In-Dining QR Code Ordering System

Design and Planning Document

2019/10/11, version 1.0

2019/10/31, version 2.0 - added the testability of external interface in the integration testing, stated the automation testing explicitly

Table of Contents

1. System Architecture

1.1. Overview

1.2. Model-View-Controller (MVC) Architecture

1.3. Flux Architecture

1.4. Alternate Architecture Design

1.5. Design Risks

2. Design Details

2.1. Web Application for General Users

2.2. Mobile Application for Clients

2.3. Alternative Design

2.4. Design Risks

2.5. External Interfaces

3. Implementation Plan

3.1. Web Application for General Users

3.2. Mobile Application for Clients

3.3. Design Detail

4. Testing Plan

4.1. Unit Testing

4.2. Integration Testing

4.3. Regression Testing

4.4. End-to-End Testing

System Architecture

1.1 Overview

We will be using the Model-View-Controller architecture for our mobile client and Flux architecture for the web application. We chose MVC architecture because it provides us with a clear separation of the responsibility by dividing the applications into three main logical components - model, view and controller. And for the user web app, because the order information needs to be shared by more than one users and any pages at the same time, so the Flux can efficiently update the data to all components. This separation will aid fast and efficient development by enabling parallel focus on each logical component by different members in the team. Besides, since we are using a single database that will be accessed by both the web and the mobile app, the MVC architecture allows us to organize a single Model component for both the applications. The front-end will be developed in React (for the web application) and React Native (for the mobile application) and for the backend we will be using the Firebase SDK with a Realtime database.

1.2 Model-View-Controller (MVC) Architecture

1.2.1 Model

The model consists of a Realtime database which acts as the data store for our application. Realtime database is a cloud-based NoSQL database that stores data as JSON and enables synchronization across all client applications that have access to it. The database will act as a bridge between the web app and the mobile app, thus making

1.2.2 View

The View will be rendered using React components. For the web app, we will be using the standard React library to create components, Cascade Style Sheets (CSS) to style the components and the ReactDOM library to render the components in an HTML DOM tree. For the mobile app, we will be using React Native’s component library to create custom components and its Stylesheet class for styling.

1.2.3 Controller

The Controller's responsibility is to respond to user requests by fetching data from the model and returning it with the correct view. It essentially serves as a bridge between the two other components of MVC. For both our applications, the Javascript code will have several helper methods to handle and process requests.

1.3 Flux Architecture

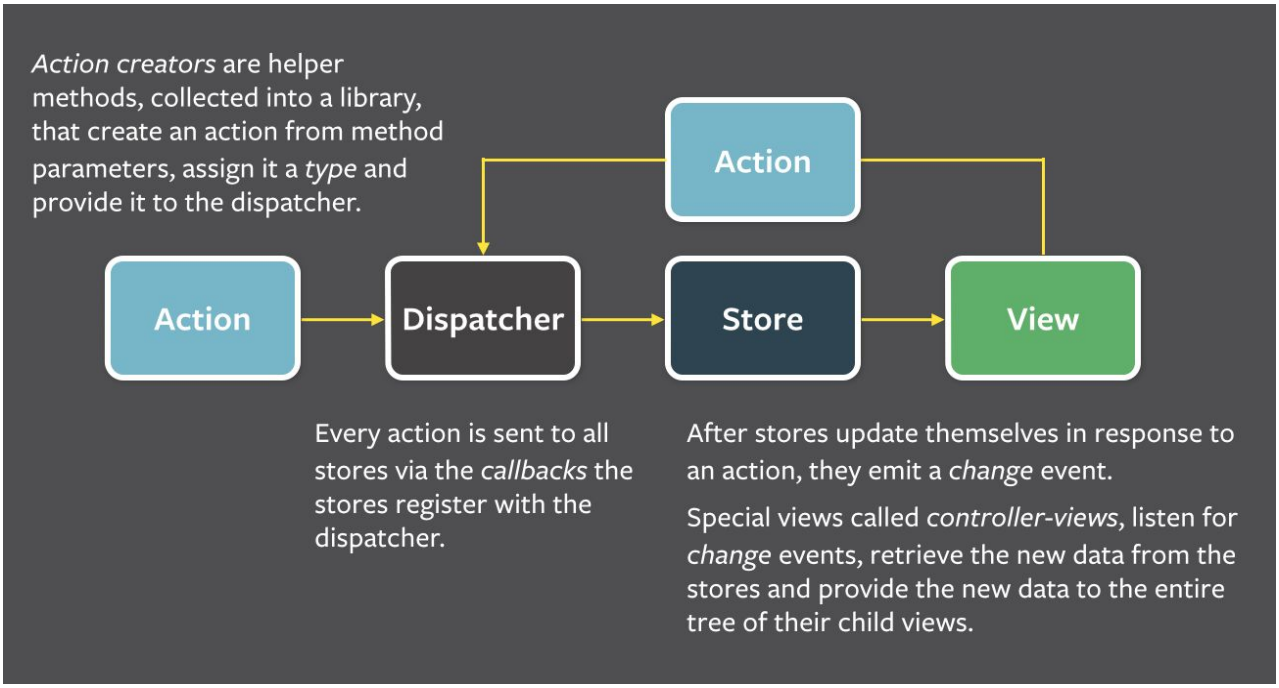

Fig 1. Flux Architecture

https://facebook.github.io/flux/docs/in-depth-overview

For the Web app, we might not fully use the MVC(Model View Controller) if there might be too much data interacts in the client side. Our alternative way would be using all of the design by the flux. However, we might collaborate this alternate design with the original plan as flux is the way to complement the bad point of MVC :

- Flux use the unidirectional data flow, so it could solve the relation of changeable scope in the client part
- Similar pattern as using the MVC, so it could be easier to implement

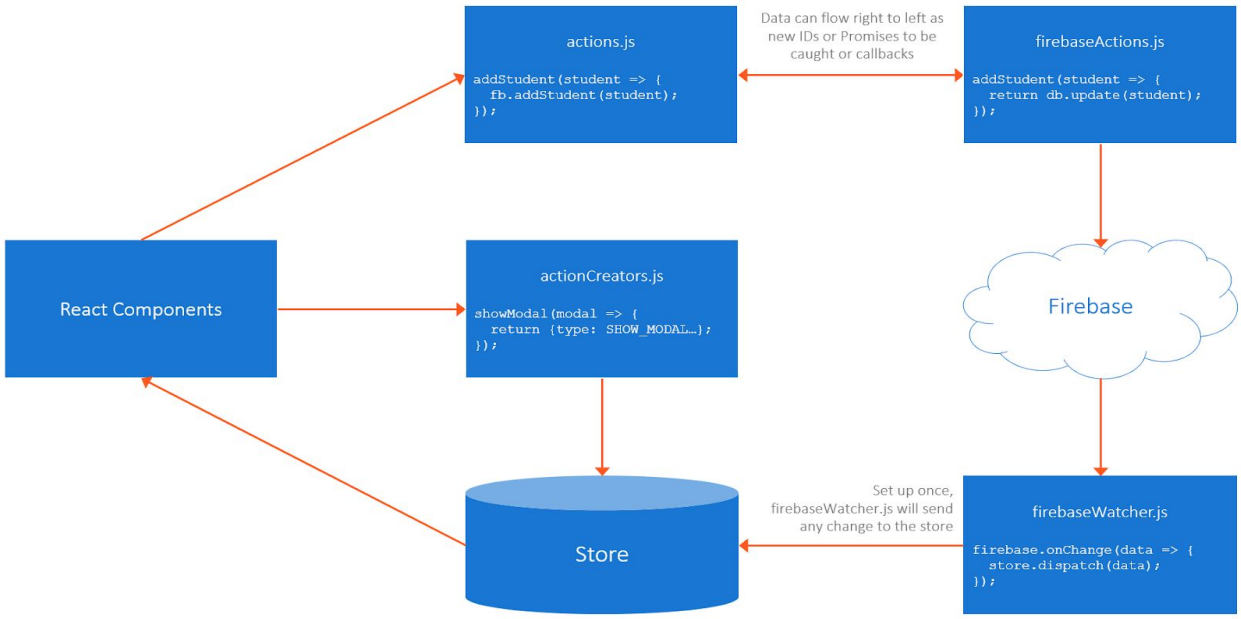

Fig 2. Flux Architecture

The image above shows how data flows between different components of our applications (source:https://medium.com/@david.gilbertson/react-and-firebase-sittin-in-a-tree-a00d481786cb)

1.4 Alternate Architectural Design

Since our web application will use flux (Redux) and need to put all menu information into store and refresh and distribute to all pages while refreshed. At the same time, our database is quite big because of the large amount of images. Therefore, if the flux slows down our system and greatly affect the user experience, maybe we will need to change to traditional MVC and only update the view relevant to the model refreshed.

1.5 Design Risks

One of the design risks that we could face is, as described above, the data storage of our app would be significantly decided by the menu of each of the restaurant and the menu could be added or deleted by the owner every time. So this could make some issue for some clients to take longer time for loading all of the images and menu details. And because the scope would vary every time, this could be much harder to debug for the client side if we only use the MVC architecture.

Design Details

2.1 Web Application for General Users (Restaurant Customers)

>Name: PromptDishComponent(Jiujiu)

Content	Component	Description
CategorySidebar	Button	Onclick, overlay a sidebar display all the categories of the menu
SearchBar	TextInput	Allow users to type in some keywords for searching for specific dishes
Cart	Button	Onclick, navigate to the cart page which shows all the dishes added to the cart by the users in that table
OrderInfo	Button	Onclick, navigate to the order page which shows all the orders that that table already made
Assistant	Button	On click, send an alert to the mobile client of of restaurant.

Prompt Sentences	Text	Sentences set by the restaurant to prompt some dishes special for certain time
DishImage	Image	Pictures set by the restaurant to prompt some dishes special for certain time

>Name: MenuComponent(Jiujiu)

Content	Component	Description
CategorySidebar	Button	Onclick, overlay a sidebar display all the categories of the menu
SearchBar	TextInput	Allow users to type in some keywords for searching for specific dishes
Cart	Button	Onclick, navigate to the cart page which shows all the dishes added to the cart by the users in that table
OrderInfo	Button	Onclick, navigate to the order page which shows all the orders that that table already made
Assistant	Button	On click, send an alert to the mobile client of of restaurant.
DishItem	Card	Displaying the basic information about a dish and can quickly add to the cart. On click, pop up a **modal** to display the detailed information. DishComponent(Jiujiu)

Content	Component	Description

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Dish image</td><td>Image</td><td>Big and clear picture of the dish user selected</td></tr>
<tr><td>Name</td><td>Text</td><td>The name of the dish</td></tr>
<tr><td>Price</td><td>Text</td><td>The price of the dish</td></tr>
<tr><td>Description</td><td>Text</td><td>The description of the dish</td></tr>
<tr><td>Available choice</td><td>Selector</td><td>Some choice offered by restaurant that able to choose for that dish(cold/hot, extra serving etc.)</td></tr>
<tr><td>Amount</td><td>TextInput</td><td>The amount user want for this dish</td></tr>
<tr><td>Add Button</td><td>Button</td><td>Add the amount of that dish to the cart</td></tr>
</table>
</td></tr>
<tr><td>renderDishItem</td><td>Function</td><td>Helper method to retrieve the dish information from database and use for render</td></tr>
</table>

>Name: ShoppingCartComponent(JiMin)

Content	Component	Description

<table>
<tr><td>CategorySidebar</td><td>Button</td><td>Onclick, overlay a sidebar display all the categories of the menu</td></tr>
<tr><td>SearchBar</td><td>TextInput</td><td>Allow users to type in some keywords for searching for specific dishes</td></tr>
<tr><td>Cart</td><td>Button</td><td>Onclick, navigate to the cart page which shows all the dishes added to the cart by the users in that table</td></tr>
<tr><td>OrderInfo</td><td>Button</td><td>Onclick, navigate to the order page which shows all the orders that that table already made</td></tr>
<tr><td>Assistant</td><td>Button</td><td>On click, send an alert to the mobile client of of restaurant.</td></tr>
<tr><td>DishItem</td><td>Card</td><td>Displaying the basic information about a dish and can quickly add to the cart. On click, pop up a modal to display the detailed information.

DishComponent(Jiujiu)<table><tr><th>Content</th><th>Component</th><th>Description</th></tr><tr><td>Dish image</td><td>Image</td><td>Big and clear picture of the dish user selected</td></tr><tr><td>Name</td><td>Text</td><td>The name of the dish</td></tr><tr><td>Price</td><td>Text</td><td>The price of the dish</td></tr><tr><td>Description</td><td>Text</td><td>The description of the dish</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Available choice</td><td>Selector</td><td>Some choice offered by restaurant that able to choose for that dish(cold/hot, extra serving etc.)</td></tr>
<tr><td>Amount</td><td>TextInput</td><td>The amount user want for this dish</td></tr>
<tr><td>Add Button</td><td>Button</td><td>Add the amount of that dish to the cart</td></tr>
</table>
</td></tr>
<tr><td>TaxPrice</td><td>Text</td><td>The tax price for the total price of dish</td></tr>
<tr><td>TotalPrice</td><td>Text</td><td>The total price calculated automatically depending on the number of dishes that customers erase or add</td></tr>
<tr><td>Clear</td><td>Button</td><td>On click, erase all of the order list(=Dish Component) that put it in the shopping cart</td></tr>
<tr><td>Order</td><td>Button</td><td>On click, pop up the confirmation alert modal on the display to proceed the order

OrderProceed(JiMin)
<table>
<tr><th>Content</th><th>Component</th><th>Description</th></tr>
<tr><td>Confirmation</td><td>Text</td><td>Show the text “Do you want to proceed?” on the modal</td></tr>
</table>
</td></tr>
</table>

		<table><tr><td>Yes</td><td>Button</td><td>When user clicks it, the order list data will be stored in the order history page</td></tr><tr><td>No</td><td>Button</td><td>When user clicks it, the modal will disappear and the user can keep going on for ordering navigation</td></tr></table>

>Name: OrderHistoryComponent(Han, Tainy)

Content	Component	Description
CategorySidebar	Button	Onclick, overlay a sidebar display all the categories of the menu
SearchBar	TextInput	Allow users to type in some keywords for searching for specific dishes
Cart	Button	Onclick, navigate to the cart page which shows all the dishes added to the cart by the users in that table
OrderInfo	Button	Onclick, navigate to the order page which shows all the orders that that table already made
Assistant	Button	On click, send an alert to the mobile client of of restaurant.
OrderHistory	Container	If no order history, shows NoHistory else shows the orderList..

<table>
<tr><td>NoHistory</td><td>Text</td><td>Prompt the user there is no history found</td></tr>
<tr><td>OrderList</td><td>Table</td><td>Displaying the details of history orders

OrderinfoComponent

<table>
<tr><th>Content</th><th>Component</th><th>Description</th></tr>
<tr><td>Time</td><td>Text</td><td>Time of the order created</td></tr>
<tr><td>OrderNumber</td><td>Text</td><td>The number of this order</td></tr>
<tr><td>Name</td><td>Text</td><td>The name of the dish</td></tr>
<tr><td>Price</td><td>Text</td><td>The price of the dish</td></tr>
<tr><td>Number</td><td>Text</td><td>The number of the dish in this order</td></tr>
<tr><td>OverallPrice</td><td>Text</td><td>The tax and overall price of this order</td></tr>
</table>
</td></tr>
</table>

2.2 Mobile Application for Clients (Restaurant Owner)

>Name: LoginComponent(Han)

Content	Component	Description

Logo	Image	The Logo of our Application
UserName	TextInput	Allow users to type in username
PassWord	TextInput	Allow users to type in the password
Login	Button	On click navigate to homepage
Remember	SelectBox	On choice the app will remember the identity of user and login automatically at the next time
SignUp	Button	On click navigate to register page

>Name:RegisterComponent(Han, Lan)

Content	Component	Description
UserPhoto	image	The user image.
UserName	TextInput	Allow users to type in username
Email	TextInput	Allow users to type in email
Password	TextInput	Allow users to type in a password
PhoneNumber	TextInput	Allow users to type in phone number
Save	Button	On click send the register information to back end and navigate to homepage

Cancel	Button	On click navigate to login page

>Name: OrderPage(Anubhav)

Content	Component	Description
Today	Button	On click display all orders from today
OrderFilters	View	Buttons used to filter orders Content | Component | Description Yesterday | Button | On click display all orders from yesterday All | Button | On click display orders of all status New | Button | On click display orders in progress Served | Button | On click display orders that have been served but not yet paid Cancelled | Button | On click display cancelled orders

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>OrderList</td><td>FlatList</td><td>Displays lists of all orders according to applied filters</td></tr>
<tr><td>OrderDetails</td><td>View</td><td>Details for each order

<table><tr><th>Content</th><th>Component</th><th>Description</th></tr><tr><td>Time</td><td>Text</td><td>Time of the order created</td></tr><tr><td>OrderNumber</td><td>Text</td><td>The number of this order</td></tr><tr><td>Name</td><td>Text</td><td>The name of the dish</td></tr><tr><td>Price</td><td>Text</td><td>The price of the dish</td></tr><tr><td>Number</td><td>Text</td><td>The number of the dish in this order</td></tr><tr><td>OverallPrice</td><td>Text</td><td>The tax and overall price of this order</td></tr></table></td></tr>
<tr><td>TotalSale</td><td>Text</td><td>Display total sales for that day</td></tr>
<tr><td>Table</td><td>Button</td><td>On click navigate to ‘Table’ (current) page</td></tr>
</table>

Orders	Button	On click navigate to “Orders” page
Profile	Button	On click navigate to “Profile” page

>Name: ProfileComponent(Lan)

Content	Component	Description
UserPhoto	Image	Shows the user image
UserName	TextInput	Show user’s username
Phone Number	TextInput	Show user’s phone number
Email	TextInput	Show user’s email
Today’s Sale	TextInput	Show today’s sale
Today’s orders	TextInput	Show today’s number of orders
Menu	Button	On click navigate to menu edit page
Edit Table	Button	On click navigate to table edit page
Help	Button	Displaying emailSendingComponent emailSendingComponent Content | Component | Description

Email	TextInput	Allow users to fill in their contact email
Username	TextInput	Allow users to fill in their username
Content	TextInput	Allow users to fill in their problem
Send	Button	On click send the email to the technique support team
Cancel	Button	On click dismiss the request

>Name: MenuEditComponent(Lan)

Content	Component	Description
Category	Scrollable tab view	Display all the categories, each is a button
Edit category	Button	On click allow user to edit the categoryEditComponent categoryEditComponent

Content	Component	Description
category	TextInput	Display existing

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>categories</td></tr>
<tr><td>Add</td><td>Button</td><td>On click add a new TextInput to the end</td></tr>
<tr><td>Confirm</td><td>Button</td><td>On click send new categories to the database, and navigate to the menu page</td></tr>
<tr><td>Cancel</td><td>Button</td><td>On click dismiss the request</td></tr>
</table>
</td></tr>
<tr><td>Dish</td><td>Card</td><td>Display information about the dish

dishCardComponent

<table>
<tr><th>Content</th><th>Component</th><th>Description</th></tr>
<tr><td>Name</td><td>TextInput</td><td>Display name of the dish</td></tr>
<tr><td>picture</td><td>Image</td><td>Display the image of the dish</td></tr>
<tr><td>Price</td><td>TextInput</td><td>Display price of the dish</td></tr>
<tr><td>Availability</td><td>Picker</td><td>Pick the availability of the dish</td></tr>
<tr><td>Edit</td><td>Button</td><td>On click navigate</td></tr>
</table>
</td></tr>
</table>

		<table><tr><td></td><td></td><td>to DishEditComponent</td></tr><tr><td>Cancel</td><td>Button</td><td>On click cancel the dish</td></tr></table>
Reset	Button	On click display a model asking the user if he really wants to reset all the menu
Add	Button	On click navigate to DishEditComponent

>Name: DishEditComponent(Han, Lan)

Content	Component	Description
Save	Button	On click add/update the new dish information to menu
Cancel	Button	On click navigate to Menu Edit page
Image	Button	On click navigate to upload image for dish
Name	TextInput	Allow users to input the name of dish
Category	SelectBox	On choice add selected category label to dish
Ingredients	SelectBox	On choice add selected ingredient label to dish

Price	TextInput	Allow users to input Price of dish
Description	TextInput	Allow users to input Description of dish
discount	TextInput	Allow users to input discount of dish

>Name: TableMapComponent(Anubhav, Shunyu)

Content	Component	Description
DateSpecificData	View	Displays specific information about the current date DateSpecificData Content | Component | Description Today's Date and Time | Text | Current time and date New Orders | Text | The number of new (active) orders that have not been served yet
Table	TouchableHighlight	Color coded buttons representing the order status of each table. On click opens a "TableOrderHistory" component
TableOrderHistor	Drawer (from react-navigation),	Display order history for the current customer on the table.

<table>
<tr>
<td>y</td>
<td>ScrollView</td>
<td>
<table>
<tr><th>Content</th><th>Component</th><th>Description</th></tr>
<tr><td>Table Number</td><td>Text</td><td>Table number of the selected table</td></tr>
<tr><td>Order Number</td><td>Text</td><td>Current Order Number</td></tr>
<tr><td>Time</td><td>Text</td><td>Time when order was placed</td></tr>
<tr><td>Order details</td><td>Text</td><td>Name of the dish</td></tr>
<tr><td>Price</td><td>Text</td><td>Cost of dish</td></tr>
<tr><td>Cancel</td><td>Button</td><td>On click cancel orders (that have not been served) and change color of table icon</td></tr>
<tr><td>Confirmation</td><td>Alert</td><td>Double checks the user’s decision. Users can either select ‘cancel’ or ‘OK’.</td></tr>
<tr><td>Paid</td><td>Button</td><td>On click confirm payment by customer and change color of table icon</td></tr>
<tr><td>Assistance</td><td>Button</td><td>Renders only when customer</td></tr>
</table>
</td>
</tr>
</table>

		<table><tr><td>Complete</td><td></td><td>sends assistant call. On click confirm that the customer has been assisted with their request</td></tr></table>
Setting	Button	On click navigate to setting page
Table	Button	On click navigate to ‘Table’ (current) page
Orders	Button	On click navigate to “Orders” page
Profile	Button	On click navigate to “Profile” page

2.3 Design risks and Limitations

One risk our program will take is when there are more than one table is requesting assistant. As a result, the later request’s window will cover other request immediately.

Another potential problem is that if the restaurant owner change the menu during the ordering of a customer, the deleted dish still be ordered.

2.4 Alternative Design

For the duplicate assistant call, we can change some notification policy that only push notification after some time period in order to not the first come assistant call be overwritten by the later call.

For the changed menu risk, we may set a regulation that the restaurant owner can only change the menu after one day. Instead, they can change the availability of each dish

anytime. When the customer order a dish no longer provided, there will be a warning message pops up to remind customers change their order.

2.5 External Interfaces

2.5.1 Firebase Server

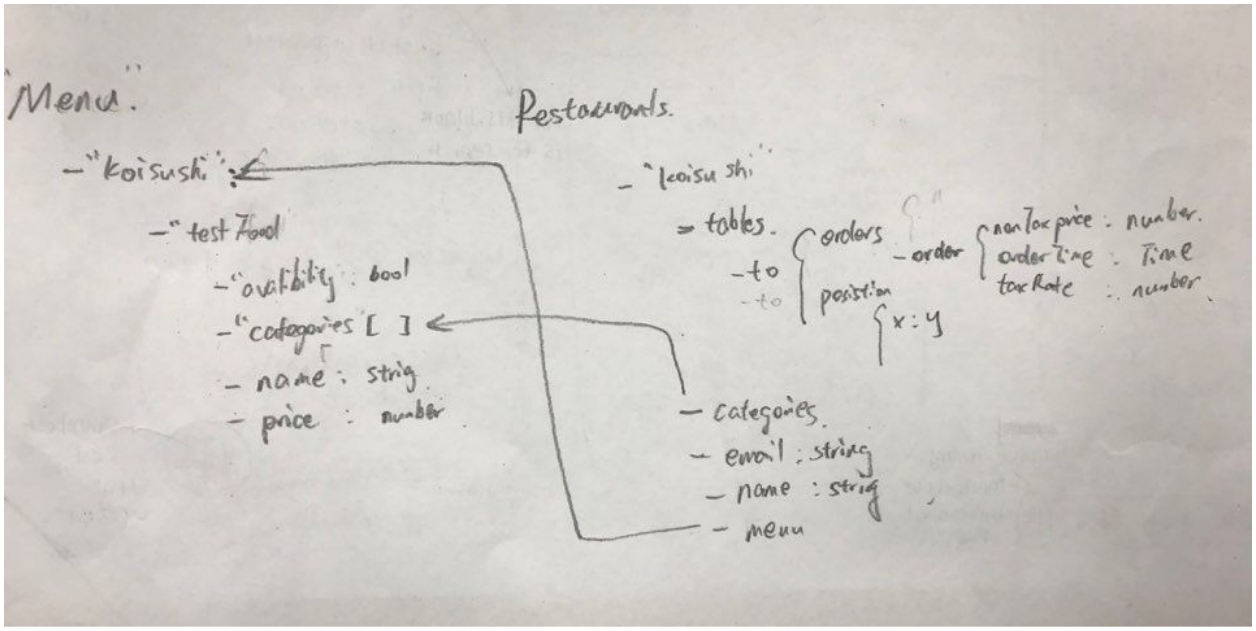

Fig 3. Database Diagram

Our application will rely on the Firebase server for store and push data to the client. Here is the list.

https://console.firebase.google.com/u/0/project/qr-code-ordering-system/database/firestore/data~2Fmenu~2Fkoisushi

https://console.firebase.google.com/u/0/project/qr-code-ordering-system/database/firestore/data~2Frestaurants~2Fkoisushi~2Ftables~2Ft0~2Forders

https://console.firebase.google.com/u/0/project/qr-code-ordering-system/database/firestore/data~2Frestaurants~2Fkoisushi

https://console.firebase.google.com/u/0/project/qr-code-ordering-system/database/firestore/data~2Fmenu~2Fkoisushi~2Ffoods~2FuDAEkXSbupddtw8vUQQk

https://console.firebase.google.com/u/0/project/qr-code-ordering-system/storage/qr-code-ordering-system.appspot.com/files

2.5.2 Apple Push Notification Service (APNs)

APNs is the channel through which push notifications, originating at our server, can be sent to our client devices. Each client device establishes an encrypted IP connection with APNs to receive notifications. We will monitor when new data is ready on our server for a device and send a notification to the APNs. The APNs will in turn push the notification to the device, even if the app is not running. The payload for APNs is 256 bytes per notification. This payload must include the data to present to the user, as well as how to present the data in a notification. Any notifications larger than this are rejected by APNs, so we must be careful with the notifications our server will generate.

2.5.3 Google Cloud Messaging Service (GCM)

The server can push notifications to mobile clients using the HTTP Spost request. The notification key is used to push notifications to a set of registration ids. As the server pushes notifications to the mobile clients, the response to the push request from the Google cloud service includes the list of failed registration ids.The server may use this data refresh the registration ids in its database.

2.5.4 Stripe and Bolt

In order to secure the user's information while paying the bill, we might need to use third party online payment Stripe and Bolt. They already have the security program called PCIDSS(which is the Payment Card Industry Data Security Standard) which we need to compliance. The specific instructions for installing is specified here:
https://www.paywithbolt.com/?setup

Implementation Plan

3.1 Web Application for general users(Restaurant Customers)

1. Prompt Dish Page -- Assigned to Jimin

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Display promoted dish a. Picture b. Description	2	None	1	1	4
2. View dish categories	1	None	2	1	4
3. Search for a specific dish	3	None	4	2	2
4. Go to Cart	1	3	2	1	4
5. Go to Order History	1	4	2	1	4
6. Call Assistance	2	/	2	2	4

2. Menu Page -- Assigned to Jiujiu

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Navigate to other	1	None	2	1	4

categories					
2. Search for a specific dish	3	None	4	2	2
3. Add dish in the overall dish menu page	1	2	3	1	5
4. Go to Cart	1	3	2	1	4
5. Go to Order History	1	4	2	1	4
6. Call Assistance	2	/	2	2	4

3. Cart Page -- Assigned to Han

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Change the number of a specific dish	2	/	2	1	5
2. Delete a dish	2	/	2	1	5
3. Clear the Cart	2	/	2	1	4
4. Confirm the order a. Yes b. No	1	/	1	1	5
5. Go to Menu	1	2	2	1	4
6. Go to Order History	1	4	2	1	4
7. Call Assistance	2	/	2	2	4

4. Order History Page -- Assigned to Tainy

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Display current order information a. Name b. Quantity	2	/	2	1	5

c. Price d. time					
2. Pay for the order	1	/	1	2	2
3. Go to Menu	1	2	2	1	4
4. Go to Cart	1	3	2	1	4
5. Call Assistance	2	/	2	2	4

3.2 Mobile Application for Clients(Restaurant owner)

1. Login Page -- Assigned to Shunyu

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Sign Up	1	none	1	1	5
2. Login	1	none	1	1	5
3. Logout	1	1.1	1	1	5
4. Forget password	3	none	2	2	3

2. Table Page -- Assigned to Shunyu

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Display table Status a. Availability b. Order detail(2)	4	2.3	3	1	4
2. Display current time	1	none	1	2	3
3. Display new orders	3	2.1.b	2	1/2	4
4. Edit table status	3	2.1	3	2	5

3. Order Page -- Assigned to Anubhav

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Display the following order details: a. Table number b. Ordered time c. Dishes d. Tips e. Notes f. Total sale	3	2.1.b	2	2	5
2. Search order a. By date b. By Status	3	2.1.b	2	2/3	2

4. Profile Page -- Assigned to Lan

Use Cases	Difficulty	Dependency	Effort	Iteration	Priority
1. Display user info a. Username b. Email c. Phone number d. image	1	1.2	2	1	5
2. Display statistics a. Total sale b. Total orders	3	3.1	3	1/2	4
3. Edit menu a. Add dish b. Delete dish c. Edit categories d. Edit dish e. Cancel dish f. Reset dish(deleted)	4	/	5	1	5
4. Edit table a. Add tables	4	3.1	5	1	5

b. Remove tables c. Table layout					
5. App assistance	2	/	2	2/3	1

3.3 Design Detail

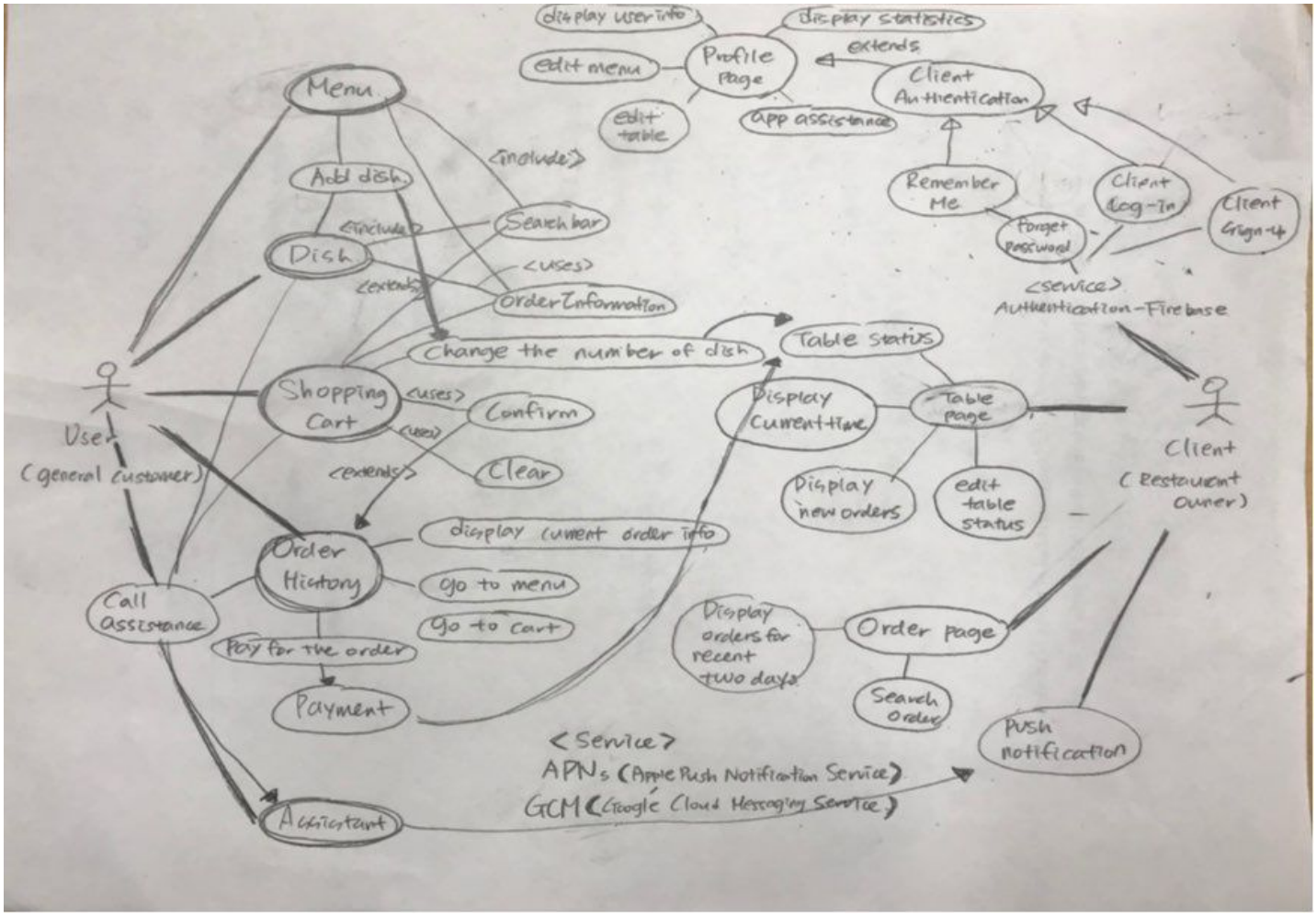

Fig 4. UML diagram for the implementation

Testing Plan

We will be relying heavily on unit and regression testing.

4.1 Unit Testing

Scheduled Time: At the end of each iteration

The goal of unit testing is to make sure the smallest piece works in its context in isolation.

4.1.1 Tools

- **Jest and Enzyme:** We will use Enzyme to test our React Components' output. Jest is popular among Facebook developers to write **automated tests** for all Javascript code such as the React application. Enzyme is previously used for Airbnb, but recently popularized for **automation testing** the React Components' output.

  https://www.testingexcellence.com/reactjs-test-aut%C3%A5omation-tools/

- **Snapshot Testing:** Snapshot tests are used to make sure if our UI is rendered correctly given certain props and states and does not change unexpectedly after editing the code
- **React-native-testing-library:** As Enzyme only supports shallow rendering, this supported library could help test for deep rendering of the each of the component

4.1.2 Visual Testing

This part should be tested in every stage in the development. This test include the display of dish, order and daily dish. We will check if it has ideal visual performance after finishing one component. For those components which have multiple user cases, we will have a detailed test in the visual test. We are using snapshot test for visual testing part.

4.1.3 Program Logic Testing

Logic is the part which decides the running of whole program based on the user cases. It is not obvious in the user end but it is essential to organize the flow and processing of data. We should test this part as they are used in whole program by passing virtual values to a specific function and check if it get the value expected.

4.2 Integration Testing

Scheduled Time: 1 week before the end of the iteration2

The integration testing is used to ensure the proper functioning of various components when they are integrated together. As we have two big components to implement, customer and clients(which is restaurant owners in our app), first, we are going to separate tests for each of the side's intended function to work well.

4.2.1 Web Application for General User (Restaurant Customer)

After the Unit tests of the web application, we will test the web page as a whole to make sure every component and navigate and pass the data correctly the next page. To implement that, we will check:

- integration of the actions and dispatcher of each page.
- integration of the views and stores of each page.
- integration of the stores and server data.

4.2.2 Mobile Application for Clients (Restaurant Owner)

After unit tests for different components of the mobile application, we will test how these different components interact with each other to provide desired features. We will haves tests such as:

- Restaurant authentication
- Adding, deleting and editing menu items
- Adding and removing tables.

4.2.3 Interaction Between User and Client

After integration tests for both web and mobile application, we will run the different tests to ensure two applications can work together. We will have tests such as:

- Adding menu from the client-end and check if the menu correctly displays on the user-end
- Confirming an order from the user-end and check if the client-end can receive the order information that matches the order history on the user-end
- Calling assistant from the user-end and check if the client-end can see the assistant notification

- Integration of the Google Cloud Service with the server for Apple Pushing Notification to test whether the receival of notification works well

4.3 Security Testing

Scheduled Time: 1 week before the end of the iteration2

In order to test for the user account and payment security, we are going to use the OWASP(Open Web Application Security Project), which is a great resource for software security professionals.

http://softwaretestingfundamentals.com/security-testing/

4.4 Regression Testing

Scheduled TIme: 2 days after each iteration

In order to prevent the new bugs brought by new feature added in recent iteration. Each time we finished a round of iteration, we will keep all the test code and try to pass them to check our program still work as expected. By doing the regression testing, we can make sure there are no negative effect between the component implemented in different iteration and it makes us easier to debug based on a part of reliable code.

4.5 End-to-End Testing

Scheduled Time: After the completion of each feature with cross-application dependencies and after each iteration.

End-to-end testing is used to test the flow of data in applications from start to end. Since our system has two independent systems that share a common data store, it is vital that we constantly test features that depend on the retrieval of data stored through the other application. Examples including displaying menu stored by the client on the web app and displaying the revenue from an order from the web app on the client app. In order to test these features, sample data (like the menu) will be pushed to the database through the app responsible for updating this data, and the app that wishes to use this data will pull it from the database.

End-to-end testing will enable us to identify dependencies in the flow of data between the two applications and will also help us gauge data integrity for our system.