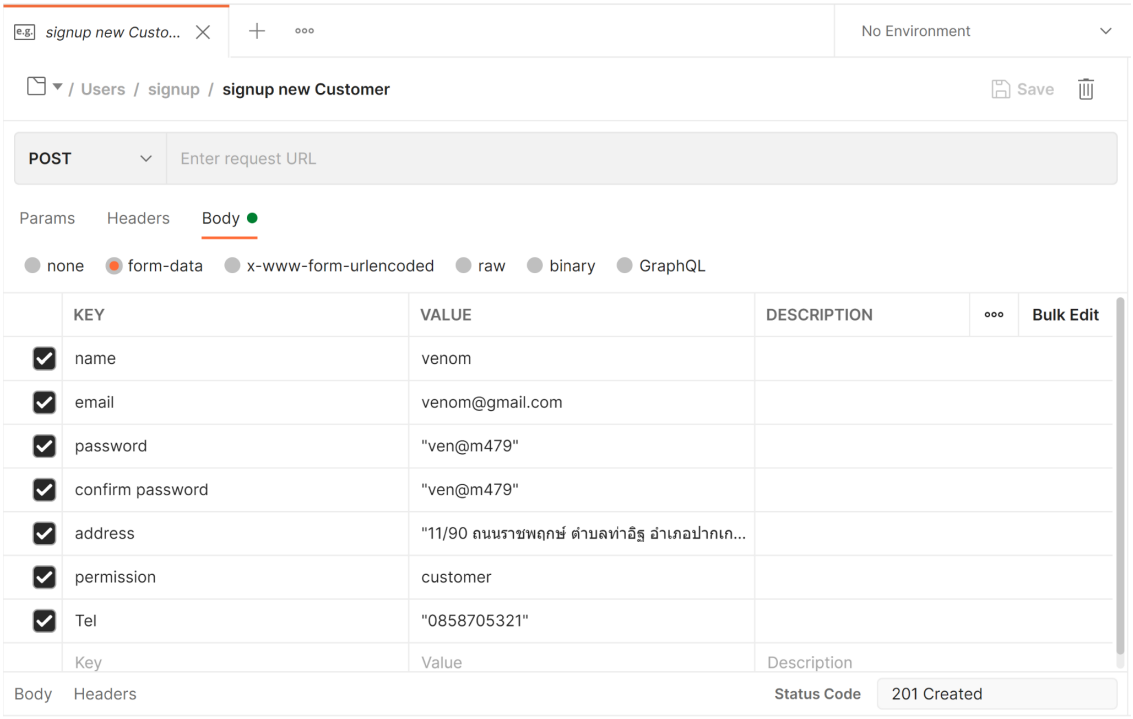
https://api.example.com/users/:id
Request Headers:
Content-Type: application/json
Path Variables:
id: User Id

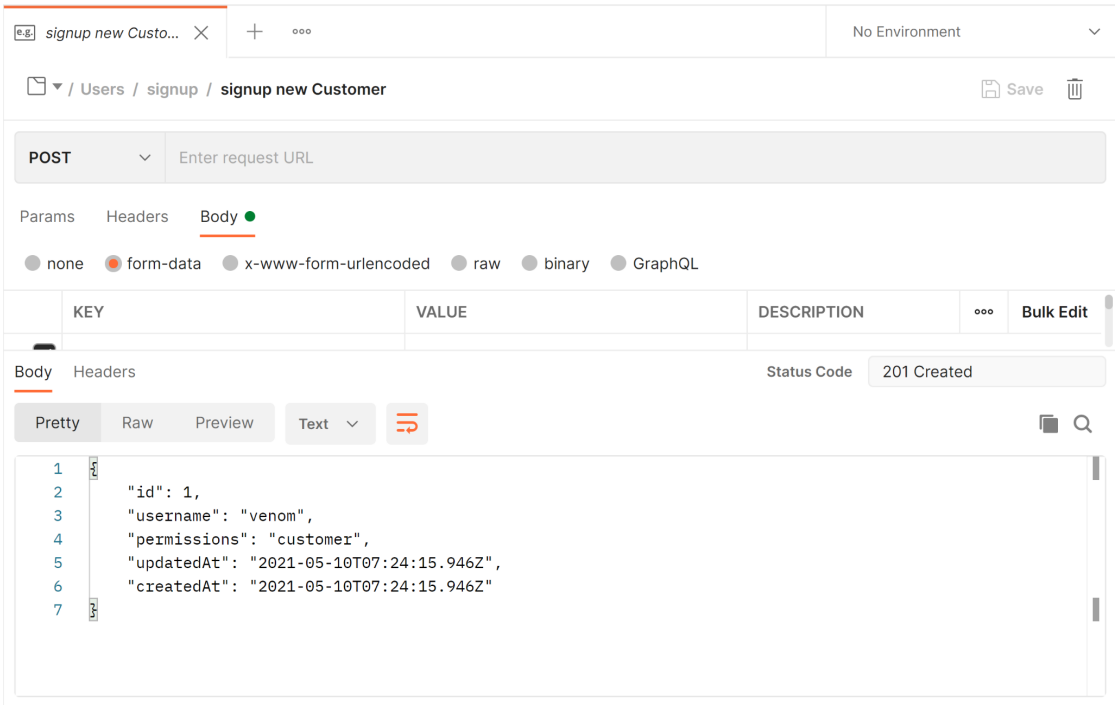


POST: sign up

Using POST method , this API will allow you to sign up based on type of users i.e. if a regular customer or restaurant admin.

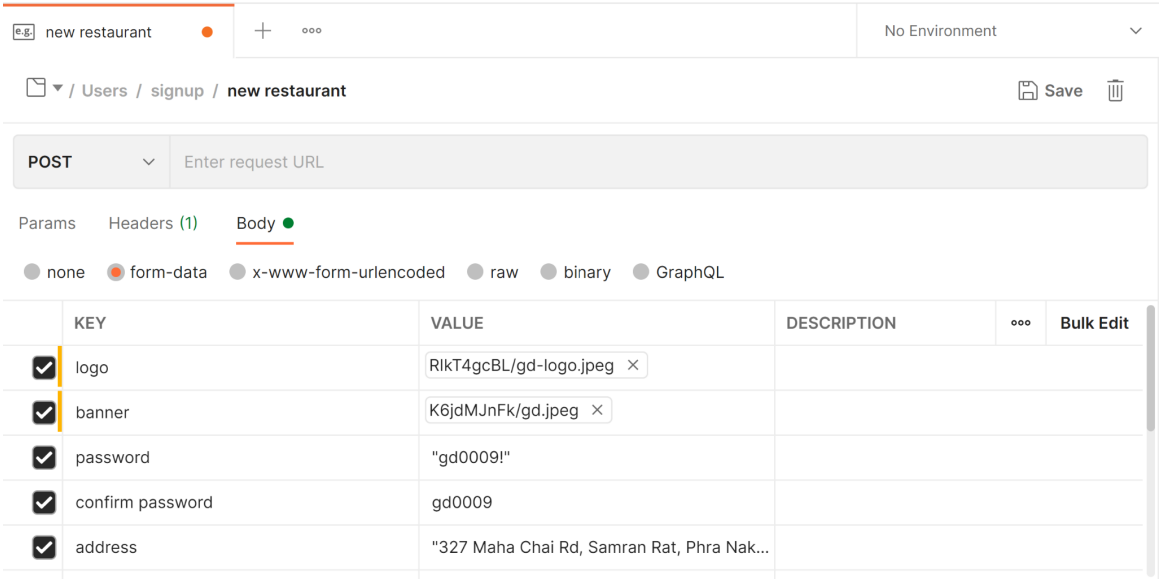
POST: Sign up as customer



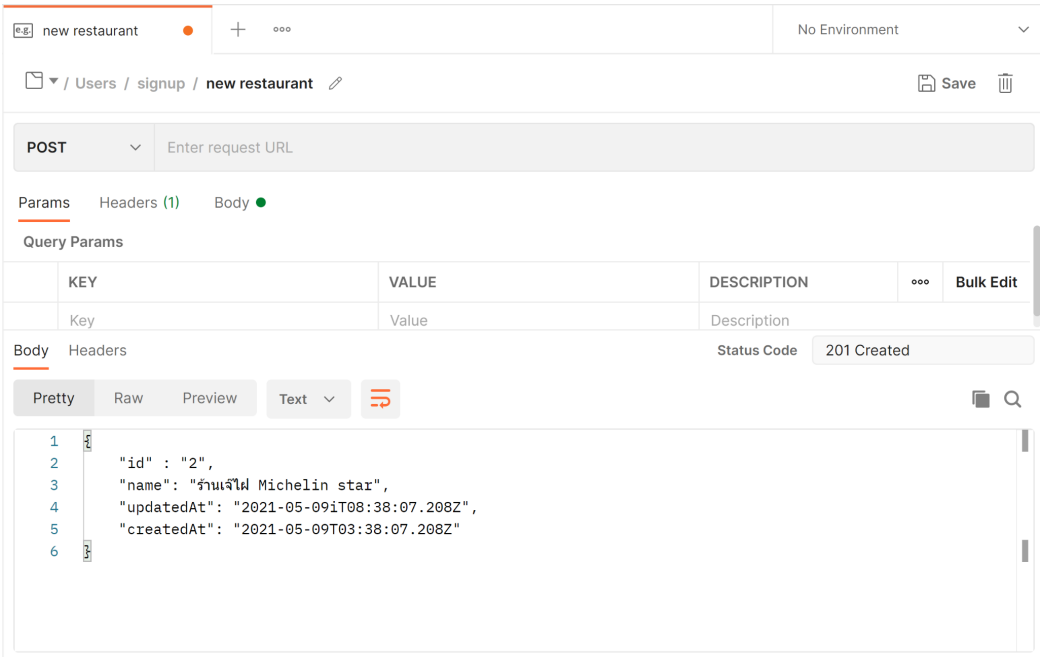


Return value

POST: Sign up as restaurant



☑	restaurant name	"ร้านเจ๊ไฝ Michelin star"
☑	typeoffood	"thai"
☑	open time	"10 AM - 23 PM"
☑	status	open
☑	description	อาหารไทย อาหารทะเลสดรีทฟู้ด ระดับมิชิลิน
☑	Tel	"022239384"



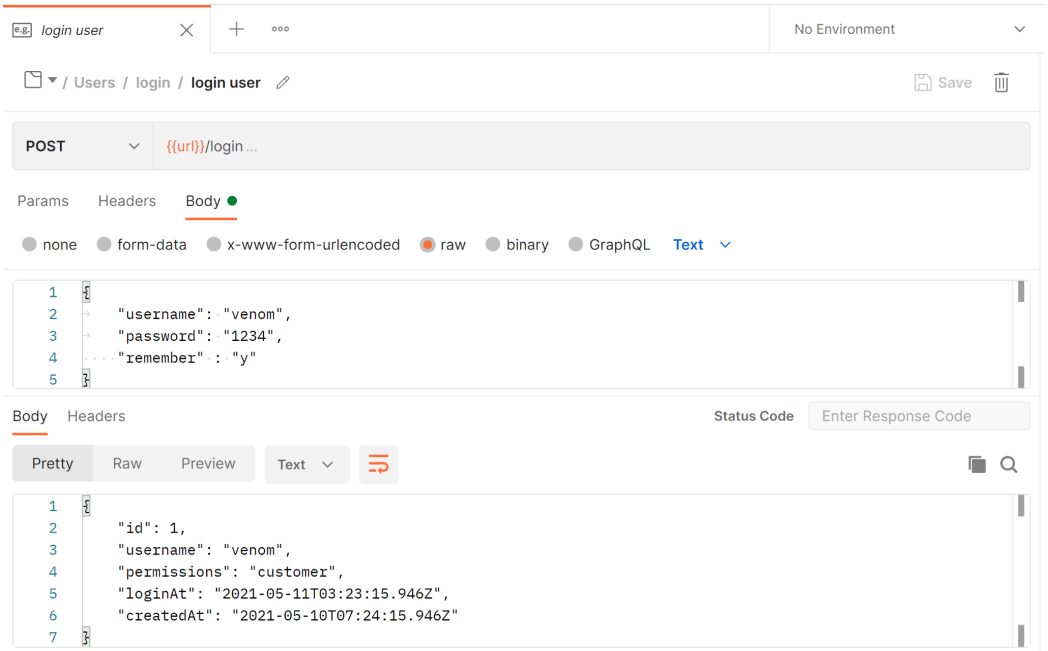
Return value

POST: log in

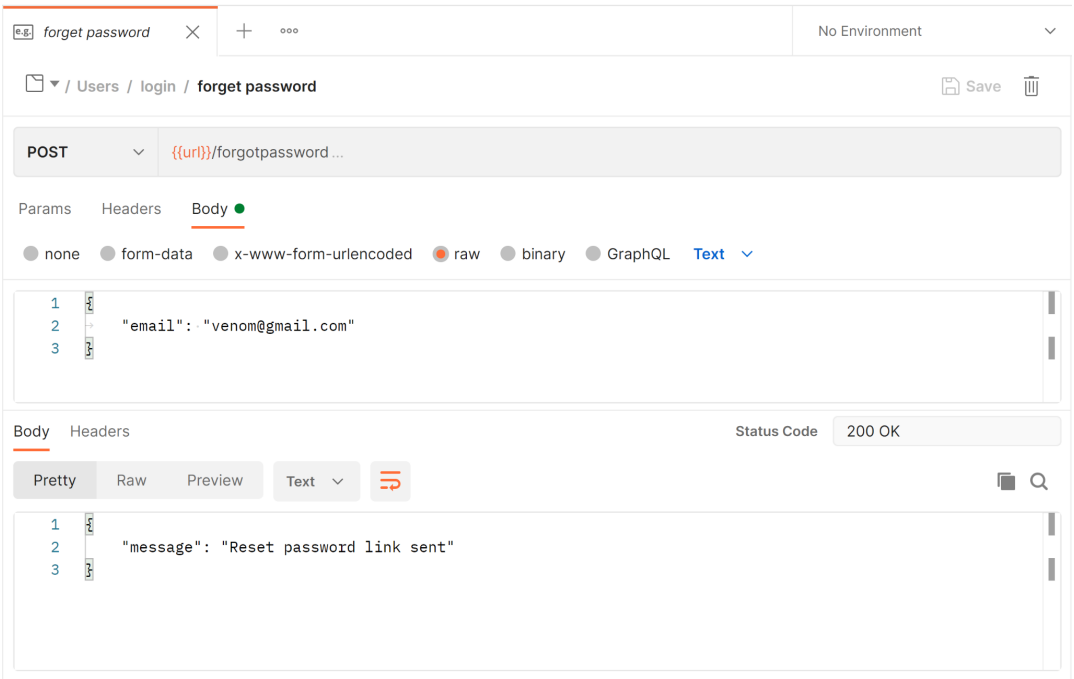
These API will allow to

- login
- forget password which a reset password link will be sent to registered email.

Log in



Forget password



Restaurants API

This is a set of API that allows CRUD with restaurants. Only restaurant owner permission can do this activity.

GET: restaurants list

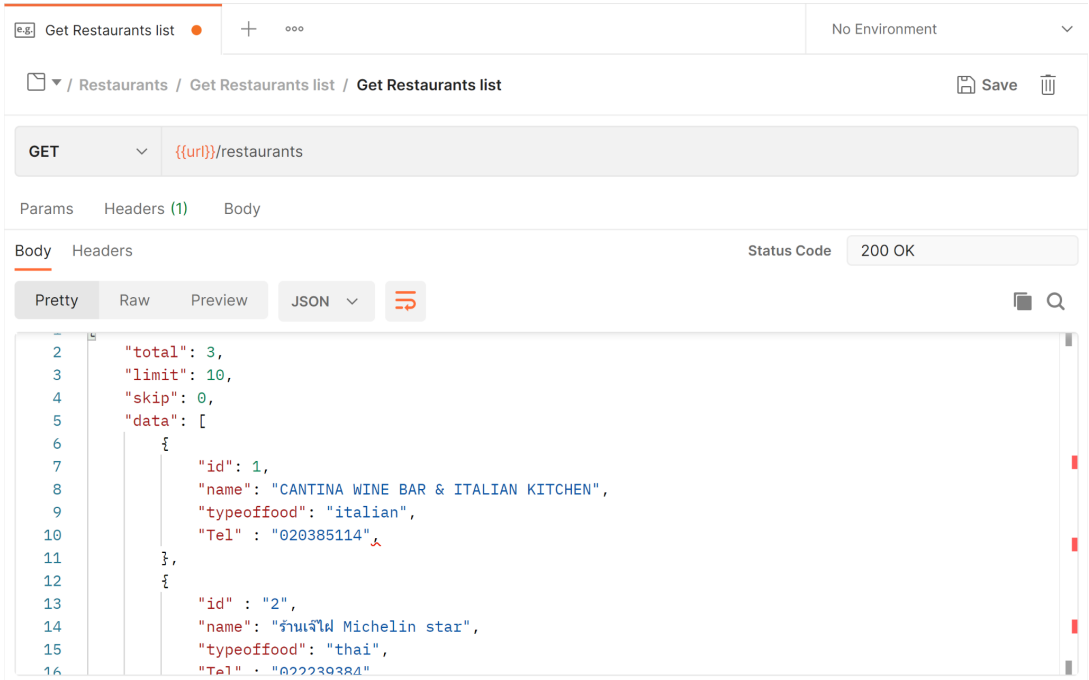
This is to get restaurant.list. It will return 10 restaurants at a time.

https://api.example.com/restaurants/?last=id

Request Params:

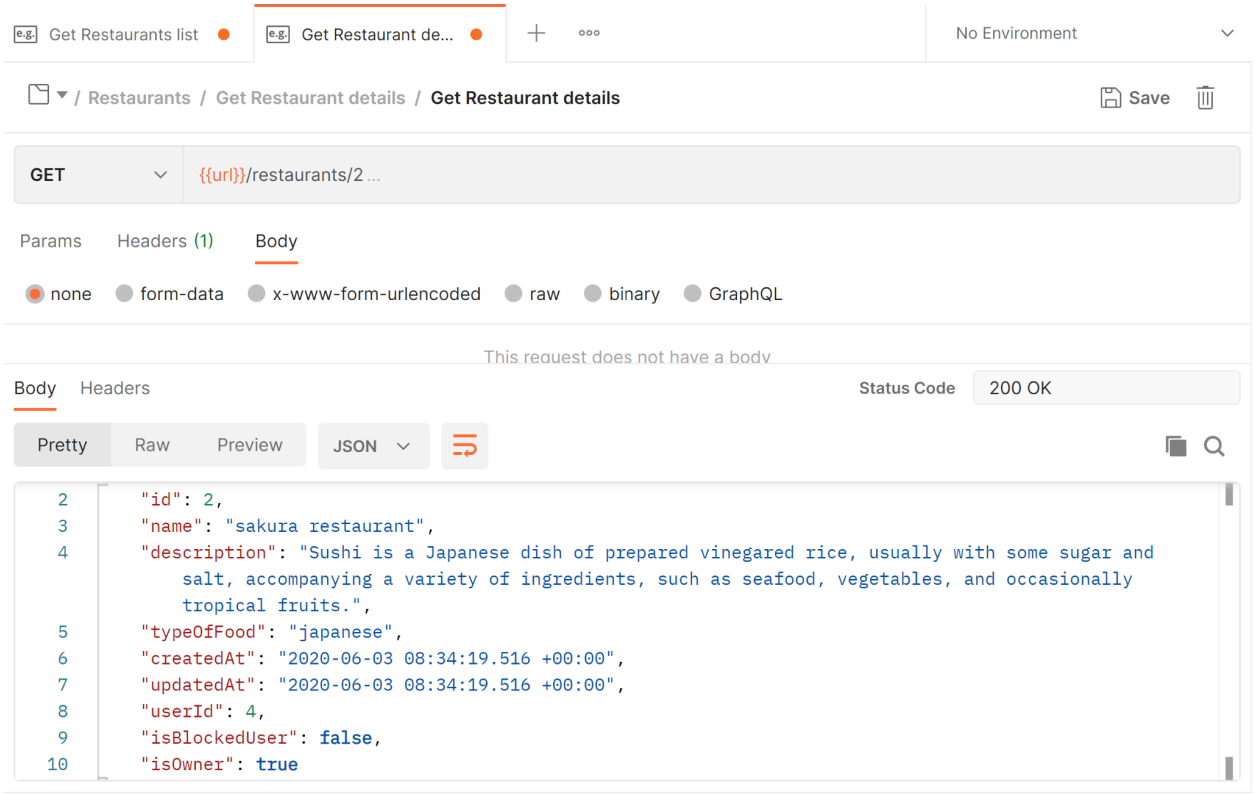
last: id

last id of restaurant return by previous call. the first call will be 0.



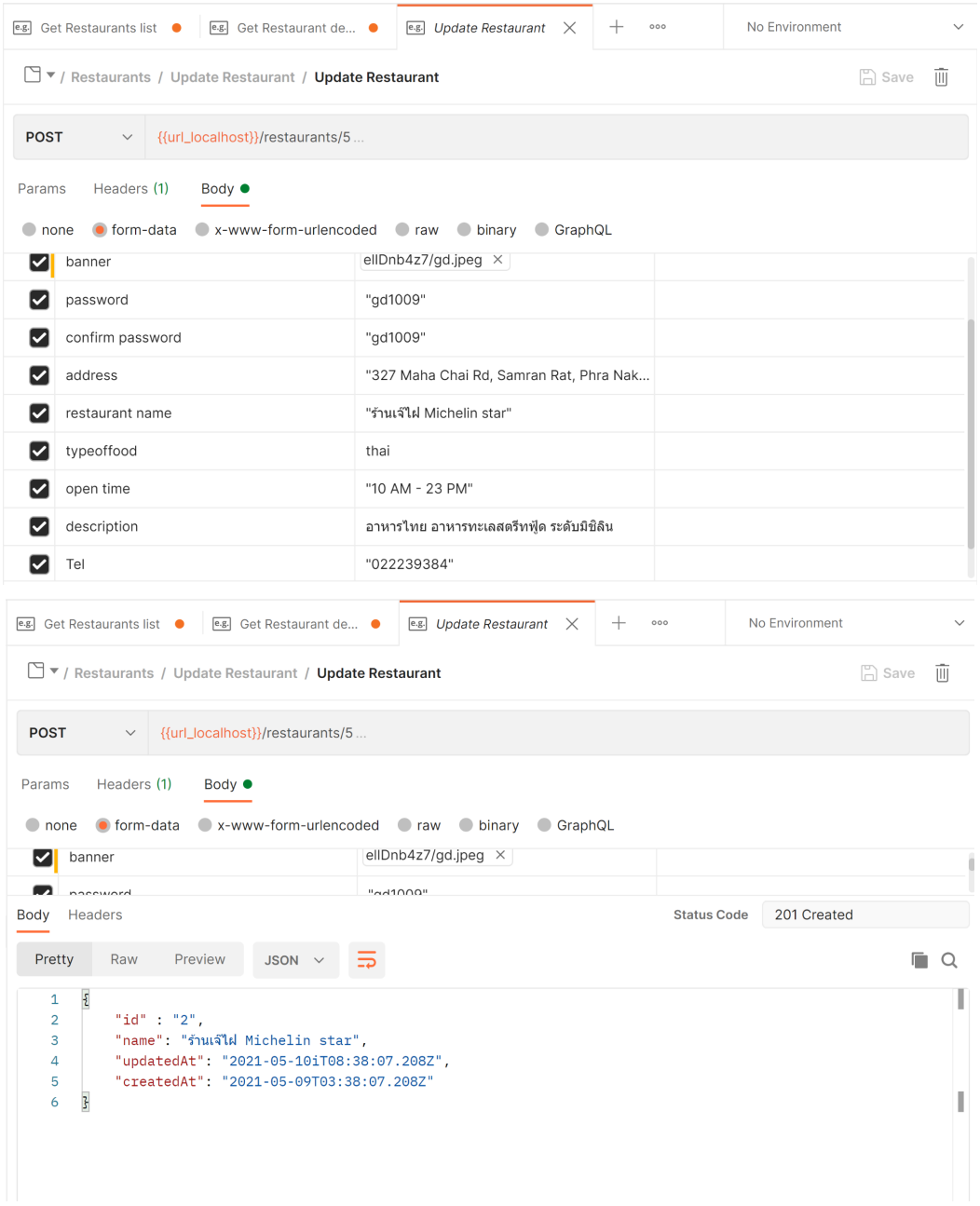
GET: restaurant details
This API will return specific details of this restaurant.

https://api.example.com/restaurants/:id
Path Variables:
id: Restaurant Id



PATCH: update restaurant
This API is to update information about this restaurant.

https://api.example.com/restaurants/:id
Request Headers:
Content-Type: application/json
Path Variables:
id: Restaurant Id



Return value

Menu API

API to create/read/update/delete about the menu of the restaurant.

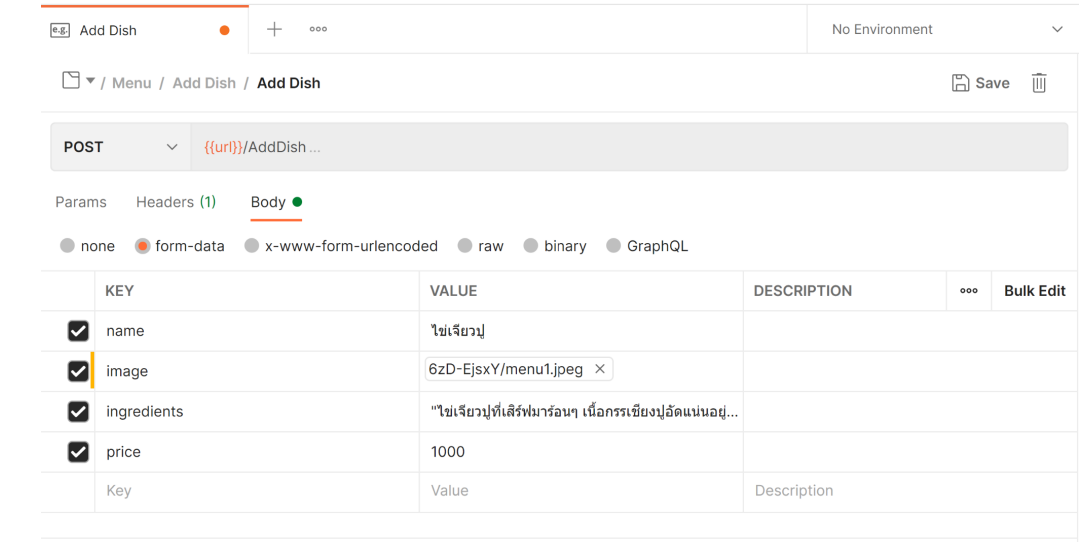
POST: add dish

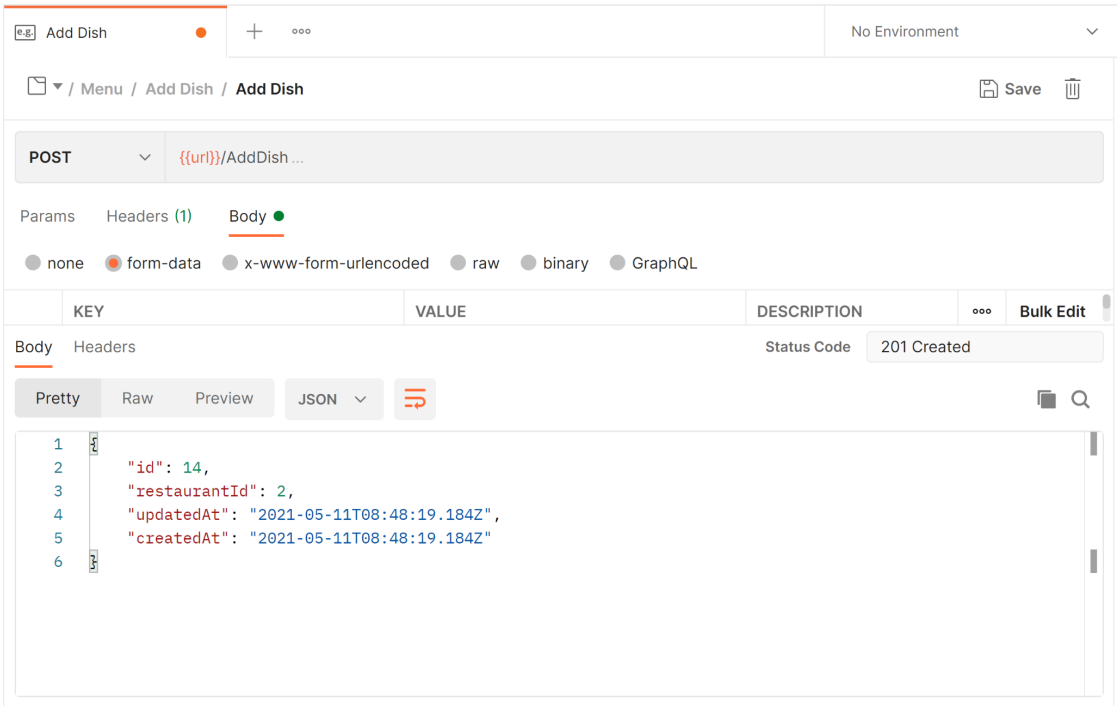
This request creates a new dish. info to add are name, image, ingredients, and a price.

https://api.example.com/adddishs

Request Headers:

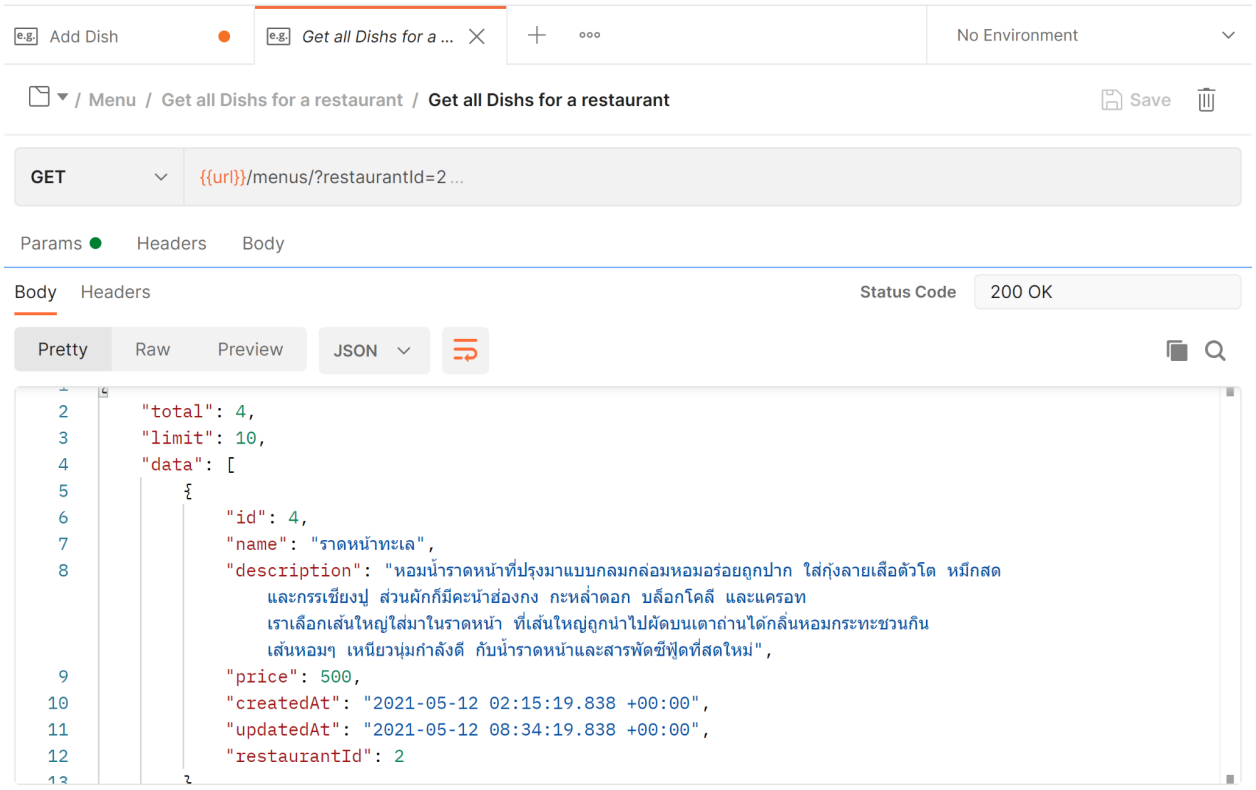
Content-Type: application/json





GET: get all dishes for a restaurant
Obtain a list with all dishes of this Restaurant.

https://api.example.com/Menu/?restaurantId=:id
Request Params:
restaurantId: id (Restaurant Id)



PATCH : update dish

This API is to update dish information.

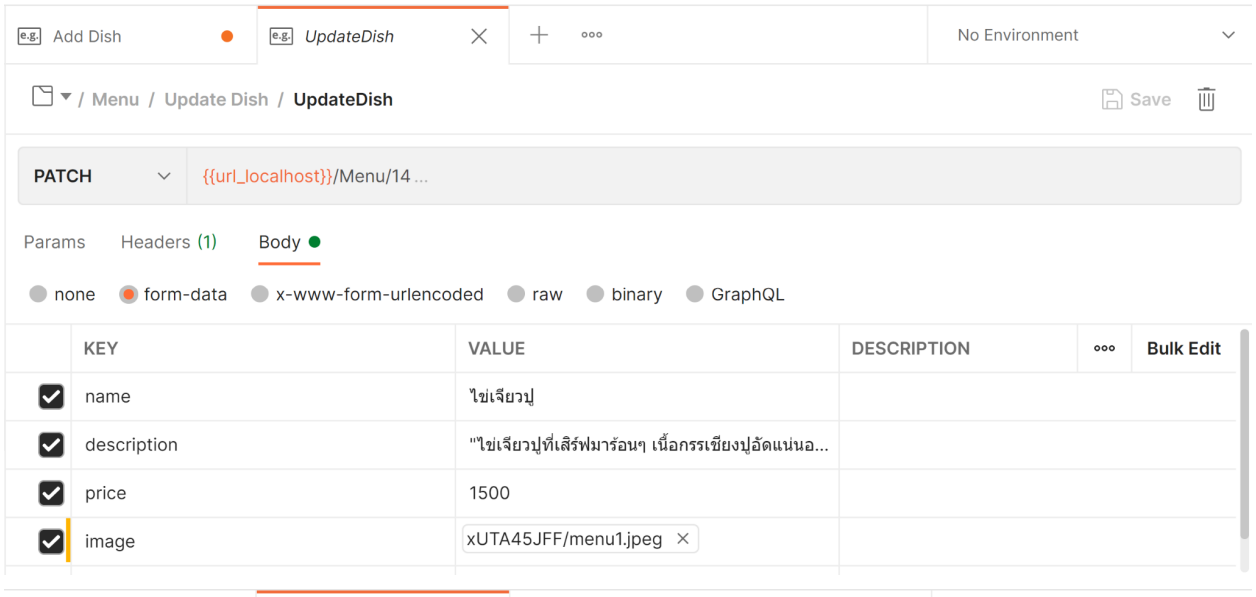
https://api.example.com/meals/:id

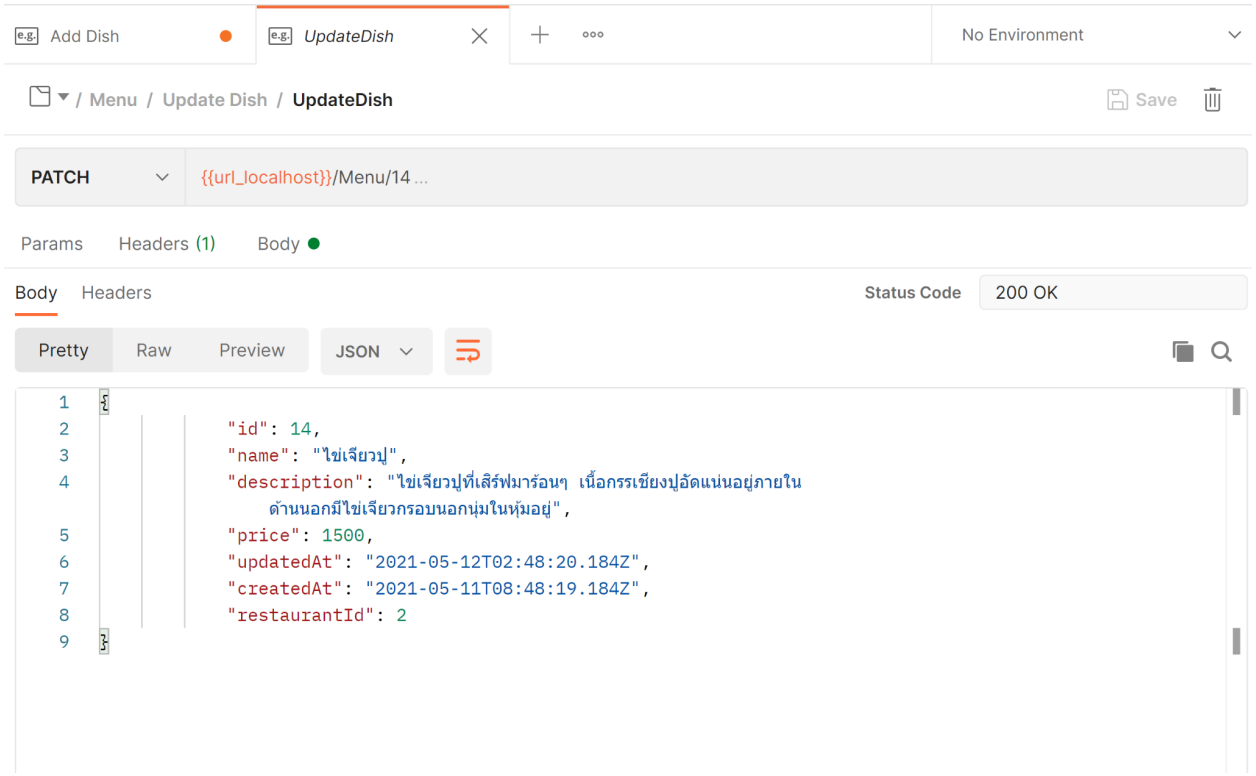
Request Headers:

Content-Type: application/json

Path Variables:

id

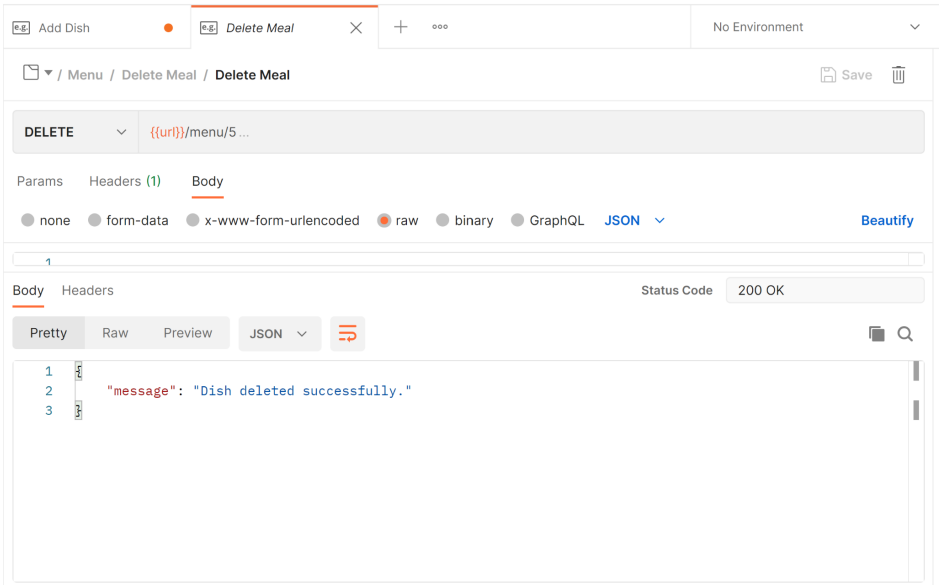




Return value

DELETE : delete meal
This request deletes a dish.

https://api.example.com/menu/:id
Request Headers:
Content-Type: application/json
Path Variables:
id

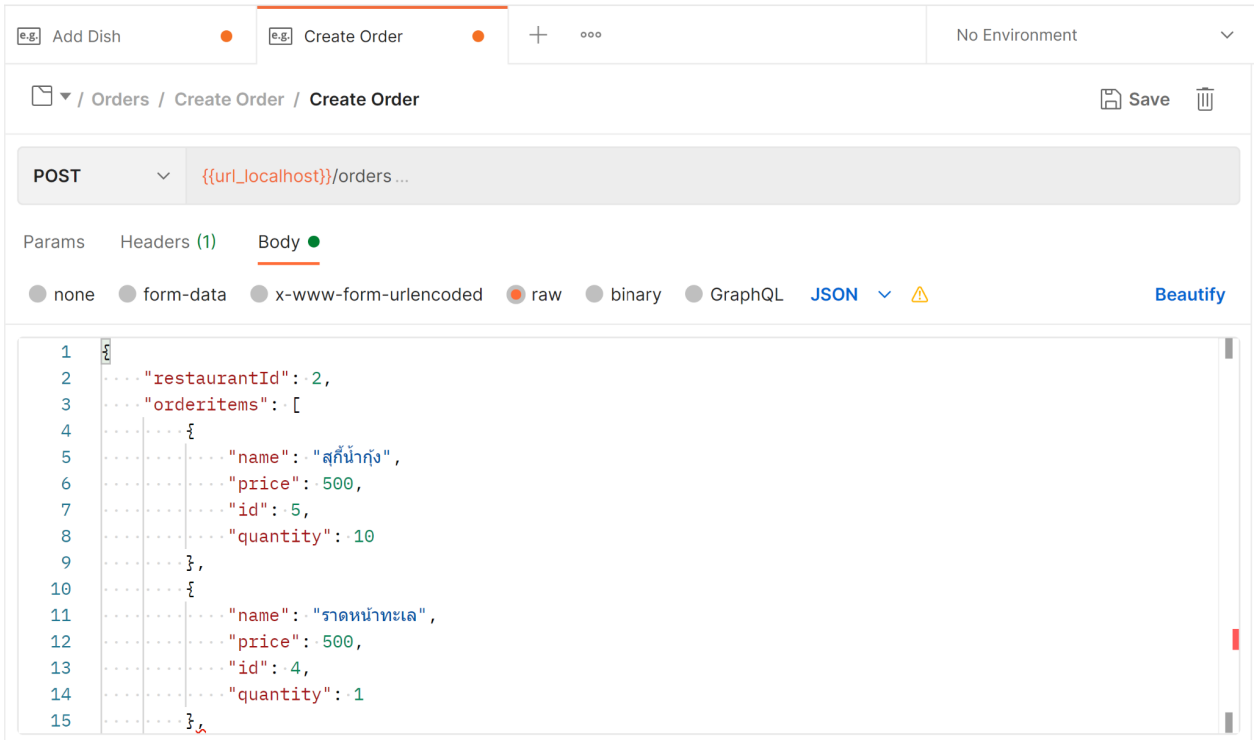


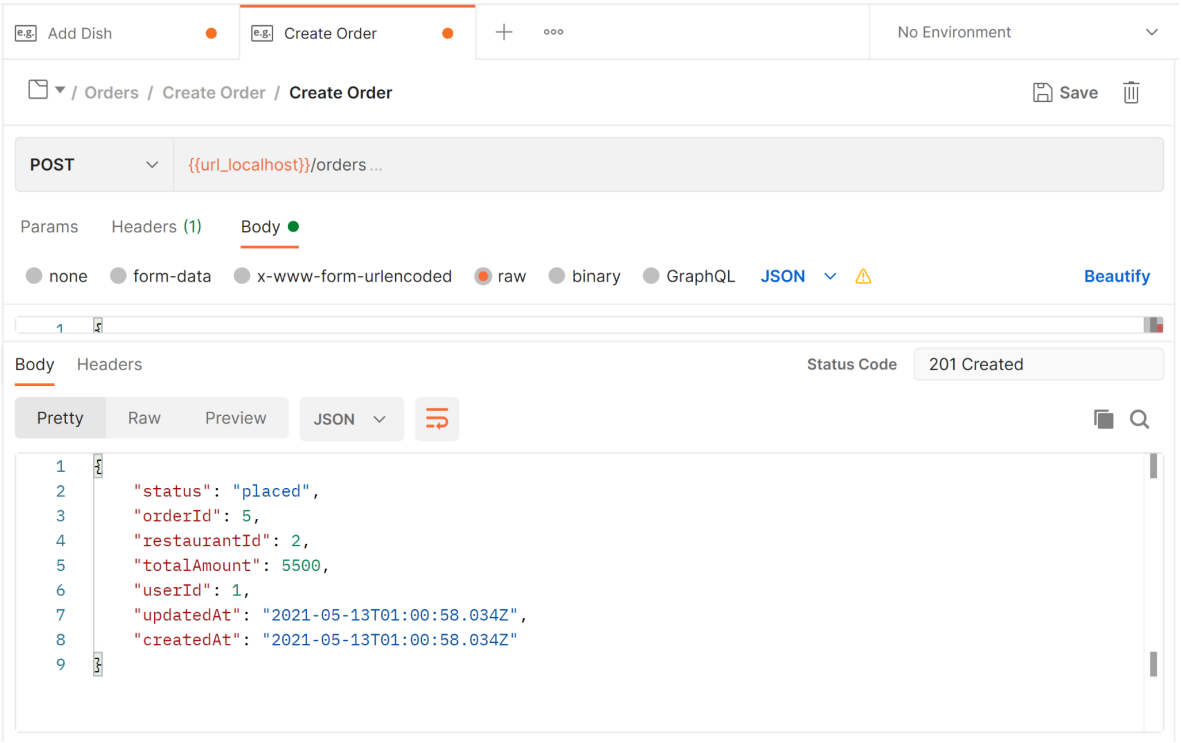
Orders API

This set of API will allow you to submit and view order information.
An Order should be placed for a single Restaurant only, but it can have multiple dishes.

POST: create order
This API is to place a new Order.

https://api.example.com/orders
Request Headers:
Content-Type: application/json





Return value

GET: get order details

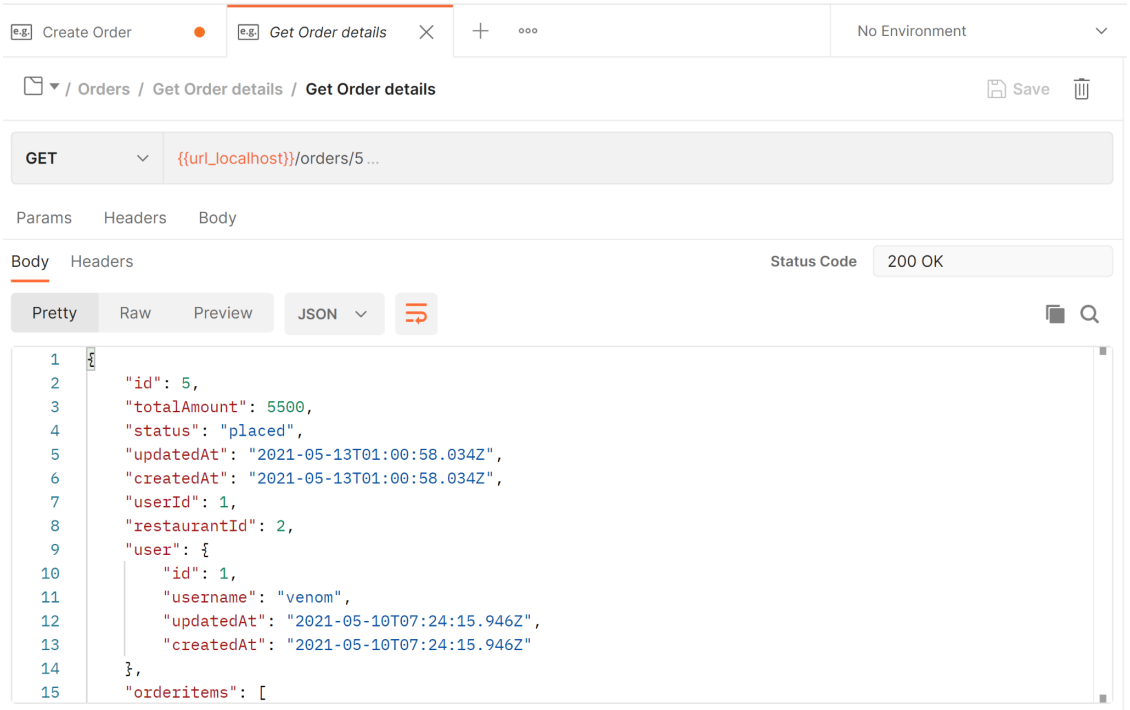
It returns the details of an Order, and it also includes:

- The list of meals, with its quantity.
- Related User information.
- Related Restaurant information.

https://api.example.com/orders/:id

Path Variables:

id: Order Id



PATCH : update order status

Restaurant employees can change the order status. Status respecting below flow and permissions:

- Processing: Once a client places an Order
- Canceled: A customer cancel the Order
- Rejected: A Restaurant employee starts to make the meals.
- Delivering: The Restaurant is delivering the order.
- Success: Order is received by a customer updated by a Restaurant employee.
- Status should not be allowed to move back.

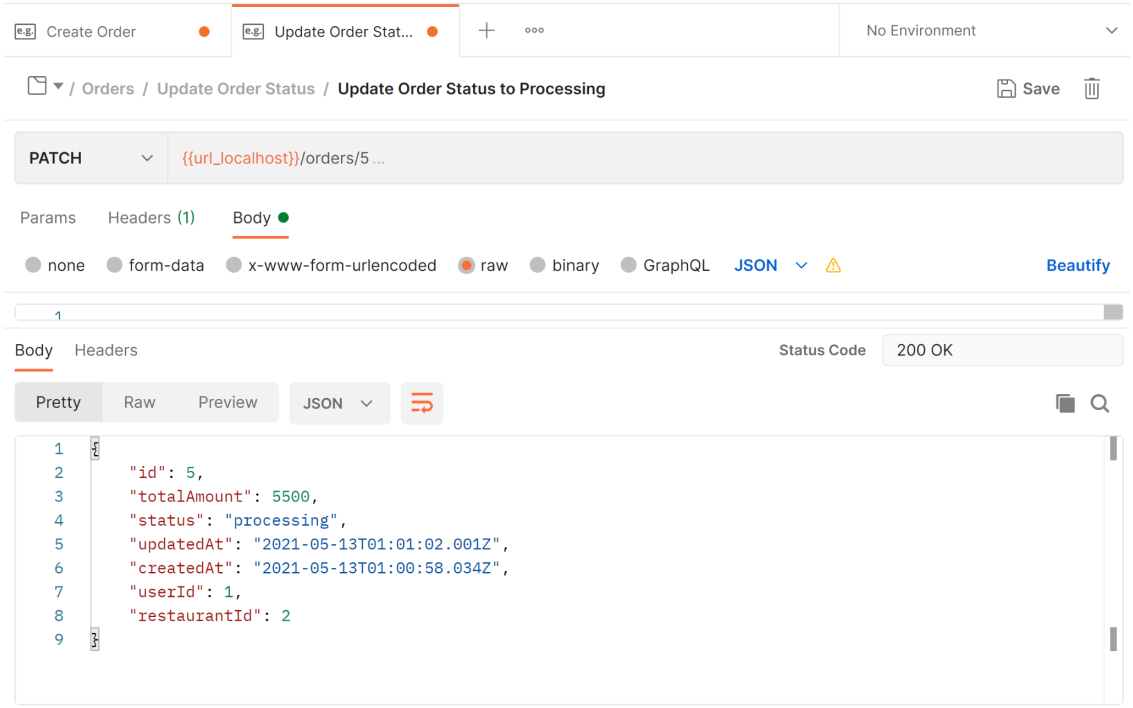
https://api.example.com/orders/:id

Request Headers:

Content-Type: application/json

Path Variables:

id: Order Id



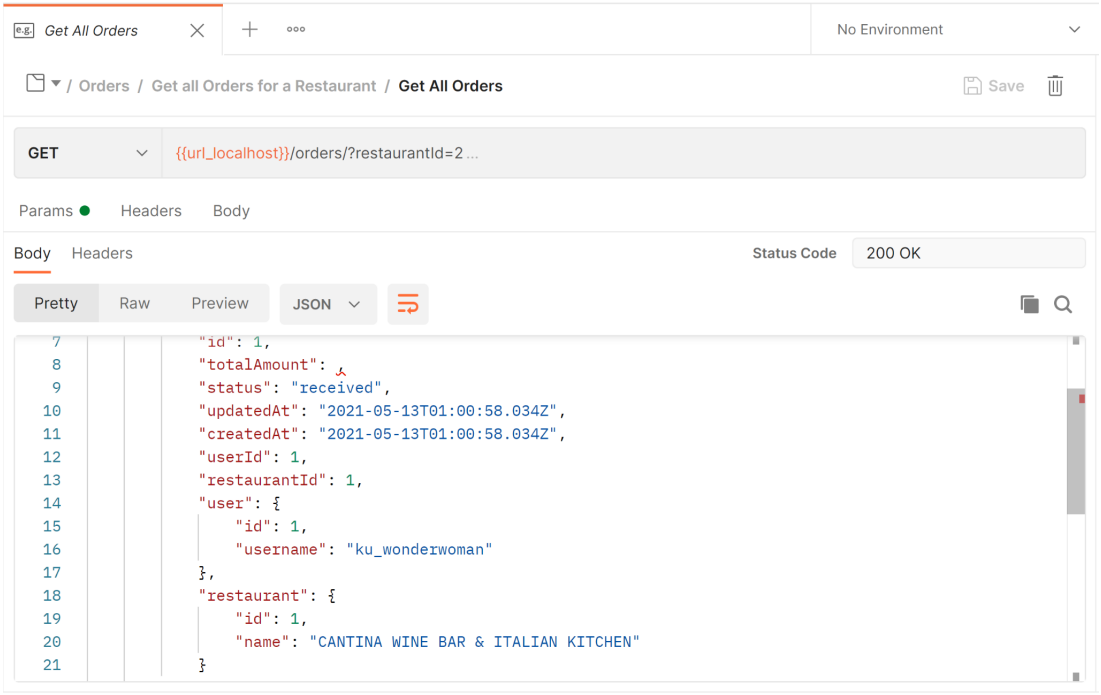
GET: get all orders for a restaurant

This API is to get all orders handled by this restaurant.

https://api.example.com/orders/?restaurantId=:id

Request Params:

restaurantId: id



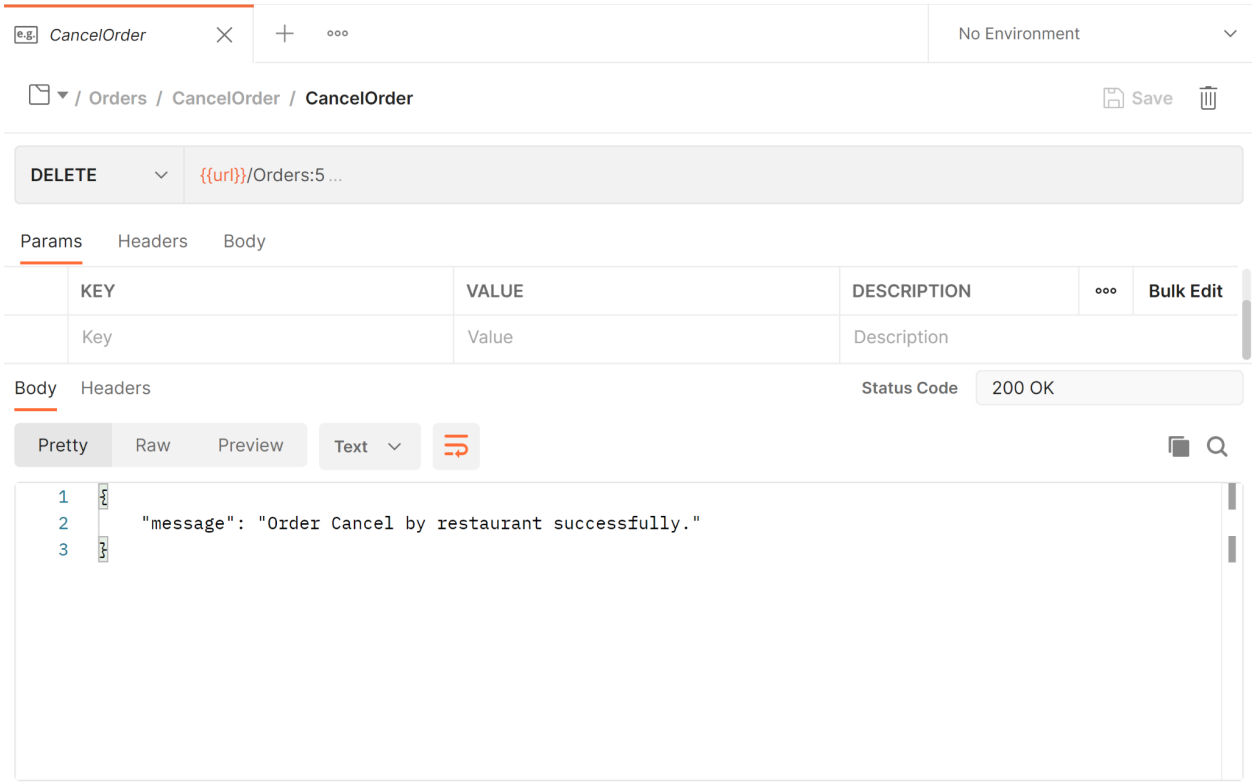
DELETE :cancelOrder

This API is used to cancel orders by customers or restaurant employees.

https://api.example.com/Orders/:order id

Path Variables:

order id: 5 (Order Id)



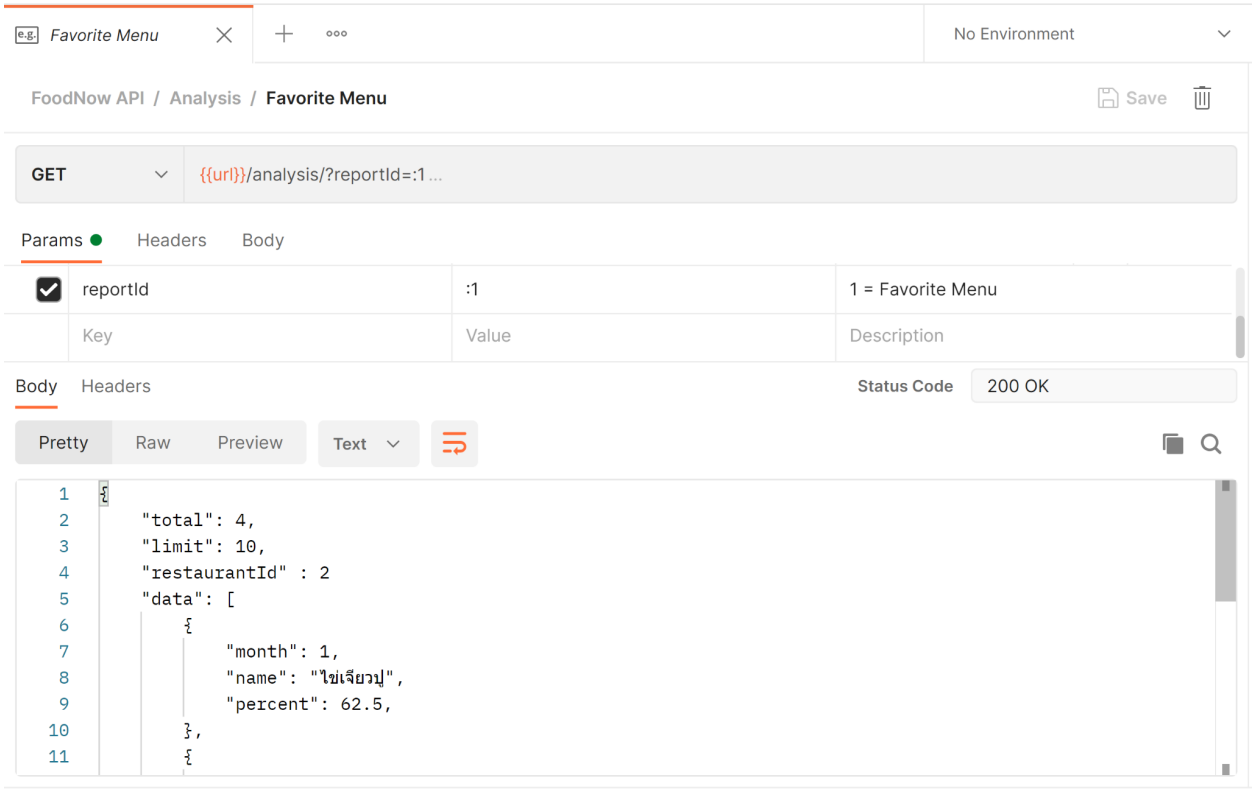
GET: Analysis

This set of API is used to generate 3 reports i.e.

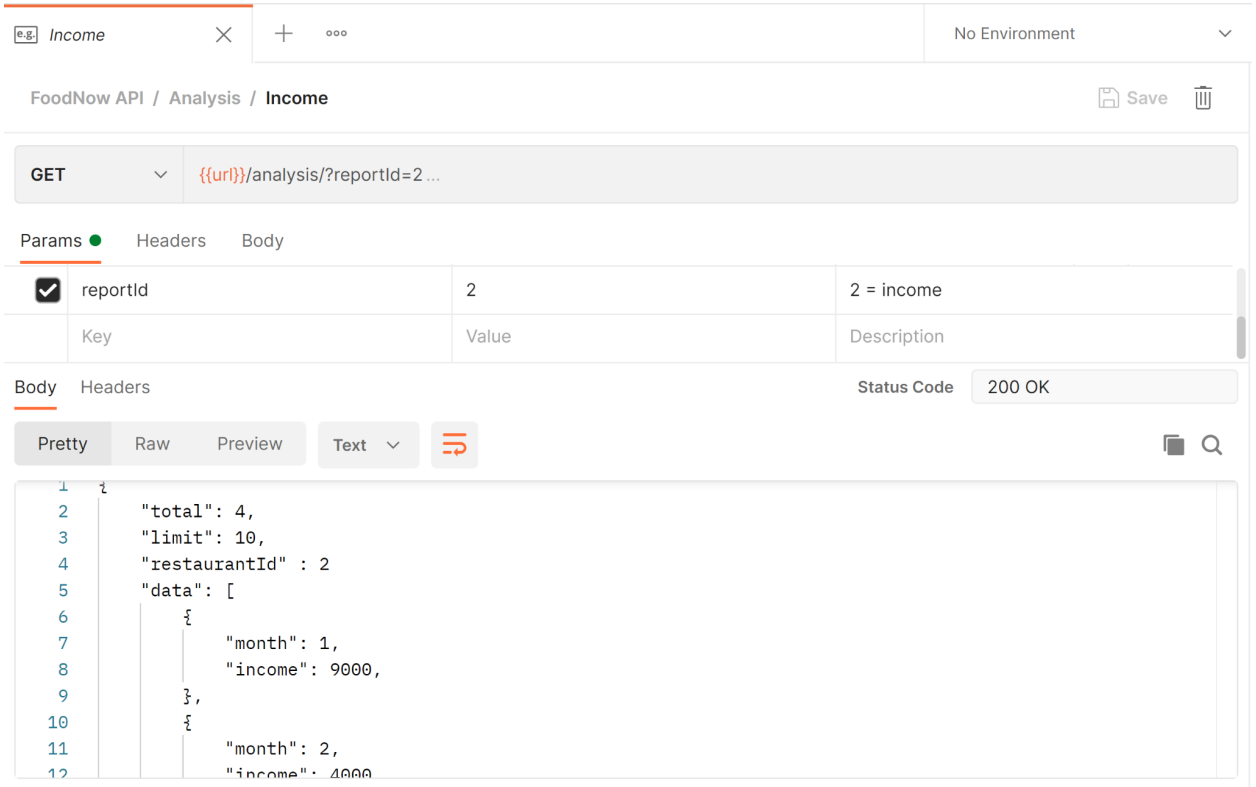
- Favorite menu.
- Income
- Frequent customers.

This report will be for a specific restaurant. It should be used with restaurant permission.

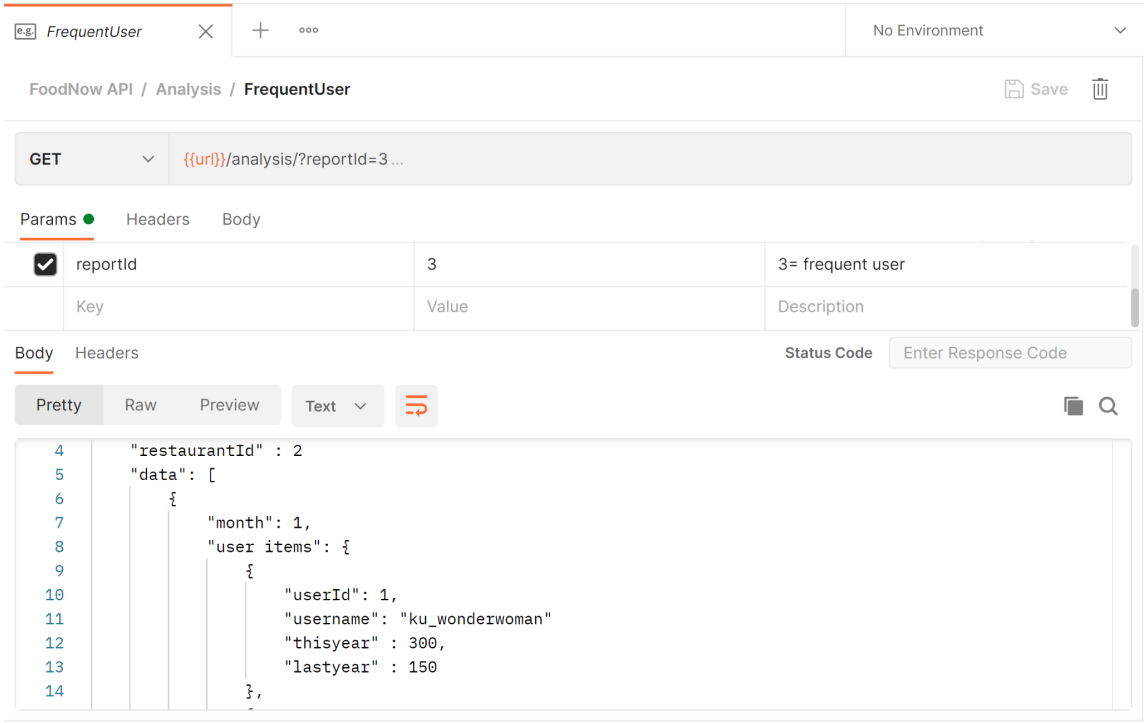
Favourite menu



Income



Frequent User



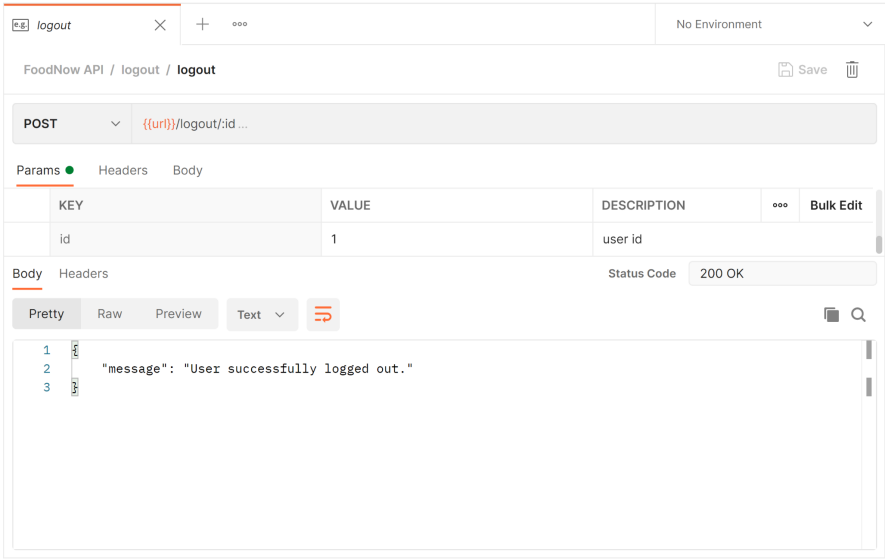
POST: log out

This API will allow users to gracefully log out.

https://api.example.com/logout/:id

Path Variables:

id: id (user id)



Link for all API images

Link to FoodNow API as json collection

System features

Use Case

Use Case I

Description
This use case describes a scenario when a hungry person orders food from this application (will be referred to as a customer).

Primary Actor
The Customer

Supporting Actors
The restaurant
The delivery employee

Stakeholder
The FoodNow developers
The customer

Pre-Conditions
Customers already have a customer account on FoodNow.

Post-Conditions
The customer's order reaches the restaurant.
The customer receives tiger orders by the delivery employee.

Main Success Scenarios

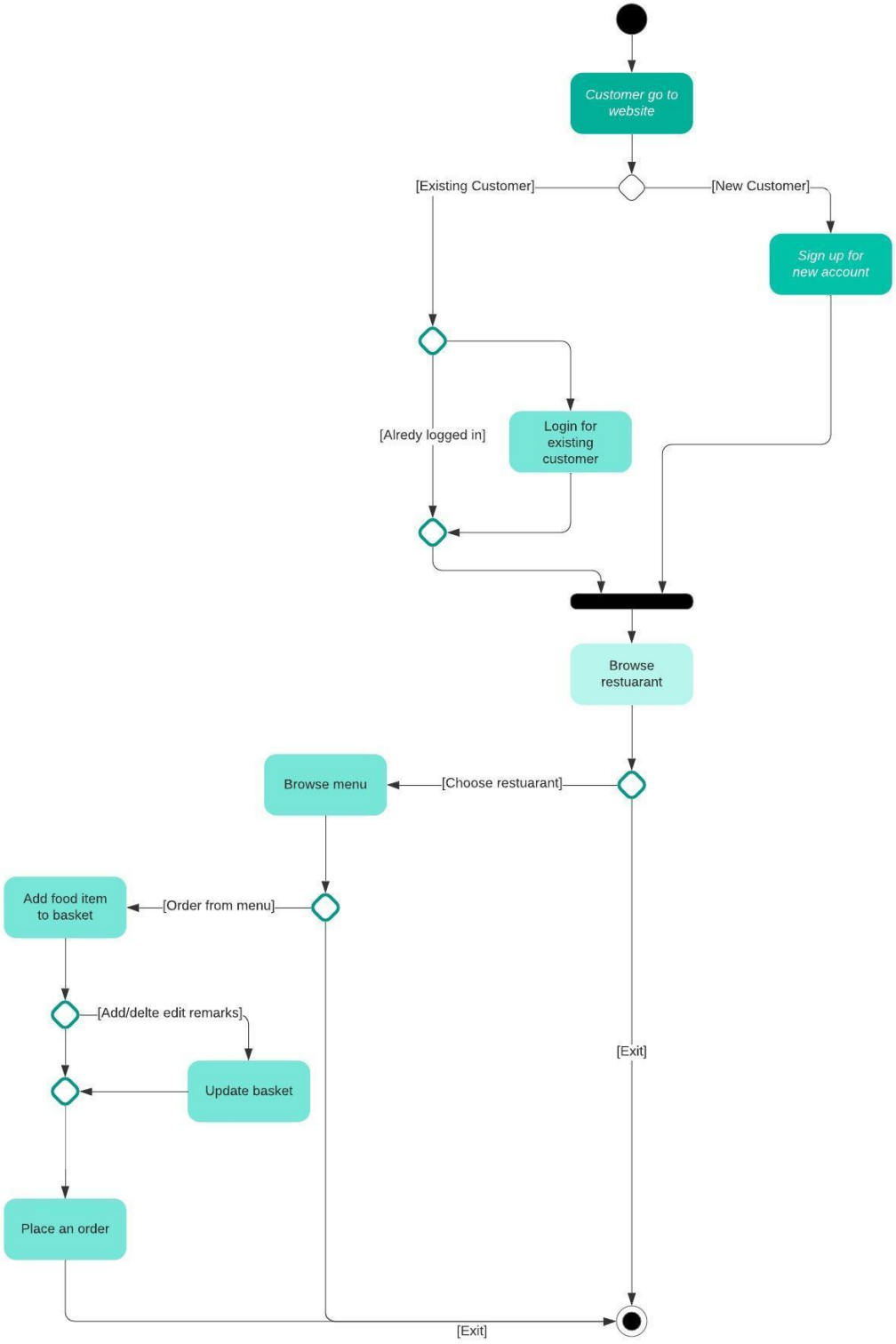
1. The customer selects their desired restaurant.
2. The customer selects their desired food items in the restaurant's menu.
3. The customer input the food item's quantity.
4. The customer may or may not add any special remarks for food items.
5. The customer clicks the 'Basket' link on the navigation bar.
6. The customer checks the Basket page if they order everything correctly.
7. The customer presses the confirm button.
8. Their order is sent to the restaurant.
9. The delivery employee comes to pick up the customer's order.
10. The customer picks up their order from the delivery employee and pays them.

Alternative Paths

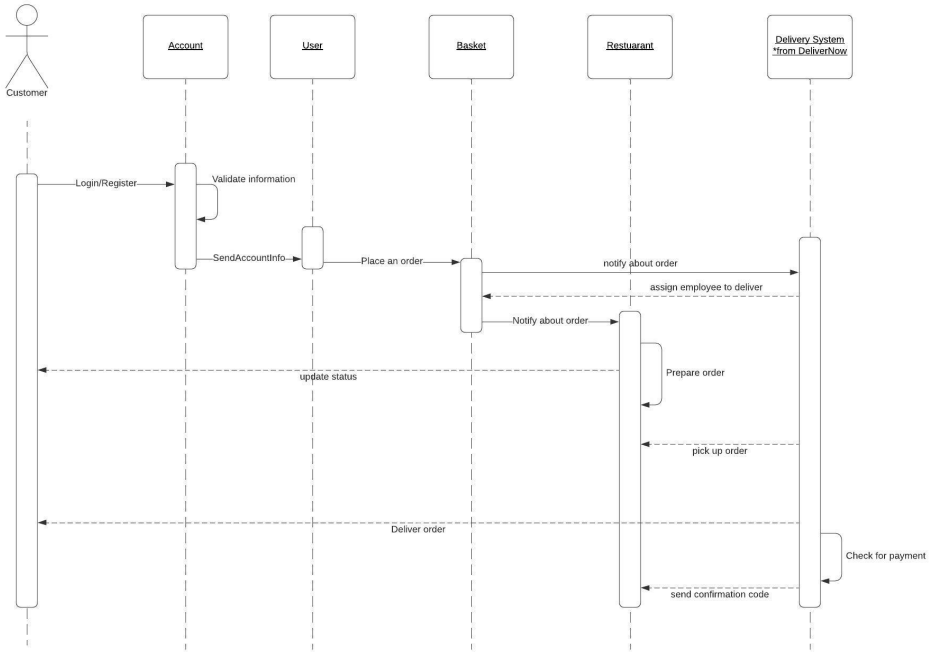
8A. The customer decides they no longer want the order and cancel it.

- System will send a cancellation notification to the restaurant.

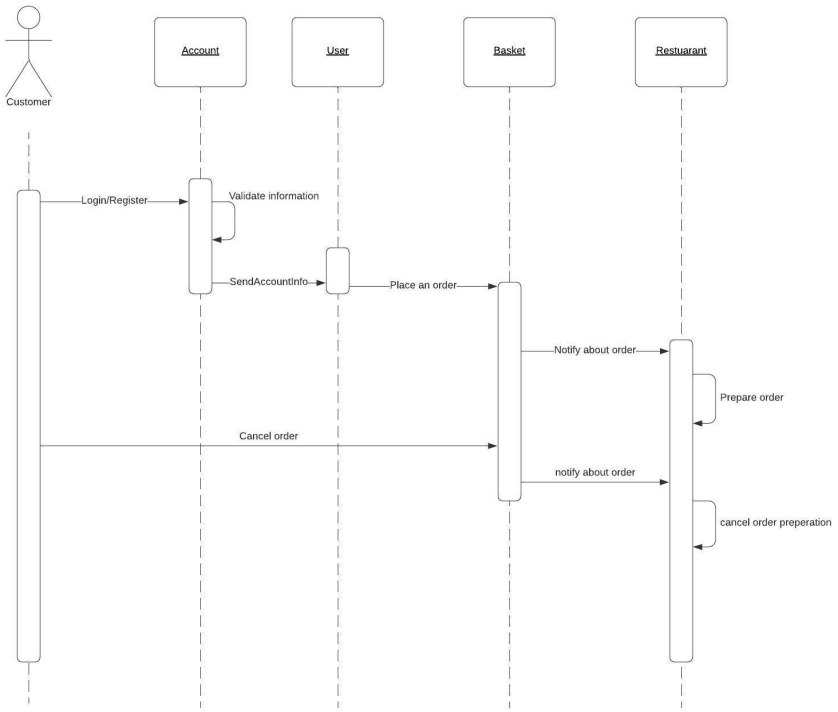
Activity Diagram for Ordering Food



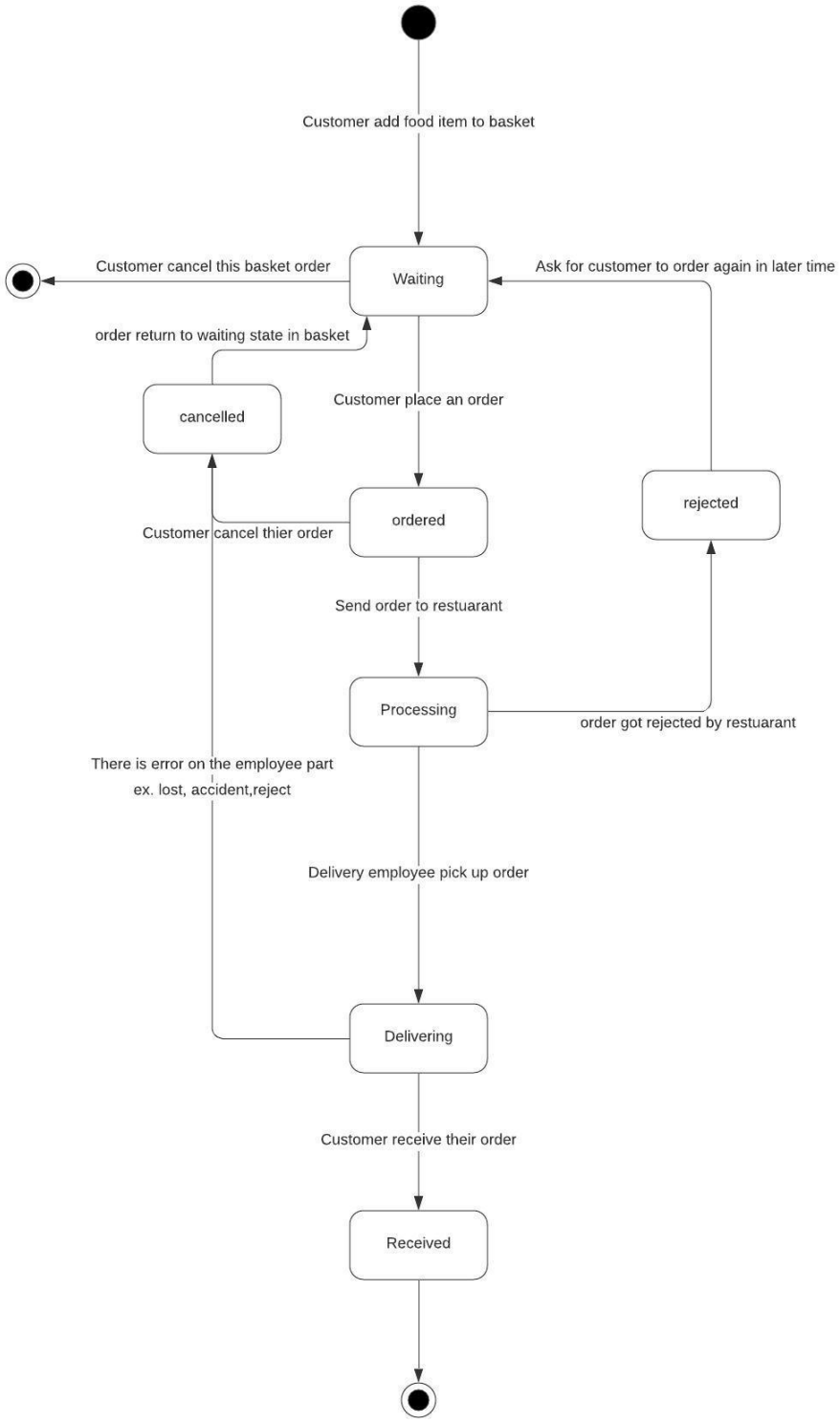
Sequence Diagram for ordering food



Sequence Diagram for alternative path 8A: cancel order



State Diagram for ordering food



Use Case II

Description
This use case describes when the restaurant owner opened their online restaurant on FoodNow and is ready for service (will be referred to as the restaurant).

Primary Actor
The restaurant

Supporting Actors
The customer
The delivery employee

Stakeholder
The FoodNow developers
The restaurant

Pre-Conditions
The restaurant has a restaurant account on FoodNow and their restaurant is already setup with menus and food items.

Post-Conditions
The restaurant receives information that the customer's order reaches the customer successfully.

Main Success Scenarios

1. The restaurant receives notification from FoodNow that there is a new order.
2. The restaurant prepares the order while FoodNow will assign the delivery employee to go to the restaurant.
3. The delivery employee picks up the order and delivers it to the customer.
4. The restaurant receives information that the customer's order reaches the customer successfully

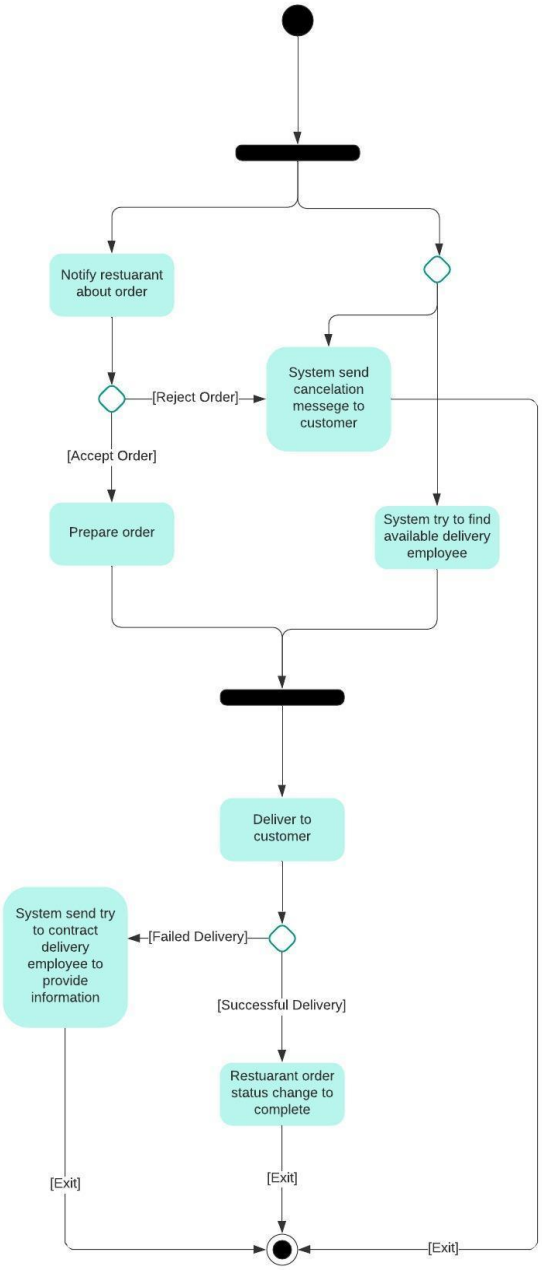
Alternative Paths

1A. The restaurant did not receive any notifications or information on the new order.
- System will send rejection notifications to the customer side.

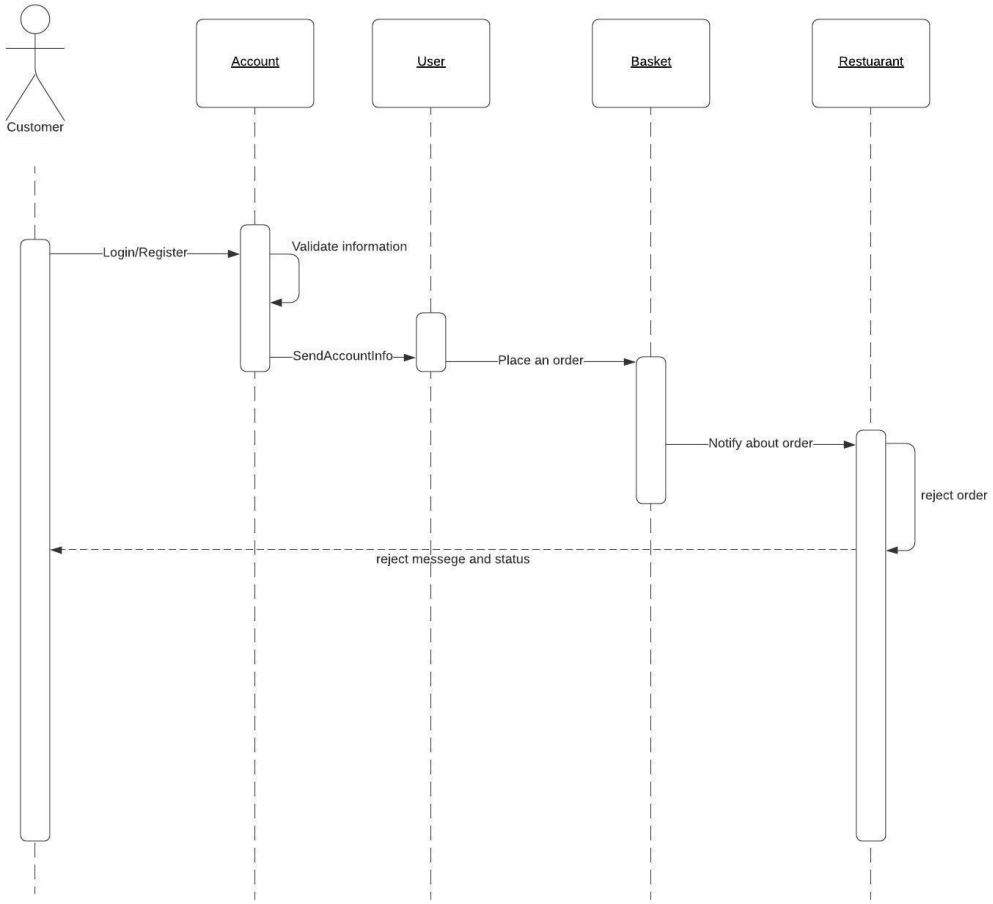
1B. The restaurant is not ready or has some other error and refuses to take this order.
- Restaurant will reject this order.

4A. The restaurant did not receive any information about the customer's order reaching its destination. (trigger an hour after passing the estimated time)
- System will try to contract the assigned employee for the current status of the order.

Activity Diagram for processing order



Sequence Diagram for alternative path 1B: order reject



Use Case III

Description
The restaurant want to check their overall page , know the past data, and check on their performance

Primary Actor
The restaurant

Stakeholder
The FoodNow developers
The restaurant

Pre-Conditions
The restaurant has a restaurant account on FoodNow.

Post-Conditions
The restaurant read their past data on FoodNow.

Main Success Scenarios

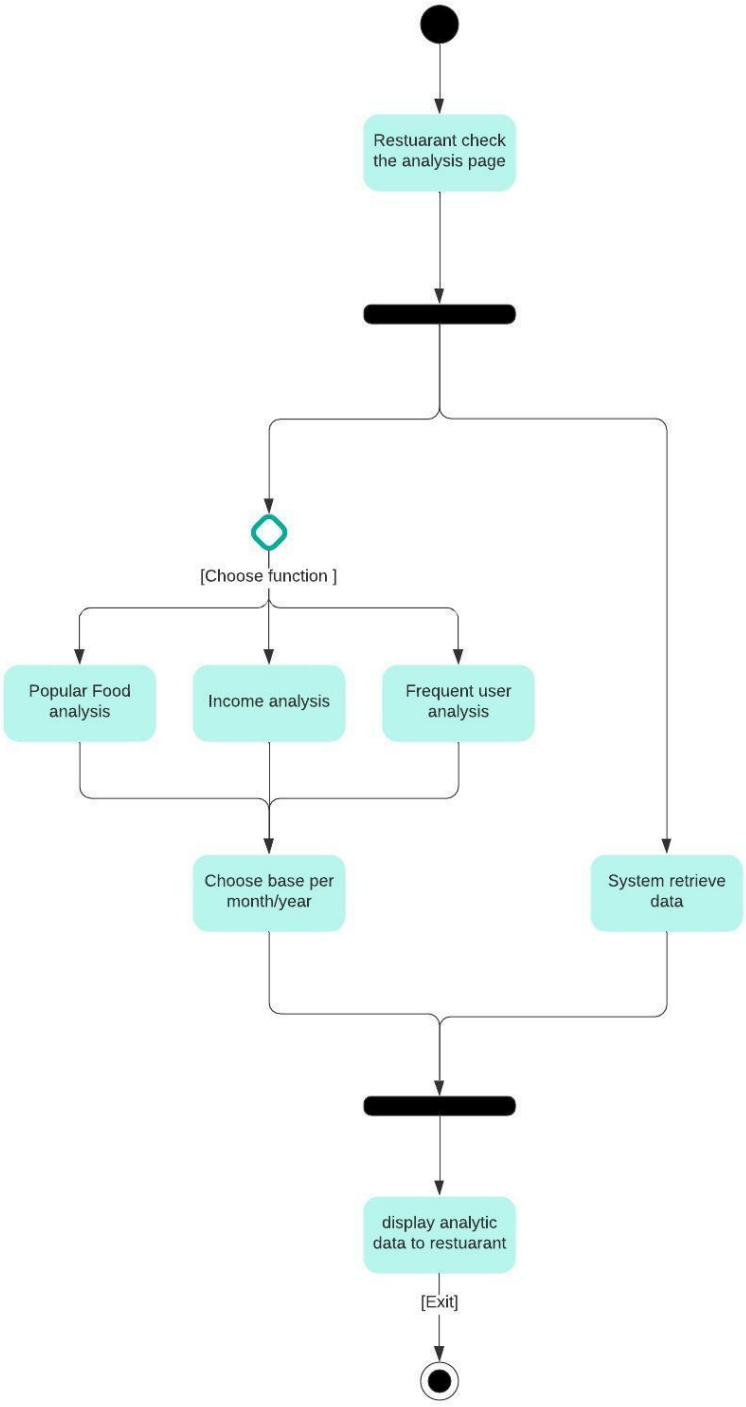
1. The restaurant login to their account home page.
2. The restaurant checks on the analysis page.
3. The restaurant chooses which from the 3 analysis functions they want to see.
4. The restaurant chooses analysis based on month or year.
5. The restaurant read their past data on the analysis page.

Alternative Paths

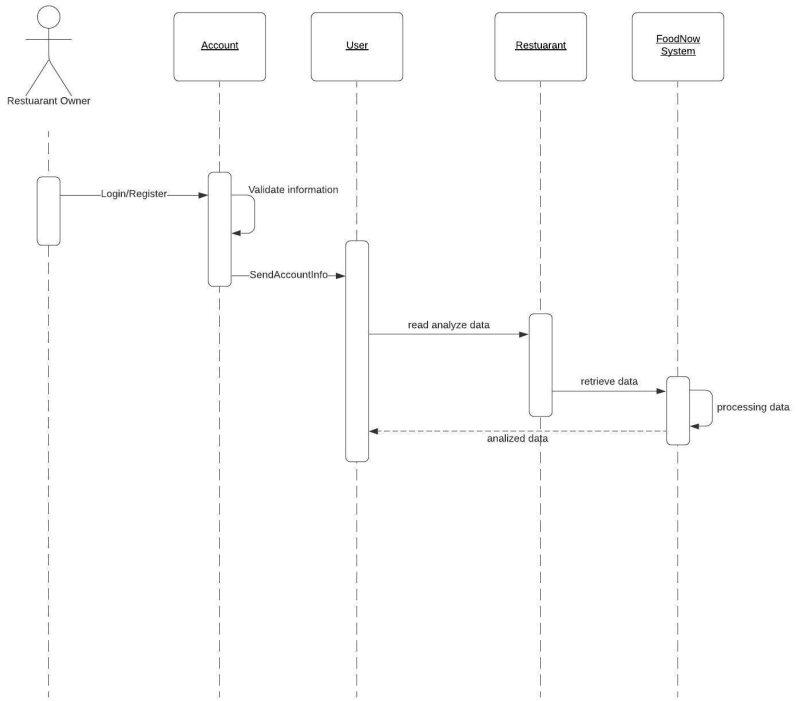
1A. The restaurant wants to add, customize, or edit their manu and food items.

- The restaurant will choose the component that they want to edit and customize it.

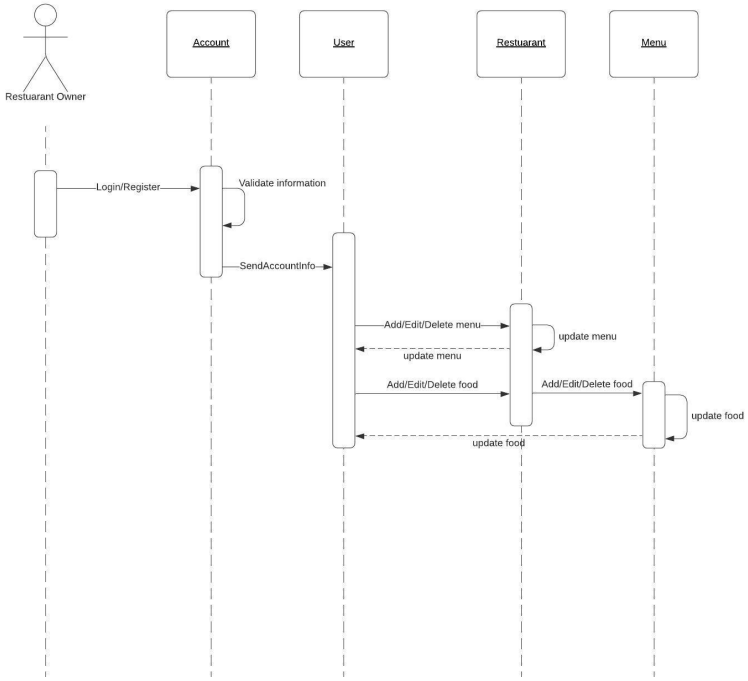
Activity Diagram for analysis function



Sequence Diagram for analysis function



Sequence Diagram for Alternative Path 1A: Edit components



Link for all activity and sequence diagrams

User Stories

- As a customer, I want to order food online, so that I don't need to go out and have it delivered.
- As a customer, I want to see some restaurant information such as name, address, shipping fee, and promotion, so that I can properly decide on the restaurant I want to order from.
- As a customer, I want to see restaurant menus, so that I can properly decide which food item I will order.
- As a restaurant owner, I want to run my restaurant service online, so that I can increase my customer base during this time of pandemic.
- As a restaurant owner, I want to check on my past performance data, so that I can use it to improve my restaurant's performance in the future.

The functional requirements for the FoodNow project based on use cases and user stories.

Customer

System feature I

Customers can interact with the menu lists and food items by clicking the cursor on their computer or single touch gesture through smartphone and tablet.

System feature II

Customers can close the menu or or food items by clicking (or touch gesture on smartphone and tablet) the provided X button.

System feature III

Customers can navigate through the list of available menus.

System feature IV

Customers can add their desired food items into their basket via click (or tap on smartphone and tablet) on that food item's add to cart button.

System feature V

Customers can add special remarks on certain food items ordered.

System feature VI

Customers can set or reset the quantity of the order in their basket.

System feature VII

Customers can track their order process and delivery.

System feature VIII

Customers can cancel their order.

Restaurant

System feature I

Restaurants can add new menus and food items into their restaurant page.

System feature II

Restaurants can edit their menus and food items by clicking (or touch gesture on smartphone and tablet) the provided ‘Edit’ button.

System feature III

Restaurants can delete their menus and food items by clicking (or touch gestures on smartphone and tablet) on the button on the left of each food item and menus.

System feature IV

Restaurants can receive notifications of order from customers.

System feature V

Restaurants can receive some information about the employee assigned to each order such as the name of the employee that this order is assigned to.

System feature VI

Restaurants can cancel orders.

System feature VII

Restaurants can use the analysis function to see their most popular food item each month, year.

System feature VIII

Restaurants can use the analysis function to see their raw income from customer payment each month, year.

System feature IX

Restaurants can use the analysis function to see the frequency of each customer that orders from it.

System feature X

Restaurants can see all their past orders within some time constraint (ex. Within 1 month, 3 weeks, etc.).

Non functional requirements

Performance requirements

- This project should be able to support at least 500 simultaneous users. The capacity may be expanded in the future version.
- All duration lags for each click to load up components must be within 1 second.

Safety requirements

- The system should log every state and change and display to provision recovery from system failure.
- The system should display menu lists and food items at all times to prepare for any circumstance that the customer may request.
- The system should restore itself when the previous self has any malfunctions or failures.

Security requirements

- Password should have a mix of characters and numbers with at least 8 characters long.

Software quality requirements

Availability

- The system should be available at all times to accept any customers orders.

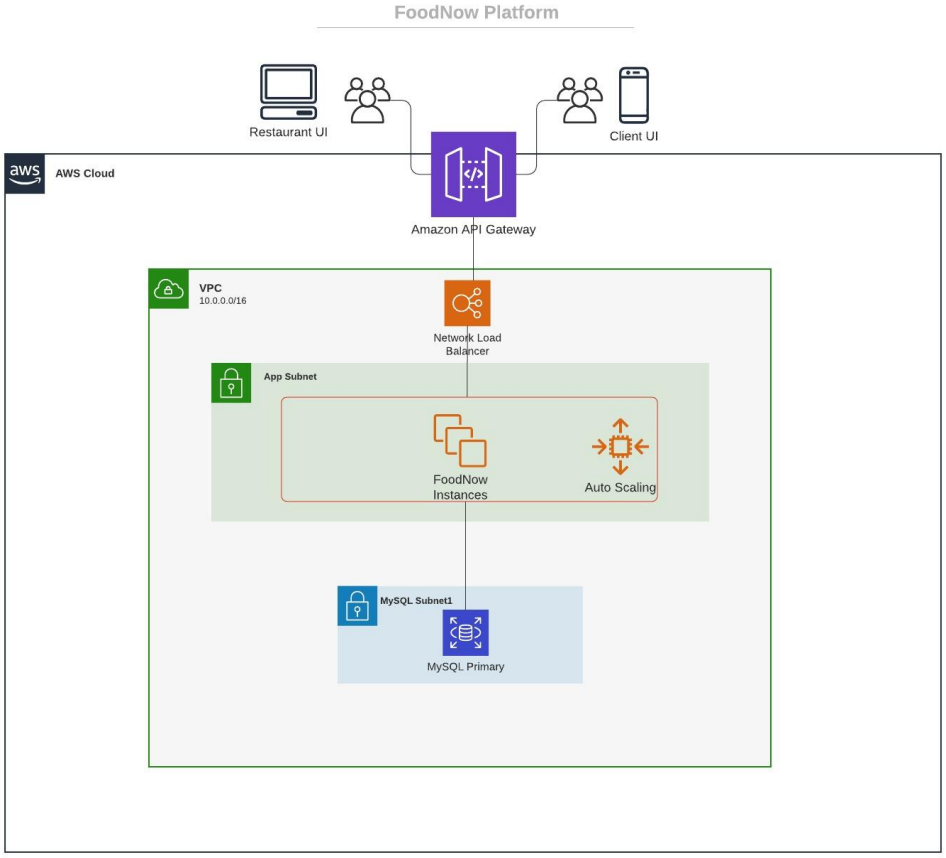
Data Integrity

- The order from the customers end to the restaurant end should be the exact same order.
- The order must be sent to the correct destination.
- The information of previous customers, orders, and delivery should be correct when retrieved.

Maintainability

- New features should be easy to implement with separation of concern

System Infrastructure Design



Link for better quality image

FoodNow Platform is hosted on AWS cloud. This design will allow FoodNow to support scalable and high availability requirements. This enables the FoodNow platform to be set up and deployed very instantly.

As shown in the above figure, FoodNow platform is set up in AWS and uses a number of AWS technologies to allow FoodNow platform to achieve non-functional requirements. Those technologies are listed below

AWS virtual private cloud (VPC)

By using VPC , this allows the FoodNow platform to have complete control over the environment, including creating subnet and network gateway. This is helping to ensure secure and easy to access FoodNow backend services..

AWS API Gateway

FoodNow platform is using REST API as a means to connect FoodNow UI and FoodNow backend services. By using AWS API Gateway, this technology will handle all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management.
This has lifted a lot of heavy work from FoodNow development team and enabled the team to focus on business logic and faster time to market.

Amazon Relational Database Service (Amazon RDS)

RDS is a managed service that allows the FoodNow development team to set up, operate, and scale MySQL on AWS easily and quickly. This service enables the FoodNow development team from time-consuming MySQL administration tasks and free up FoodNow team to focus on business logic only.

AWS Network Load Balancing

Network Load Balancer will automatically distribute incoming traffic to FoodNow backend across multiple FoodNow EC2 instances. It can automatically scale to the vast majority of workloads. This will ensure that FoodNow platform gives the best performance.

Amazon Elastic Compute Cloud (Amazon EC2)

EC2 is a web service that provides compute capacity that FoodNow will use to host FoodNow platform backend services. The image on EC2 will use Ubuntu. FoodNow also enables AutoScaling groups to add or remove EC2 instances based on the current load.