

# Software Requirements Specification

*OPENJIO: Invitation can never be simpler*

Lim Jun Weng	A0139722A
Rhynade See Ey	A0142897N
Tan Jin Wei	A0139100X
Tan Qing Yang	A0139508Y

## 1. Introduction

Existing reservation apps often have the additional function of allowing users to pre-order their food together with their reservations. However, when making a reservation for a large group of people, pre-ordering food becomes a hassle, requiring the person making the reservation to have to do additional work in collating the orders from different users. The problem increases in severity as the number of people in the group increases. Additionally, any members of the group with special requirements for their food will also add to the hassle of the person placing the order.

Openjio was envisioned as an app to simplify the process of pre-ordering food for a group of people, while still maintaining the core function of allowing users to make reservations. Openjio helps the user to create order baskets, and provides the option of sharing the basket so that other users in the group can make their own personal additions. All the orders from the different users will then be calculated under one reservation. Hence, the individual placing the order will only need to create the central basket, and share it with other members of the group through email. The need for collating orders and special requests is removed.

### 1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the Openjio software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and as a reference for developing the first version of the system for the development team.

This document consists of a total of 4 sections, including this one.

Section 2 will cover the functional overview of the application, as well as the interaction between different systems. The constraints and assumptions about the product will also be discussed in this chapter, together with the system interactions with different stakeholders.

Section 3 will cover the detailed system requirements as well as describe the various system interfaces available for different types of users.

The final section will cover the feature prioritization of the app in the event that the full, finished product cannot be delivered within the specified deadline.

## 1.2 Definitions, acronyms, and abbreviations

Users	Refers to the global set including all users
Client	Refers to restaurant customers using the app to make a reservation
Creator	Refers to the client who created an order basket by making a reservation
Restaurants	Refers to users from the restaurants that maintain the menu
Admin	Refers to overall administrators that govern the functionality of the app
DESC	Feature description
RAT	Rationale for feature
DEP	Dependencies on other features
ID	A tag used to identify functional requirements
GIST	A short, simple description of the concept
SCALE	The scale of measure used by the requirement
METER	The process or device used to establish location on a SCALE
MUST	The minimum level required to avoid failure
PLAN	The level at which good success can be claimed
WISH	A desirable level of achievement that may not be attainable through available means

## 2. Overall description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

### 2.1 Product perspective

The system consists of a web application that provides different functions based on the type of user logged in, as well as multiple databases for storing user information, order information, and menu information. Clients will be able to create orders and invite other clients to edit orders. Restaurants can use the app to edit the menu, view all existing orders, and view the dashboard visualization of data collected from the orders.

Each user will be assigned a role, which will be used to decide which functions they will have access to. The role is stored as a variable in the user database. All orders are stored in a central database, each tagged with the clients in that order. Clients can make changes to orders they are tagged in, and can invite other clients into the order. However, they will only be able to view orders they are tagged in. Restaurants will be able to view all orders, but will not be able to make changes to any of them.

The menu is also stored in a database. Only restaurants will be allowed to make changes to the menu database, such as adding or removing a menu item. The client will be able to view the menu items and add them to their order, but will not be able to make any changes.

### 2.2 Product functions

OpenJio is a web application, where users will get access to different functionalities based on their assigned roles as a client or as a restaurant.

A client can initiate a reservation by selecting the time and number of people. Once the reservation is made, an order basket is created, and the creator can then see the menu items, displayed as tiles. The creator is given the option to add items to his order basket, and is able to select the quantity desired, as well as special requests. The creator can view his own order basket at any time.

When viewing the order basket, the creator can then choose to share the basket with a number of other clients by sending them a unique invitation link. With the invitation link, other clients should be able access the same menu view, and add menu items similarly to a shared basket. The creator of the basket should also be able to see additions made to his basket.

When a client in a basket decides that the order is completed, he or she can finalize the order to the restaurant.

A restaurant will get access to different functions of the app, and should be able to make changes to the menu, as well as view the collection of finalized orders and reservations made by the clients. Additionally, a restaurant can get access to a visualization tool that enables them to track the popularity of various food items, as well as customer trends.

## 2.3 User Characteristics

There are three types of users that interact with the system: clients, restaurant owners and administrators. Each of these three types of users has different use of the system so each of them has their own requirements.

The clients of the application can only use the app to make a reservation, place orders, and invite other clients into an existing order. Hence, they will need to be able to create an order basket, view the menu items, add menu items to their order basket, and invite other clients who can then add menu items to the same order basket.

The restaurants will log in through the same interface, but will get access to a different set of functions. Since this is primarily a reservation app, restaurants need to be able to see all existing confirmed reservations and orders. For obvious reasons, they should not be able to edit the orders as well. Since the menu is dependant on the restaurant, the restaurant should be able to edit the menu to remove outdated items and update new items. Lastly, restaurants should get access to a dashboard visualisation tool that allows them to track the popularity of their menu items, as well as the distribution of reservations through the day.

The administrators manage the overall system so there is no incorrect information within it. The administrator can manage the information for each restaurant as well as for each user.

## 2.4 Constraints

Since the ordering takes place through the web, the app requires all users making an order to have a functioning Internet connection to place an order.

The web application will be constrained by the capacity of the database. Since all users will be accessing the same common database for menu items and orders, the database may have to queue incoming requests, which will increase the time needed to process each request.

The application was developed primarily as a web application - accessing the site on through a mobile platform, while it should not affect the functionality, might affect the overall layout and navigation within the site.

## 2.5 Assumptions and dependencies

The functionality of the site is dependant on the clients each having a functioning email address, since the sharing function will be linked through the client's' emails.

### 3. Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

#### 3.1 External Interface Requirements

This section provides a detailed description of all inputs and outputs of the system. It also gives a description of the software, hardware and communication interfaces and provides basic prototypes of the user interface.

##### 3.1.1 Client Interface

There are two ways in which a client can enter the website:

1. Normal URL
2. Invited URL

###### *Normal URL*

The client will see the login page where he/she is prompted to enter his/her email and password upon entering the website (Figure 3.1).

If he/she has not registered, he/she should be able to register on the site. After logging in, the client should be prompted to key in the details for a reservation (Figure 3.2).

Once a client clicks on the create order button, an order code should be generated and he/she should be directed to the menu page. (Figure 3.3)

###### *Invited URL*

The client should see the login page where he/she is prompted to enter his/her email and password upon entering the website via the invited URL (Figure 3.1).

If he/she has not registered, he/she should be able to do that on the login page. After logging in, the client should be directed to the menu page right away, with the details of the reservation already keyed in. (Figure 3.3)

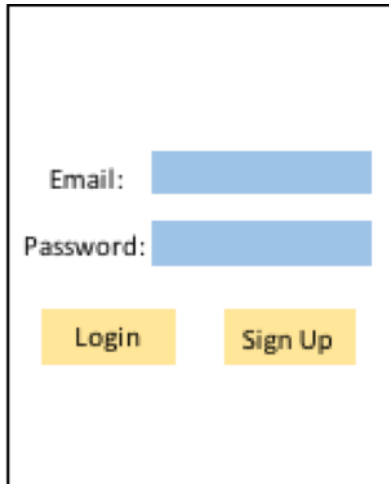


Figure 3.1 - Login Page

The login page features two input fields: 'Email:' and 'Password:', each followed by a blue rectangular text box. Below these fields are two yellow rectangular buttons labeled 'Login' and 'Sign Up'.

Figure 3.1 - Login Page

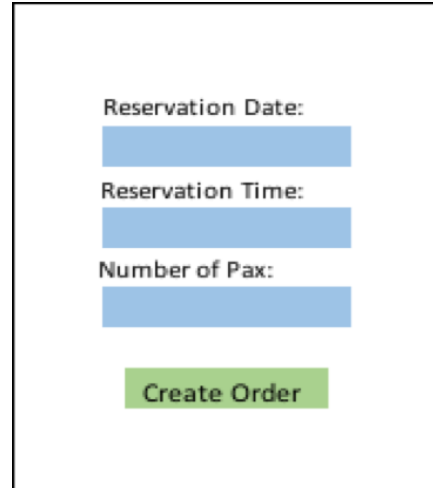


Figure 3.2 – Reservation Page

The reservation page includes three input fields: 'Reservation Date:', 'Reservation Time:', and 'Number of Pax:', each followed by a blue rectangular text box. A green rectangular button labeled 'Create Order' is positioned at the bottom.

Figure 3.2 – Reservation Page

The menu page consists of tabs on the left and top right of the window as seen in Figure 3.3. Clients should be able to click on those tabs which should direct them to the respective pages.

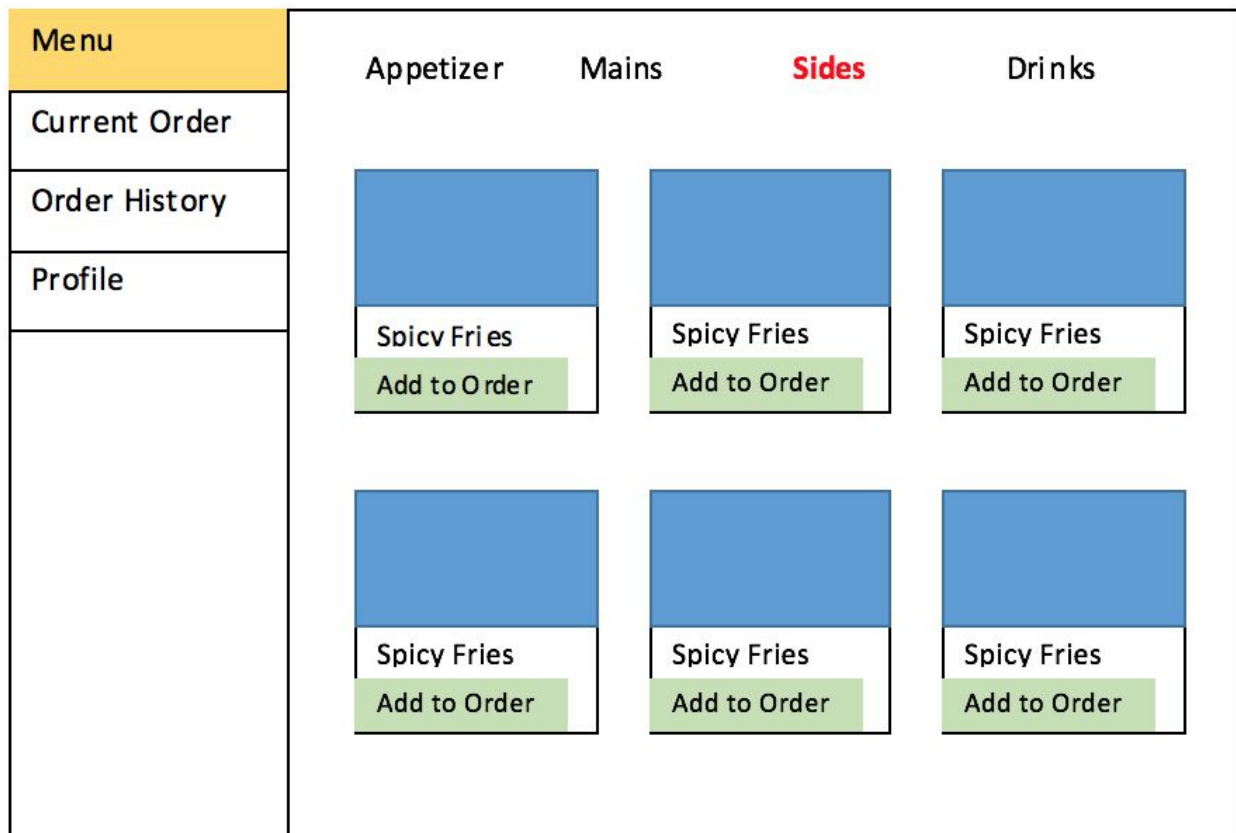


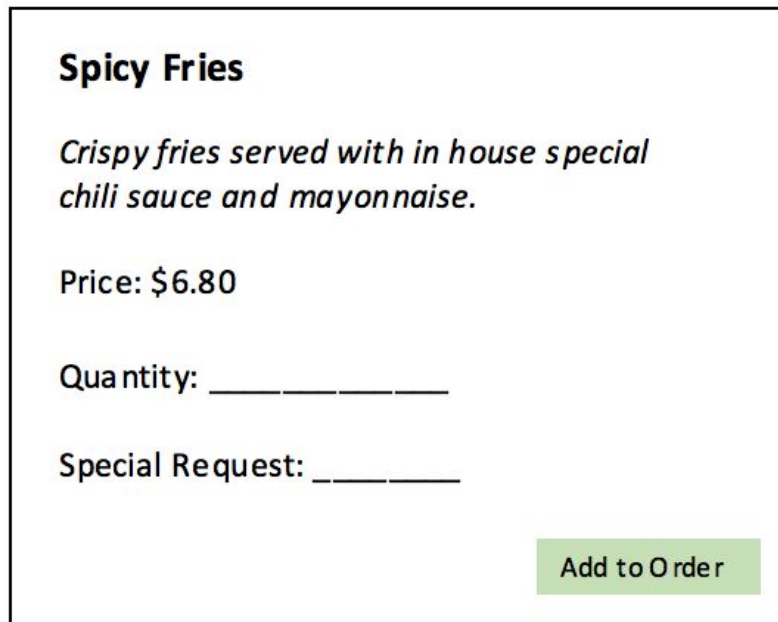
Figure 3.3 – Menu Page

The menu page is divided into a left sidebar and a main content area. The sidebar contains a yellow 'Menu' header and three white buttons: 'Current Order', 'Order History', and 'Profile'. The main content area has four category tabs: 'Appetizer', 'Mains', 'Sides' (highlighted in red), and 'Drinks'. Below these tabs, there are two rows of items. Each item consists of a blue placeholder image, the text 'Spicy Fries', and a green 'Add to Order' button.

Figure 3.3 – Menu Page

A client should be able to select the food items they wish to order on the menu page. When he/she clicks on a food item, a modal should appear. The modal should show the name, description, and price of the food item. He/she would have to fill in the quantity and special request (optional) before adding the food item to his/her order (Figure 3.4).

Upon clicking on add to order, the modal should disappear and the client should be brought back to the menu page with the item added to his/her order.



**Spicy Fries**

*Crispy fries served with in house special chili sauce and mayonnaise.*

Price: \$6.80

Quantity: \_\_\_\_\_

Special Request: \_\_\_\_\_

Add to Order

Figure 3.4 – Modal



Current order should display the details of all the orders (Figure 3.5). Clients should be able to invite their friends to start the collaborative ordering process by inputting their friends email after clicking on the invite link placed beside the order number.

Clients should also be able to confirm their order by clicking on the place order button and to cancel their order by clicking on the cancel order button.

Menu	<div> Order Number: M7ivRzJyk <a href="#">Invite</a>  Reservation Date: 15 Oct 2016, 2pm  Number of Pax: 4   <a href="#">rhy@gmail.com</a>  Spicy Fries x 1 \$6.80  <i>Special Request: More sauce</i>   <a href="#">jw@gmail.com</a>  Salmon Pasta x 1 \$12.90   <a href="#">qy@hotmail.com</a>  Fish and Chips x 1 \$10.90   <a href="#">jw@hotmail.com</a>  Apple Juice x 2 \$7.00   <div>Place Order Cancel Order</div> \$37.60 </div>
Current Order	
Order History	
Profile	

Figure 3.5 – Current Order

Order history should display a list of all the past orders of the client (Figure 3.6).

Menu	<div> Order Number: M7ivRzJyk  Reservation Date: 15 Oct 2016, 2pm  Number of Pax: 4   <a href="mailto:rhy@gmail.com">rhy@gmail.com</a>  Spicy Fries x 1 \$6.80  <i>Special Request: More sauce</i>   <a href="mailto:jw@gmail.com">jw@gmail.com</a>  Salmon Pasta x 1 \$12.90   <a href="mailto:qy@hotmail.com">qy@hotmail.com</a>  Fish and Chips x 1 \$10.90   <a href="mailto:jw@hotmail.com">jw@hotmail.com</a>  Apple Juice x 2 \$7.00   <div>Order Confirmed</div> <div>\$37.60</div> </div>
Current Order	
Order History	
Profile	

Figure 3.6 – Order History

The profile page should allow clients to update his/her details (Figure 3.7).

Menu	<div><div>First Name: _____ Last Name: _____</div><div>Gender: M/F</div><div>Address: _____</div><div>Update</div></div>
Current Order	
Order History	
Profile	

Figure 3.7 – Profile Page

### 3.1.2 Restaurant Interface

The restaurant owner should see the login page where he/she is prompted to enter his/her email and password upon entering the website (Figure 3.1).

After logging in, the restaurant owner should be directed to the menu page (Figure 3.8). However, the tabs on the left should be different from that of a client interface. The restaurant owner should also be able to delete any existing dishes from the menu page.

Menu	Appetizer	Mains	Sides	Drinks
Orders				
Add Dishes				
Dashboard				
	<div>Spicy Fries</div> <div>Delete</div>	<div>Spicy Fries</div> <div>Delete</div>	<div>Spicy Fries</div> <div>Delete</div>	
	<div>Spicy Fries</div> <div>Delete</div>	<div>Spicy Fries</div> <div>Delete</div>	<div>Spicy Fries</div> <div>Delete</div>	

Figure 3.8 - Menu Page (Restaurant)

Orders should display the details of all the confirmed reservation made by clients (Figure 3.9).

Menu	<div> Order Number: M7ivRzJyk  Reservation Date: 15 Oct 2016, 2pm  Number of Pax: 4    <a href="mailto:rhy@gmail.com">rhy@gmail.com</a>  Spicy Fries x 1 \$6.80  <i>Special Request: More sauce</i>    <a href="mailto:jw@gmail.com">jw@gmail.com</a>  Salmon Pasta x 1 \$12.90    <a href="mailto:qy@hotmail.com">qy@hotmail.com</a>  Fish and Chips x 1 \$10.90    <a href="mailto:jw@hotmail.com">jw@hotmail.com</a>  Apple Juice x 2 \$7.00 </div>
Orders	
Add New Dishes	
Dashboard	

Figure 3.9 – Orders (Restaurant)

Restaurant owners should be able to add in additional dishes into their menu by selecting the add new dishes tab (Figure 3.10). To successfully add in a new dish, all the fields have to be filled in.

Menu	<b>Select Category</b> <input type="radio"/> Appetizer <input type="radio"/> Mains <input type="radio"/> Sides <input type="radio"/> Drinks  <b>Name</b> <input type="text"/>  <b>Price</b> <input type="text"/>  <b>Description</b> <input type="text"/>  <b>Image</b> <input type="text"/>  <input type="submit" value="Submit"/>
Orders	
Add New Dishes	
Dashboard	

Figure 3.10 – Add New Dishes

The dashboard page should contain relevant charts plotted by aggregating data from the database (Figure 3.11).

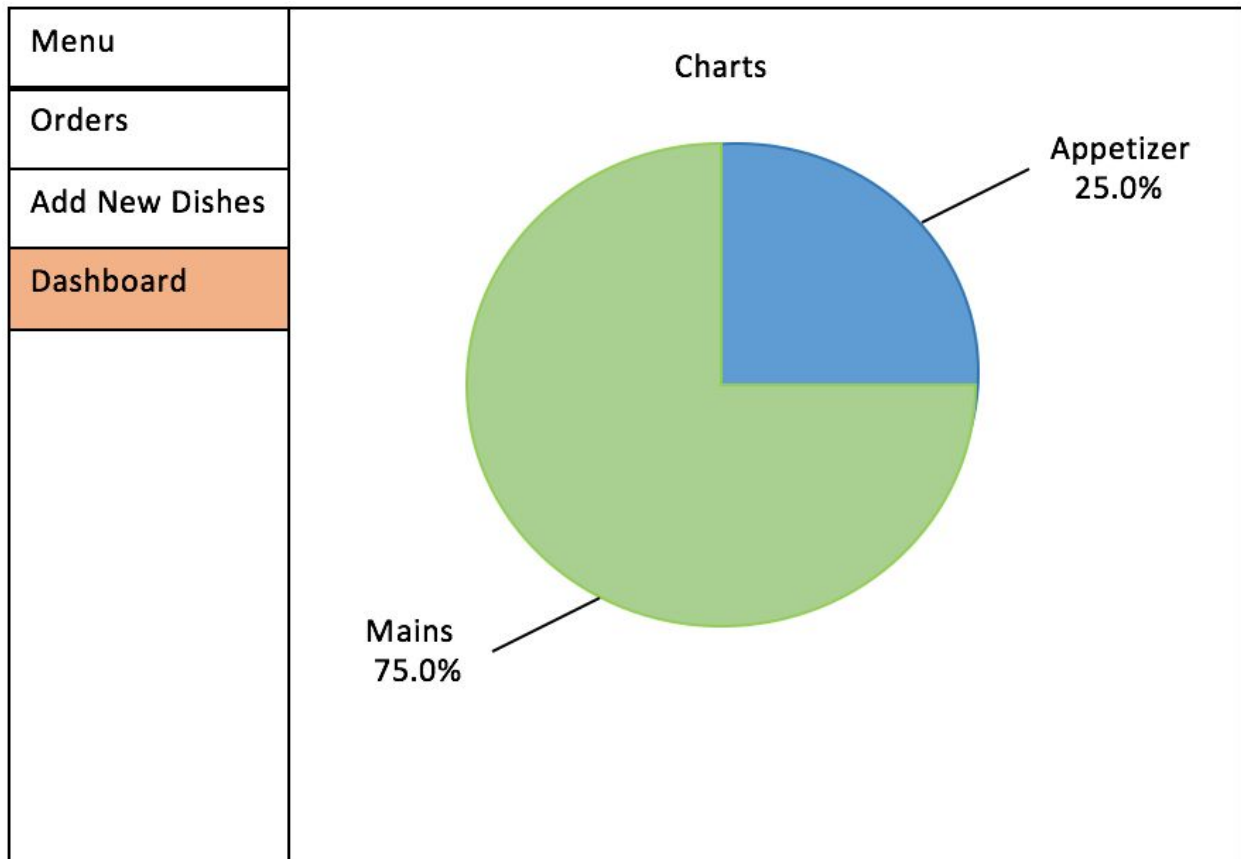


Figure 3.11 – Dashboard

### 3.1.3 Hardware Interfaces

Since this web application does not have any designated hardware, it does not have any direct hardware interfaces.

### 3.1.4 Software Interfaces

The communication between the database and the web application consists of operations that include reading and modifying the data. The level of communication should be dependant on whether the user is logged in as a restaurant, or as a client.

## 3.2 Functional Requirements

This section includes the requirements that specify all the fundamental actions of the software system.

### 3.2.1 User Class 1: The Client

#### 3.2.1.1 Functional requirement 1.1

**ID:** FR1

**GIST:** User registration

**DESC:** Given that a user has launched the web application, the user should be able to register through the web application. The user must provide an email and password. The user can update their profile after login to include more information.

**RAT:** Registration provides the identification and allows the server to store the key information into the database.

#### 3.2.1.2 Functional requirement 1.2

**ID:** FR2

**GIST:** User login

**DESC:** Given that a user has registered, the user should be able to log in to the web application.

**RAT:** This allows the user to interact with the web application upon identification.

**DEP:** FR1

#### 3.2.1.3 Functional requirement 1.3

**ID:** FR3

**GIST:** User logout

**DESC:** Given that a user has logged in, the user should be able to logout of the web application.

**RAT:** This allows the user to sign out.

**DEP:** FR1 and FR2



#### 3.2.1.4 Functional requirement 1.4

**ID:** FR4

**GIST:** Retrieve password

**DESC:** Given that a user has registered and is in a situation where he/she forgets his/her password, the user should be able to reset his/her password by the link sent to via email.

**RAT:** A solution that allows user to be able to login the web application after forgetting their password.

**DEP:** FR1

#### 3.2.1.5 Functional requirement 1.5

**ID:** FR5

**GIST:** Change password

**DESC:** Given that a user has registered and in a situation where he/she feels the need to change his/her password, the user should be able to change his/her password by verifying the old password and inputting the new password into the respective fields.

**RAT:** This allows user to change their password.

**DEP:** FR1 and FR2

#### 3.2.1.6 Functional requirement 1.6

**ID:** FR6

**GIST:** Profile page

**DESC:** In the web application, the side nav bar has a tab that directs users to the profile page. On the profile page a user can edit his/her information, which includes the name, gender and address.

**RAT:** To allow for an easy first-time registration and the flexibility for a user to update his/her profile page on the web application in future.

**DEP:** FR1 and FR2

#### 3.2.1.7 Functional requirement 1.7

**ID:** FR7

**GIST:** Create an order basket

**DESC:** In the web application, the side nav bar has a tab that directs clients to the create new order basket page. The client should be able to create a new order basket that specifies the reservation date, reservation time and the number of diners.

**RAT:** This allows the client to add order items from the menu into the order form and collaborate with other clients.

**DEP:** FR1 and FR2

### 3.2.1.8 Functional requirement 1.8

**ID:** FR8

**GIST:** Collaboration of order basket among many clients

**DESC:** In the order form, the client should be able to invite anyone to add their orders to the same order basket. The client will be required to input the emails of the invitees. The invitees should be able to view and add new orders.

**RAT:** A common order basket that allows many clients to access the same order form across different accounts, but restricting each client's ability to only edit his/her own order items in the basket.

**DEP:** FR1, FR2 and FR7

### 3.2.1.9 Functional requirement 1.9

**ID:** FR9

**GIST:** Routing from the invitation email link

**DESC:** Upon clicking the invitation link, the client should be brought to the login/signup page. Upon login/signup the client should be routed to the menu page and the order basket should be automatically synced.

**RAT:** To allow a seamless flow for better client experience with invitation and adding items to the same order basket.

**DEP:** FR7

### 3.2.1.10 Functional requirement 1.10

**ID:** FR10

**GIST:** Adding order items to the order basket

**DESC:** After the order basket has been created, the client should be able to add the desired items to the order basket.

**RAT:** To allow the client to add order items only after the order basket has been created.

**DEP:** FR1, FR2 and FR7

### 3.2.1.11 Functional requirement 1.11

**ID:** FR11

**GIST:** Removing order items from the order basket

**DESC:** After the order basket has been created, the client should be able to remove the order item that they have added.

**RAT:** To allow the client to remove the order item in an event of a wrong selection.

**DEP:** FR1, FR2 and FR7

### 3.2.1.12 Functional requirement 1.12

**ID:** FR12

**GIST:** Placing the order

**DESC:** After all the clients have finished updating the order form with their desired food items, any of the client within the collaboration should be able to place the order.

**RAT:** To allow submitting of the order form.

**DEP:** FR1, FR2, FR7 and FR8

### 3.2.1.13 Functional requirement 1.13

**ID:** FR13

**GIST:** Cancelling the order

**DESC:** In an event where the client wants to cancel the order, the order basket will be deleted.

**RAT:** To allow deleting of the order form.

**DEP:** FR1, FR2, FR7 and FR8

### 3.2.1.14 Functional requirement 1.14

**ID:** FR14

**GIST:** Viewing confirmed order

**DESC:** The client should be allowed to view the list of order forms that have been confirmed. The item ordered by the client within each basket should be highlighted.

**RAT:** To allow the viewing of the reservations made and highlight the items that the client has ordered.

**DEP:** FR1, FR2 and FR12

### 3.2.1.15 Functional requirement 1.15

**ID:** FR15

**GIST:** Viewing order history

**DESC:** In the web application, the side navigation bar has a tab that directs clients to the order history page. The order history should be sorted accordingly to date, housing the details of the items that the client had ordered in the past.

**RAT:** To allow the client to keep track of the food items he/she had ordered previously.

**DEP:** FR1, FR2 and FR12

### 3.2.1.16 Functional requirement 1.16

**ID:** FR16

**GIST:** Organize and access multiple order basket collaboration

**DESC:** In the menu page, a client should be able to select the order basket according to the reservation date

**RAT:** As a client may be collaborating with multiple clients and hence own multiple order baskets. They should be allowed to select the particular basket by choosing the specific reservation from a dropdown menu. They should be allowed to add order items to the correct order basket upon specification of the reservation.

**DEP:** FR1, FR2 and FR7

## 3.2.2 User Class 2: Restaurant Owners

### 3.2.2.1 Functional requirement 2.1

**ID:** FR17

**Feature:** Registration

**DESC:** In order to access the system, they have to register for an account on the web application.

1. Scenario: Full information for registration  
When the restaurant uses the system for the first time, it is required to register for an account. Details such as username, password, address, email address and contact number will be required for the account registration. Thereafter, an authentication e-mail will be sent to the restaurant.
2. Scenario: Authentication of email  
After registration, the restaurant will receive an authentication email. Upon verification, the restaurant will be redirected to the site to log in.

### 3.2.2.2 Functional requirement 2.2

**ID:** FR18

**Feature:** Login

**DESC:** For subsequent access to the system, restaurant has to log in with their registered account.

1. Scenario: Successful login  
Upon filling up the login details, restaurant will be able to log in to the system.
2. Scenario: Password retrieval  
In cases where restaurant forgets or loses its password, there will be a password retrieval link to assist them. The restaurant is required to submit the same email used during registration. Upon submitting, the restaurant will receive an email with the new password and they will be prompted to change their password after logging in.
3. Scenario: Change password  
Should the restaurant require to change their password for any reason, they should be able to do so by verifying their old password before inputting the new one.
3. Scenario: Logging out  
The user should be able to log out once they are done with the application.

### 3.2.2.3 Functional requirement 2.3

**ID:** FR19

**Feature:** Manage restaurant's information

**DESC:** In order to update or make changes to its information, restaurant must log in to the system.

1. Scenario: Updating details  
When a restaurant wants to update details about themselves, they will be able to access the form directly and make changes. Changes will be reflected immediately.

### 3.2.2.4 Functional requirement 2.4

**ID:** FR20

**Feature:** Manage menu

**DESC:** Restaurant has the authority to manage its own menu

1. Scenario: Uploading menu item  
When restaurant wants to upload the food items on its menu into the system, they can use the food item form to do so. They will have to input these required fields: catID, name, price, description, image.
2. Scenario: Updating details of menu item  
When a restaurant wants to update or make changes to the details of a particular food item, they can make changes directly. Once done, the changes will be reflected immediately.
3. Scenario: Deleting menu item  
When the restaurant wants to remove a particular food item, they can delete it directly. The food item will be removed immediately.

### 3.2.2.5 Functional requirement 2.5

**ID:** FR21

**Feature:** Dashboard

**DESC:** Restaurant has access to the analytics dashboard to understand its sales and restaurant better.

1. Scenario: View breakdown of sales  
When the restaurant wants to find out the sales of food item, they can access the analytics dashboard which shows them a pie chart of food categories. It also provides a break down to the specific food items so that the restaurant can find out how a particular food item is doing.
2. Scenario: View analytics of reservation timings  
Restaurants can find out the popular reservation timings based on the number of reservations per hour shown on the analytics dashboard.

### 3.2.2.6 Functional requirement 2.6

**ID:** FR22

**Feature:** Overview of reservations

**DESC:** Restaurant is able to view a summary of all the confirmed reservations as well as the details of each reservations.

1. Scenario: View timings of reservations  
Restaurant can view all the incoming confirmed reservation timings and necessary details such as number of people.
2. Scenario: View details of reservation orders  
Restaurant can find out the food that are ordered for each reservations and prepare them in advance.

### 3.3 Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance.

#### 3.3.1 Order feature

**ID:** QR1

**GIST:** Prominent Order feature

**DESC:** The order feature should be easy to find for the client.

**RAT:** In order for the client to find the order feature easily

#### 3.3.2 Usage of order feature

**ID:** QR2

**GIST:** Usage of order feature

**DESC:** The order feature should be simple and intuitive for the client.

**RAT:** This is to allow clients to order on our site seamlessly.

#### 3.3.3 Usage of food items in list view

**ID:** QR3

**GIST:** Usage of the food items in the list view

**DESC:** The food items displayed in the list view should be user friendly and easy to understand. Selecting an item in the list of food should only take one click

**RAT:** In order for the client to use the listview easily

#### 3.3.4 Jio feature

**ID:** QR4

**GIST:** Prominent collaboration feature

**DESC:** The collaboration feature should be easy to find for the client

**RAT:** In order for the client to find the feature easily

#### 3.3.5 Usage of jio feature

**ID:** QR5

**GIST:** Usage of collaboration feature

**DESC:** The collaboration feature should be intuitive for the client

**RAT:** This allows clients to invite their friends to the same order easily.

## 3.4 Software System Attributes

The requirements in this section specify the required reliability, availability, security and maintainability of the software system

### 3.4.1 Reliability

**ID:** QR6

**GIST:** The reliability of the system.

**SCALE:** The reliability that the system stores the right orders for each user.

**METER:** Measurements obtained from 100 orders during testing.

**MUST:** More than 100% of the orders.

**PLAN:** More than 100% of the orders

**WISH:** 100% of the orders.

### 3.4.2 Availability

**ID:** QR7

**GIST:** The reliability of the system.

**SCALE:** The average system availability (Not considering network failing).

**METER:** Measurements obtained from 10 hours of usage during testing.

**MUST:** More than 98% of the time.

**PLAN:** More than 99% of the time.

**WISH:** 100% of the time.

**ID:** QR8

**GIST:** Internet Connection

**DESC:** The application should be connected to the Internet.

**RAT:** In order for the application to communicate with the database and render the necessary views.

### 3.4.3 Security

**ID:** QR9

**GIST:** Security of accounts.

**SCALE:** If a restaurant owner tries to log in to the web portal with a non-existing account then the restaurant owner should not be logged in. The restaurant owner should be notified about the failure to login.

**METER:** 100 attempts to log-in with a non-existing user account during testing.

**MUST:** 100% of the time.

**ID:** QR10

**GIST:** Security of accounts.

**SCALE:** If user tries to log in to the web portal with a non-existing account then the user should not be logged in. The user should be notified about the failure to login.

**METER:** 100 attempts to log-in with a non-existing user account during testing.

**MUST:** 100% of the time.

**ID:** QR11

**GIST:** Access rights of pages

**SCALE:** If a user tries to access a page that they do not have access to, the user should be directed to an error page and allowed to return to it's previous page.

**METER:** 50 attempts to access pages that user has no rights to

**MUST:** 100% of the time

### 3.4.4 Maintainability

**ID:** QR12

**GIST:** Application extensibility

**DESC:** The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

**RAT:** In order for future functions to be implemented easily to the application.

### 3.4.5 Portability

**ID:** QR13

**GIST:** Application portability

**DESC:** The application is a web application, extensible to across all mobile to computer devices.

**RAT:** In order for future functions to be implemented easily to the application.



## 4. Prioritization and Release plan

Customers (clients and restaurant owners) are never thrilled to find out they can't get all the features or the quality they want in release 1.0 of a new software web application. However, if the development team cannot deliver each and every single requirement by the scheduled initial delivery date, the project stakeholders must agree on which subset to implement first. Any project with resource limitations has to establish the relative priorities of the requested features, functional or quality requirements. Prioritization thus aids in resolving conflicts, plan for staged deliveries, and make the necessary trade-off decisions.

### 4.1 Feature prioritisation matrix

The feature prioritisation matrix (Figure 4.1) is formulated based on the measurement of the value, cost and risk on each requirement; based on equal weighting. The priority column is then calculated as follows:

$$\text{Priority} = \text{Value Percentage} \div (\text{Cost Percentage} + \text{Risk Percentage})$$

#### *Key Measurements*

**Relative benefit** - Each feature provides to the client or the restaurant owner on a scale from 1 to 9, with 1 indicating very little benefit and 9 being the maximum possible benefit. These benefits indicate the alignment with the web application requirements.

**Relative penalty** - The impact on the client or the restaurant owner if the feature is not included on a scale from 1 to 9, with 1 means essentially minimum penalty and 9 indicates a very serious downside.

**Total value** - The sum of the relative benefit and penalty.

**Relative cost** - The estimated cost of implementing each feature on a scale from 1 to 9, with 1 indicating very little cost and 9 being relatively more expensive. The cost rating is based on factors such as the requirement complexity, the extent of user interface work required, the potential ability to reuse existing designs or code, and the levels of testing and documentation needed.

**Relative risk** - The estimated the degree of technicality or other risk associated with each feature on a scale from 1 to 9. An estimate of 1 means you can program it with ease, while 9 indicates serious concerns about feasibility, the availability of developers with the needed expertise, or the use of unproven or unfamiliar tools and technologies.

Feature	Relative Benefit	Relative Penalty	Total Value	Value Percentage	Relative Cost	Cost Percentage	Relative Risk	Risk Percentage	Priority
QR13	9	6	15	3.45	1	1.33	1	1.27	1.33
FR 10	9	9	18	4.14	1	1.33	2	2.53	1.07
FR 1	1	9	10	2.30	1	1.33	1	1.27	0.88
FR 2	1	9	10	2.30	1	1.33	1	1.27	0.88
FR 17	1	9	10	2.30	1	1.33	1	1.27	0.88
FR 18	1	9	10	2.30	1	1.33	1	1.27	0.88
QR3	5	5	10	2.30	1	1.33	1	1.27	0.88
FR 12	9	9	18	4.14	2	2.67	2	2.53	0.80
FR 16	9	9	18	4.14	2	2.67	2	2.53	0.80
FR 7	6	6	12	2.76	1	1.33	2	2.53	0.71
FR 13	6	6	12	2.76	1	1.33	2	2.53	0.71
FR 3	1	7	8	1.84	1	1.33	1	1.27	0.71
FR 4	2	6	8	1.84	1	1.33	1	1.27	0.71
FR 21	9	7	16	3.68	2	2.67	2	2.53	0.71
FR 8	9	9	18	4.14	2	2.67	3	3.80	0.64
FR 20	9	9	18	4.14	2	2.67	3	3.80	0.64
FR 9	8	6	14	3.22	2	2.67	2	2.53	0.62
QR2	7	7	14	3.22	2	2.67	2	2.53	0.62
FR 22	6	4	10	2.30	1	1.33	2	2.53	0.59
QR1	5	5	10	2.30	1	1.33	2	2.53	0.59
QR4	8	8	16	3.68	2	2.67	3	3.80	0.57
QR5	8	8	16	3.68	2	2.67	3	3.80	0.57
FR 11	6	6	12	2.76	2	2.67	2	2.53	0.53
FR 14	4	2	6	1.38	1	1.33	1	1.27	0.53
FR 19	5	5	10	2.30	2	2.67	2	2.53	0.44
QR8	5	5	10	2.30	2	2.67	2	2.53	0.44
FR 6	2	2	4	0.92	1	1.33	1	1.27	0.35
FR 15	3	1	4	0.92	1	1.33	1	1.27	0.35
QR6	9	9	18	4.14	6	8.00	4	5.06	0.32
QR7	9	9	18	4.14	6	8.00	4	5.06	0.32
QR12	9	5	14	3.22	6	8.00	3	3.80	0.27
QR9	9	9	18	4.14	6	8.00	7	8.86	0.25
QR10	9	9	18	4.14	6	8.00	7	8.86	0.25
QR11	1	9	10	2.30	4	5.33	4	5.06	0.22
FR 5	1	1	2	0.46	1	1.33	1	1.27	0.18
Total	201	234	435	100	75	100	79	100	

Figure 4.1 Feature prioritization matrix

*Top 10 functional requirements*

1. Adding order items to the order basket
2. User registration
3. User login
4. Restaurant owner registration
5. Restaurant owner login
6. Placing the order
7. Organize and access multiple order basket collaboration
8. Create an order basket
9. Cancelling the order
10. User logout

*Top 5 quality requirements*

1. Application portability
2. Usage of the food items in the list view
3. Usage of order feature
4. Prominent Order feature
5. Usage of jio feature

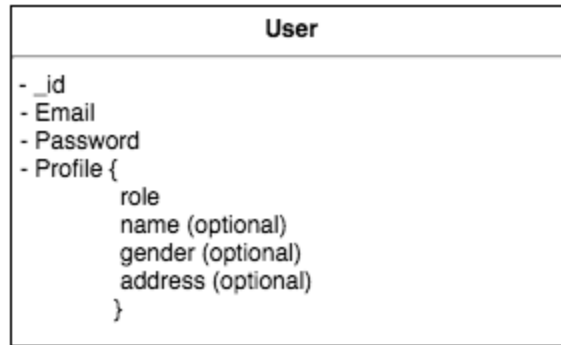
## 4.2 Release plan

The requirements were divided into two releases based on the prioritization and their dependencies. The two different releases were assembled so that each would work as a fully functional application.

In the first release the requirements that build up the foundation of the application would include the fundamentals such as login, registration to adding order items from the menu into the order basket.

The second release includes important requirements that are more exhaustive in the technical skills. However, these requirements are not vital for a functional application. They are more suited to act as additional feature, which is our theme of joining a group of friends that would be able to collaborate in adding order items to the same menu. This contributes to making the web application more attractive. Consequently, the completion of the web application is estimated to be within a period of 3 months.

## Appendix - Document Model Design



*User Collection*

### User class: Customer User

```

User = {
  _id: 2P96L32LShz57k4pP
  Email: Jim@gmail.com
  password: {
    "bcrypt": "$2a$10$AcpS2iZHa457WXQG38UNye4gpAlB6yb.ENZPuMluluYNTBufSHrzS"
  }
  Profile: {
    role : "User"
    name: Jim
    gender : Male
    address: 12 computing drive 600214
  }
}
  
```

### User class: Restaurant Owners

```

User = {
  _id: 3Q96B32AScz90k4qM
  Email: admin@openjio.com
  password: {
    "bcrypt": "$5a$10$AcpS2iZHa457WXQG38UNye4gpAlB6yb.SDNdsadaMSDnnasdsinK"
  }
  Profile: {
    role : "Admin"
    name: Openjio
    address: 10 computing drive 600210
  }
}
  
```

Orderitem
<ul style="list-style-type: none"> <li>- _id</li> <li>- foodID</li> <li>- category</li> <li>- orderID</li> <li>- quantity</li> <li>- custID</li> <li>- specialRequest (optional)</li> <li>- added</li> <li>- createdAt</li> </ul>

*OrderItem Collection*

```
OrderItem = {
  _id: 1Q00B32CSca12k0qQ
  foodID: 5P11A32WWaz12k4aP
  category: 1
  orderID: 0P97L12PAaz00k4aB
  quantity: 2
  custID: 2P96L32LShz57k4pP
  specialRequest: no chilli
  added: True/False
  createdAt: 2016-11-09 11:34:10
}
```

(ID of a food item chosen)

(Category number that the food item belongs to)

(Order id of the order that this item belongs to)

(Number of food item ordered)

(Customer ID)

(Special request (If any))

(Boolean field to check if order item is added to the order)

Order
- _id - reservationDate - reservationTime - numPax - totalPrice - confirmed - custID - createdAt

*Order Collection*

```
Order = {  
  _id: 0P97L12PAaz00k4aB  
  reservationDate: 2016-11-10  
  reservationTime: 12:00  
  numPax: 2  
  totalPrice: 21.50  
  custID: [2P96L32LShz57k4pP, 2P96L32LShz57k4pP]  
  createdAt: 2016-11-09 11:34:10  
}
```

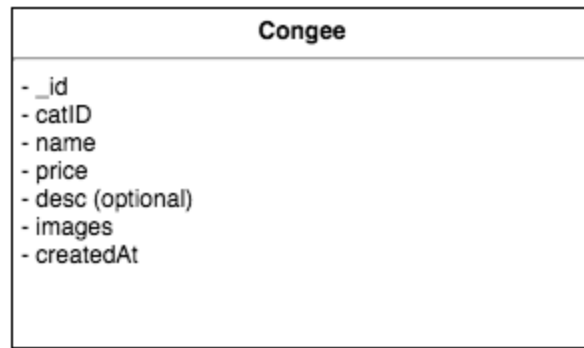
(Date of reservation)  
(Reservation timing)  
(Number of diners)  
(Total price)  
(Customer ID of all the dining customers)

FoodCategory
- _id - catNo - name

*Food Category Collection*

```
FoodCategory = {  
  _id: 2V97A12AAbz11k4qP  
  catNo: 1  
  name: Dim Sum  
}
```

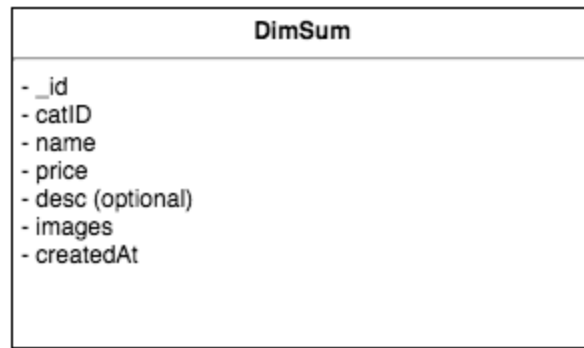
(Category number)  
(Name of category)

*Congee Collection*

```
Congee = {  
  _id: 2Q00A11CAaz22l0yP  
  catID: 2  
  name: Pork Congee  
  price: 8.50  
  desc: Sliced pork congee  
  images : image.jpg  
  createdAt: 2016-08-26 11:11:20  
}
```

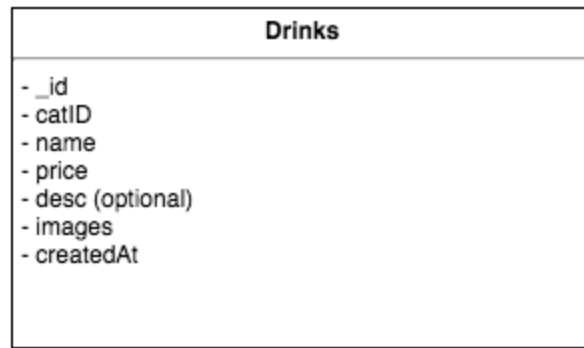
	(Category number)
	(Name of food)
	(Price of food)
	(Description of food)
	(Image of food)



*Dim Sum Collection*

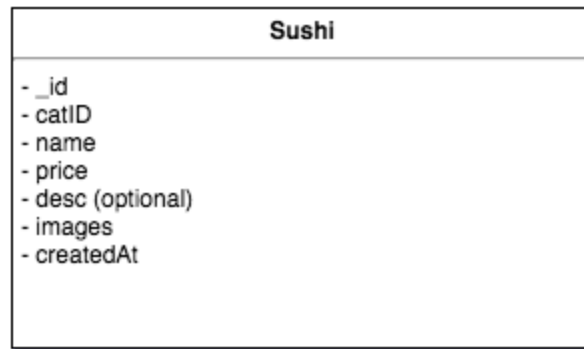
```
DimSum = {  
  _id: 5R17Q10BAwa22a4qF  
  catID: 1  
  name: Siew Mai  
  price: 3.50  
  desc: freshly steamed siew mai  
  images : image.jpg  
  createdAt: 2016-08-26 11:11:20  
}
```

(Category number)
(Name of food)
(Price of food)
(Description of food)
(Image of food)

*Drink Collection*

```
Drink = {  
  _id: 1E17A00NMbz51k4aY  
  catID: 4  
  name: Barley  
  price: 2.50  
  desc: Homemade barley  
  images : image.jpg  
  createdAt: 2016-08-26 11:11:20  
}
```

(Category number)  
(Name of food)  
(Price of food)  
(Description of food)  
(Image of food)

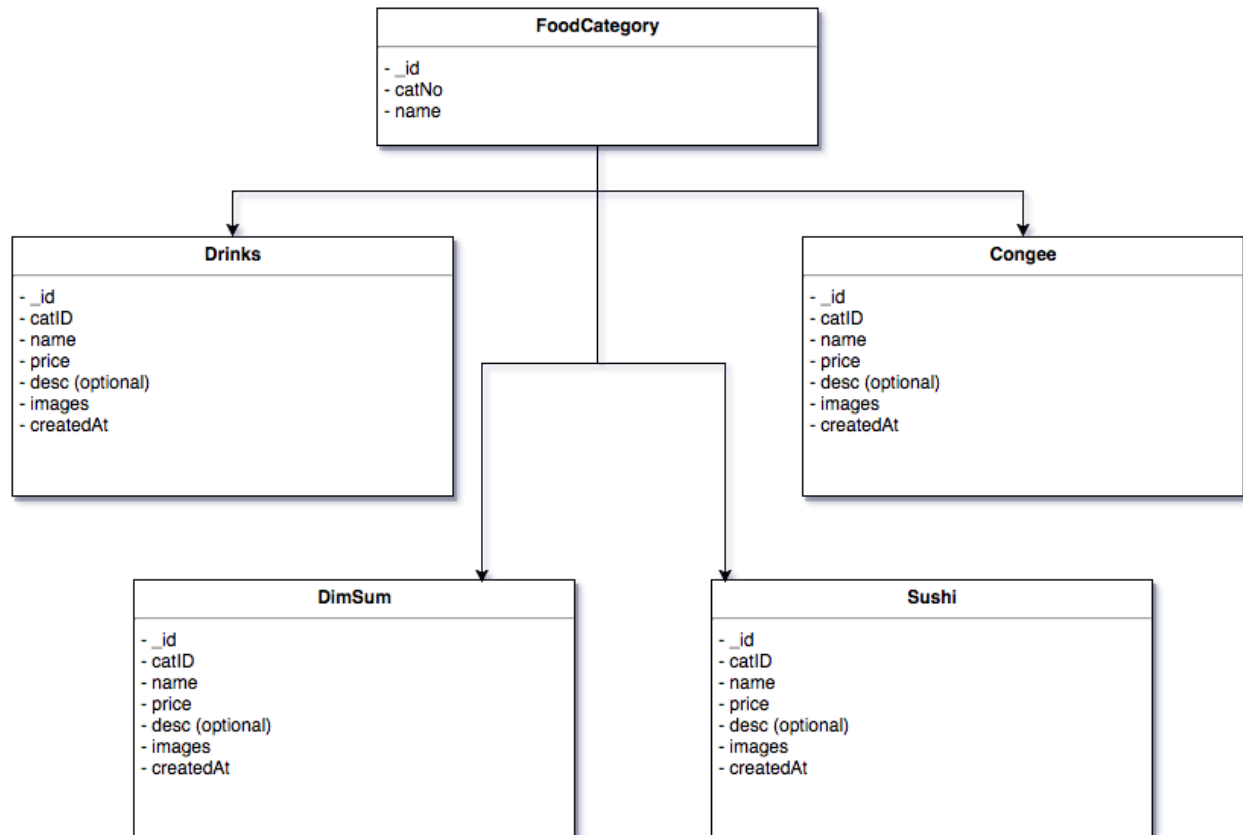
*Sushi Collection*

```
Sushi = {  
  _id: 5V00A12QQbz45k4eR  
  catID: 3  
  name: Salmon Roll  
  price: 9.50  
  desc: Fresh salmon air flown from Japan daily  
  images : image.jpg  
  createdAt: 2016-08-26 11:11:20  
}
```

(Category number)  
(Name of food)  
(Price of food)  
(Description of food)  
(Image of food)

## Hierarchical View

Each category of food is a child of the Food category



Each OrderItem belongs to an Order

