ORDER MANAGEMENT SYSTEM FOR RESTAURANT (OMSR)

TAN CHEE KIN

Bachelor of Computer Science
(Software Engineering) with Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : TAN CHEE KIN

Date of Birth :

Title : ORDER MANAGEMENT SYSTEM FOR RESTAURANT

Academic Session : 2022/2023

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_____	_____
Tan Chee Kin	Ts. Dr. Nabilah Filzah binti Mohd Radzuan
Date:5th February 2023	Date: 10/02/2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Doctor of Philosophy/ Master of Engineering/ Master of Science in ……..

(Supervisor's Signature)

Full Name : Ts. Dr. Nabilah Filzah binti Mohd Radzuan

Position : Lecturer

Date : 10/02/2023

(Co-supervisor's Signature)

Full Name :

Position :

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : TAN CHEE KIN

ID Number : CB19021

Date : 5th February 2023

ORDER MANAGEMENT SYSTEM
FOR
RESTAURANT
(OMSR)

TAN CHEE KIN

A thesis submitted in fulfilment of the requirements
for the award of the degree of
Bachelor of Computer Science (Software Engineering) with Honours

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2023

ACKNOWLEDGEMENTS

There was a huge amount of help and guidance offered to me during the process of writing this thesis.

Firstly, I like to express my sincere thanks to my supervisor, Ts. Dr. Nabilah Filzah binti Mohd Radzuan, whose unique insight and expert opinions have driven my work to new heights.

Moreover, I would like to express my gratitude to my family for their unwavering support in terms of finances and mental throughout my academic studies. Finally, I would like to show my thankfulness to my three great friends, Lee Yong Jian, Ronald Lim Sheng Wei and Tan Yi Wee who gave constructive criticism on my theses.

ABSTRAK

Makanan ialah keperluan harian manusia. Kita memerlukan makanan untuk mendapatkan tenaga untuk menjalani aktiviti-aktiviti harian. Walau bagaimanapun, proses pemesanan di restoran adalah proses yang mengambil masa yang panjang. Lebih-lebih lagi, kebanyakan restoran hari ini masih menggunakan kaedah pesanan berasaskan kertas yang mempunyai kadar ralat yang tinggi. Hal ini demikian kerana kakitangan restoran memerlukan ingatan yang baik untuk mengingati pesanan-pesanan yang dibuat. Tambahan pula, pesanan yang dibuat juga akan tercicir dalam proses penghantaran pesanan. Akibatnya, kadar kesilapan pesanan makanan yang tinggi akan meninggalkan pengalaman yang tidak memuaskan untuk pelanggan-pelanggan restoran. Oleh itu, Sistem Pengurusan Pesanan untuk Restoran (OMSR) dicadangkan untuk menangani masalah ini. Fokus utama projek ini adalah untuk membangunkan satu aplikasi pesanan yang mempunyai fungsi pembatalan pesanan untuk restoran. Tiga aplikasi pesanan yang sedia ada dalam pasaran telah dikaji untuk menghasilkan sistem pesanan yang lebih baik dan mempunyai fungsi pembatalan pesanan. Hasil projek ini adalah sebuah sistem pengurusan pesanan yang menyediakan pelbagai fungsi kepada pengguna. Sebagai contoh, pelanggan boleh memesan menu, melihat status tentang pesanan yang dibuat, melihat sejarah pesanan, memberikan maklum balas dan membatalkan pesanan mereka jika syarat-syarat tertentu dipenuhi dengan menggunakan aplikasi yang dibangunkan. Selain itu, pemilik restoran boleh melihat maklum balas pelanggan dan laporan jualan melalui aplikasi ini. Akhir sekali, kakitangan restoran boleh melihat pesanan pelanggan dan mengemas kini status pesanan dari semasa ke semasa.

ABSTRACT

Food is one of the most essential things that humans need. We need food for energy to carry out our daily routine. However, it is a time-consuming process for placing an order in a restaurant. Moreover, most restaurants today are still using a paper-based ordering method which has a high error rate due to the staff need to have a good memory. Besides, orders may also be lost during the process of delivering among the staff. Hence, the rate of delivering a wrong order is high and customers' dining experience will also be affected especially during peak hours. Therefore, Order Management System for Restaurant (OMSR) is proposed system to help in dealing with these problems. The main focus of this project is to develop a cancellation ordering application for restaurant with a single click. Three existing ordering applications have been reviewed to achieve this aim and produce a better ordering system with cancellation function at the end of this project. The expected output of this project is an order management system that provides various functions to the user. For instance, the customer can place order, view the order status of placed order, view the order history, provide feedback and cancel their order if it met the constraint via the application developed. On the other hand, the business owner can view customers' feedback and sales report via the application. Last but not least, the restaurant staff can view customers' orders and update the status from time to time.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ERD	Entity Relationship Diagram
ICT	Information and Communications Technology
IDE	Integrated Development Environment
MVVM	Model-View-ViewModel
OMSR	Order Management System for Restaurant
OS	Operating System
POS	Point of Sale
RAD	Rapid Application Development
SDD	Software Design Document
SDK	Software Development Kit
SRS	Software Requirement Specification
UAT	User Acceptance Testing
URL	Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 Introduction

There are significant growth opportunities for every company that decided to digitalize its business through the development and major success of digital businesses over the past decade. (Dahake & Bhoi, 2019) Digitalized business can expand their available market and sell their product to a larger community. Nowadays, online food ordering has become the trend of the world. (Shahjee, 2016) This is because food is one of the fundamental needs of humans. We need food for energy to carry out our daily work as a student or as an employee. The other reason online food ordering has become the trend of the world is that most people today are busy with work. People are usually lazy to cook because they are tired after working for 8 hours and above. It will be easier for them to order the food from any available restaurants.

In the traditional method, the customers are required to present physically at the restaurant and interact with the staff in order to make their order. The staff will then record the order and deliver the order to the kitchen to be cooked. The customers will encounter several inconveniences as a result of the regular method. For example, orders are lost due to human factors, the food served is different from what has been ordered and so on. All of this will have an impact on customer satisfaction with the restaurant's service. Customers' involvement with service brands is strongly linked to positive emotional factors. (Ahn & Back, 2017) Customers who are satisfied with their service experience are more likely to form a deep link with the business. (Ahn & Back, 2017) Automated processes are believed to be more efficient, dependable, and precise than those that require the involvement of humans, and it is often claimed that a machine can accomplish a task for less money than a person. (Subramanian, 2018) Therefore, an

ordering application can help in preventing the situations mentioned from happening and brings benefits to both the restaurant and the customers.

This study proposes an ordering application for restaurants which aims to digitalize the current business process of the restaurants, improve customers' experience as well as reduce operating expenses of the business.

1.2 Problem Statements

In the traditional method used by restaurants, manual mistakes happen all the time. (Singh et al., 2020). Despite advances in information and communications technology (ICT), majority of the restaurants still use a paper-based ordering method today which has a high error rate. This is because the kitchen staff and the waiters have to remember the details of the order made and deliver the meal to the correct customers. As we all know, humans are forgetful. When they are busy, the memory retention of meal orders is short. Hence, some orders made tend to be lost during the process of receiving and sending out orders to the kitchen staff as well as the cashier due to poor management of the order. For instance, during peak hours, small restaurants usually face a shortage of staff to run the restaurant properly. As a result, the customers will have a bad experience and indirectly the reputation of the restaurant will also be affected. The proposed system allows waiters to only focus on delivering meals from the kitchen to the customers, eliminating the need to take orders and seat customers. Relatively, customers can make their orders through the application provided. The customers can also check their order status after the order is made. This can help the customers to know in what state their order is and in deciding if they want to cancel their order. In the current method, the customers need to inform the waiters first before they can cancel the order. After that, the waiters will then only inform the kitchen staff about the request. Nevertheless, the waiters might be late to inform the kitchen or forget due to peak hours. As a result, the customer is unable to cancel their order. However, with the help of the system, the customers are able to cancel their order with a single click as long as the order is not yet been cooked then the system will inform the kitchen directly without any delay. Furthermore, the customers can also made feedback to the restaurant to help the restaurant in improving its service. By having this system, customers' experience can be improved and the error rate of delivering a wrong order as well as order loss can be minimized. Besides, the system also minimizes the amount of time wasted by the customer as they can make an

order for takeaway or dine in before they reach the restaurant. In addition to that, the business owner will be able to reduce the operating expenses as the number of staff required by the restaurants can be minimized. In conclusion, the system helps facilitate the communication between the customer, waiters, cashier, and the kitchen staff and ensures a smoother workflow at a lower error rate compared to the current method.

1.3 Objective

i. To determine the existing flow of cancellation in ordering applications in the market.

ii. To develop a cancellation ordering application for the restaurants with a single click before the order has been cooked.

iii. To validate the functionality of the developed ordering application for the restaurants.

1.4 Scope

- User Scope

i. End-user.

ii. Restaurants who intended to digitalize their business flow.

- System Scope

i. Covers basic online ordering applications functions enhanced with cancellation function.

ii. The proposed application will be a mobile-based application whose main language of programming language will be Dart language.

- Development Scope

i. Contains multimedia elements such as graphics and text.

ii. Using Firebase as the cloud storage, Github as version control tools and Flutter as the framework.

1.5 Significance of The Project

i. Customers

- Customers can have a better experience with the restaurant.
- Customers can provide feedback to the restaurant to improve their service.

ii. Business Owners

- Operating expenses of the business can be minimized.
- Business owners can view the performance easily.
- Business owners can review the feedback from customers and made an improvement to their service.

iii. Restaurant Staff

- Workload of restaurant staff can be reduced and focusing on a simple task.
- Error made by the restaurant staff can be minimized.

1.6 Report Organization

This report contains 3 chapters. All chapters will describe each part of the system. Chapter 1 describes the overview of the project which includes the Introduction, Problem Statements, Objective, Scope, Significance of the project and Report Organization. Chapter 2 briefly describes the literature review on three ordering applications existing in the market. Chapter 3 explains the methodology and the approach used throughout the development of the project. The details of project requirements are described in Software Requirement Specification (SRS), Appendix A, whereas the details of system designs are described in System Design Document (SDD), Appendix B.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Chapter 2 is about the review of three ordering systems existing in the market. Three existing ordering systems will be discussed and compared in depth. The comparisons will be focused on the purpose, advantages and disadvantages of the existing ordering systems. The comparison of the existing ordering systems highlights the strength and the weaknesses, such that a better ordering system can be developed in this project.

2.2 Review of Existing Systems

In this section, three existing ordering systems will be discussed in detail. Three ordering systems that have been chosen are the Point of Sale (POS) System, McDonald's app and Pizza Hut app.

2.2.1 System 1 - Point of Sale (POS) System

POS system is a system that combines hardware and software to enable employees to complete certain functions. It is usually the place where the consumer pays for items or services in the shop. (Stubbs & Conrad, 2019) These days, POS systems may be found in retail stores, restaurants, hospitals and practically everywhere else that accepts payments. The main purpose of the POS system is it allows the business to collect payments from customers and track sales.

Figure 2.1 Components of POS System

Source: Software Testing Help (2016)

Figure 2.1 depicts the components of the POS system. The POS systems usually consist of a terminal, display pole, barcode reader, cash register, handheld device, and printer. The terminal is usually a touchscreen device that acts as the main screen on which the transaction details are entered. A display pole is a device that will show the item price after a product has been scanned with a barcode scanner. A barcode reader is used to scan products. The cash register is a device that is used to keep track of money once the cashiers received cash from the customers. The handheld device is the device that handles credit card payments. The printer is utilised to print the customer receipt after each transaction. (Software Testing Help, 2016)

Customers are usually required to either place their order at the counter or find a seat first and wait for the waiter to serve them when they enter the restaurant. If the customers are required to order first before they can get their seats, they must line up and place their order at the counter. On the other hand, if customers are required to find a seat first when they arrive at the restaurant, they will be served by a waiter assisting them to place their orders.

The strength of the POS system is that it provides error control where human errors and miscommunication can be reduced. (Biswas, 2018) By using a POS system, the orders are less likely to be lost and misinterpreted by the restaurant staff as compared to handwritten orders. Besides, it also shortened the time it takes to take orders from customers because the staff can take orders by using the terminal instead of handwritten. Furthermore, the POS system also provides accurate business reports regularly which

helps the management to track sales, credit, stock, inventory, most popular goods, and other areas to estimate profits or losses, assisting them in making better decisions. (Biswas, 2018) In conclusion, the POS system helps in reducing human errors and miscommunication, shortening the time taken to place an order as well as providing accurate business reports to the management.

The weakness of the POS system is that the customers do not know the details of their last order. They must recall their latest order to reorder. Moreover, the customers might lose their patience while waiting for their turn as they need to be present physically to place their orders. Although the POS system reduces the time taken to take orders, the customers must still wait for a while before their can order. Besides, the POS system requires a certain number of staff to ensure that the business flow is smooth as the system will only be operated by the staff. In addition, the process to cancel an order is complex. Customers need to inform the waiters and the waiters will then inform the kitchen staff. However, the waiters usually have their hands full. Hence, the cancellation might not reach the kitchen before the food has been cooked. Furthermore, feedback from customers is hard because the customers need to request feedback form from the staff. Customers also find it tedious to keep track with their order status because they must interact with the waiters which will burden the waiters, especially during peak hours. Hence, for the POS system, restaurants using it will require a large workforce, the customers must be physically present to order, the tracking of orders and the order cancellation process is complex, and there is no feedback functionality.

2.2.2 System 2 - McDonald’s

McDonald's is a well-known international fast-food business with its headquarters in the United States. In 2015, McDonald's made a significant move to compete in the fast-food industry's digital space which is the launch of the McDonald’s App. (Samuely, 2017) The purpose of this McDonald’s app is to raise and enhance profits by increasing the accessibility of customers to the business. For the time being, mobile device is very common and almost everything can be done with the mobile device. Hence, McDonald’s app provides the easiest and most convenient option for customers to place orders.

McDonald’s app is free to download and install from both the Play Store on android OS and the App Store on iOS. It is easier and faster to place an order from the

app instead of queueing up at the counter and waiting for our turn to place an order. In order to place an order, the customers are required to sign in to the app or sign up for an account if they do not have one. After that, they will be required to select the franchise that they want to order from and the menu available will be displayed to the customers. The customers can then proceed to select their menu as well as the “pick up method” and lastly place their orders.

McDonald's app allows the customer to view and choose menu available under a selected menu category. Each menu is provided with a clear picture and price for the ease of customers. Besides, it allows the customer to choose their preferred way to get their food such as table service, take away and drive-through. Moreover, the app also allows the customer to check their previous orders history and perform the re-order action if they wanted to. Furthermore, there is a feedback form integrated in the application which allows the customers to provide their feedback regarding their dining or ordering experience anonymously. McDonald's is running with a minimum number of labour force as it uses the online ordering application that reduces the needs of interaction between staff and customers. As a result, the staff can focus on prepare and deliver meals for the customers. Indirectly, the probability of making mistakes such as lost or incorrect orders can be reduced.

The downside of the McDonald’s app is that it does not show the menu that is popular among the customers. Furthermore, it also does not provide the option to the customers to keep track with their order status. Moreover, McDonald’s app does not allow the customers to cancel their order once it is placed. Hence, implementing these missing yet crucial features will greatly increase the functionality of the application.

2.2.3 System 3 - Pizza Hut

Pizza Hut, the largest pizza chain in Malaysia. According to Pizza Hut Malaysia, the Pizza Hut franchise has grown to over 350 locations in Malaysia. (Pizza Hut, 2019) Pizza Hut has offered pizza delivery ordering on the internet in 1994. (Pizza Hut, n.d.) This made them have prior knowledge on doing online food ordering today. In 2009, Pizza Hut launched its ordering application on iPhone. Besides, the success of the Pizza Hut iPhone app led to the launch of the Pizza Hut application on Android devices. The

main purpose of the Pizza Hut app launched is to ensure their cuisine is easily accessible to customers on the platforms they prefer. (Hospitality Technology, 2011)

Customers can install the apps from both Play Store as well as App Store. Customers can choose to sign in or continue as a guest. However, if they intended to place an order, they are required to sign in first. After signing in, customers will be able to choose their intended menu and select whether they want to self-pickup or delivery as well as the location to pick up the pizza.

The advantages of the Pizza Hut app are that it allows the customer to choose their preferred way to get their food and the time. This allows the customers to place their orders in accordance with their schedule in advance. It also provides multiple methods for the customers to pay for their orders. Besides, the app allows the customers to keep track with their order status and view the estimated collection time. Furthermore, the app also allows the customers to view their order history as well as their latest orders. In addition, it provides a survey form to the customers to gather their feedback about their experience with the outlet. Moreover, with the use of online ordering, the number of workers required to ensure the business runs efficiently is lesser than the traditional method. The staff can direct their focus on preparing food instead of taking orders. Hence, the Pizza Hut app reduces the workforce needed in the outlet, allows the customers to track their order status, view their latest order details, provide feedback as well as choose the preferred ways to get the food, time to pick up and the payment methods. Overall, the Pizza Hut application succeeds in helping outlets reduce the required workforce, helps improve business operations through customer feedback, as well as conveniences customers by providing a variety of payment methods, and ways to get the food.

The weakness of the Pizza Hut app is that it does not display the most popular food among consumers which acts as a reference to the customers. Furthermore, it also does not allow the customers to cancel their orders once they have been placed on the app. Although it provides a shortcut that allows the customers to have a quick view of the latest order, it does not provide the shortcut that allows customers to reorder the pizza. In conclusion, the Pizza Hut app does not show the most popular food to the customers and does not allow them to cancel their orders as well as reorder the orders. The most popular food recommendations and shortcuts to reorder may seem to be minor details, however, it helps customers in making decisions faster.

2.3 Comparisons of Three Existing Systems

Table 2.1 summarises the comparison of the functions provided by the three existing ordering systems, based on the review completed in Section 2.2

Table 2.1 Comparison of Three Existing Ordering Systems with Proposed Apps

Function	**POS System**	**McDonald's App**	**Pizza Hut App**	**Proposed App**
Cancel Order	Available, but highly dependent on the staff	Unavailable	Unavailable	Available
Track Order Status	Unavailable	Unavailable	Available	Available
View Order History	Unavailable	Available	Available	Available
Place Order via Internet	Unavailable	Available	Available	Available
Reordering	Available, but highly dependent on the customers	Available	Unavailable	Available
Gather Customer Feedback	Available, but highly dependent on the customers	Available	Available	Available
Sales Reports	Available	N/A	N/A	Available
Number of Staff Required to Run the Business Smoothly	High	Low	Low	Low
Year's develop	1973	2015	2009	2022
Programming Language	C / C++	Android Native	React Native	Dart
Database	SQL / MySQL	Firebase	Google Cloud	Firebase

Based on Table 2.1, it was found that the POS system is the only system that can cancel orders while the other two systems are unable to cancel the order. However, the process of cancelling an order for the POS system is highly dependent on the staff because the customers need to inform the staff and the staff need to inform the kitchen staff before their order can be cancelled. If the workload of the staff is full, the cancellation request might be unable to reach the kitchen staff due to human factors. In contrast, the proposed app allows customers to cancel their orders with a single click before the order has been cooked.

The POS system and McDonald's app do not provide the function for the customers to track their order status. In contrast, the Pizza Hut app provides the feature to keep track with the order status as well as give an estimated collection time to the customers. It is important to let the customers keep track with their order status such that they will be able to know if their order is still on hold, being cooking or ready to be served. Overall, the Pizza Hut app is better because the customers are able to estimate the duration that they still have to wait instead of blindly waiting in the restaurant. Hence, the order tracking function will be implemented in the proposed app.

Next, the POS system does not provide the features of viewing order history and placing orders via the internet. This is because the POS system treats all customers as one entity instead of every customer as a distinct entity. Hence, it does not keep the record of orders placed by a particular customer. Besides, the customers have to be present physically to place their orders. Therefore, they cannot place their order via the internet before they reach the restaurant. On the other hand, McDonald's app and Pizza Hut app provide the features of viewing order history and placing orders via the internet as every customer must sign in before they place their orders. As a result, the McDonald's app and the Pizza Hut app are better as the order history is recorded and the order can be placed via the application, customers do not have to recall their memory or wait for a long queue just to place an order. The proposed app also provides the features of viewing order history and placing orders via the internet if the customers are signed in.

Moreover, the POS system and the McDonald's app allow the customers to reorder their orders quickly while the Pizza Hut app is not providing this feature. However, the reorder process in the POS system is highly dependent on the customers. This is because the POS system treats all customers as one entity instead of each customer as a distinct entity. Hence, the customers have to recall the details of the order which is difficult for humans. On the other hand, McDonald's app treats each customer as one distinct entity, so there will be a record of orders placed by each customer. Hence, the customer can reorder their food easily. To summarise, the McDonald's app is doing great among the reviewed systems as it is the only system that allows the customers to reorder easily. The reorder feature will also be implemented in the proposed app to provide the customers with a better experience.

In addition, both the McDonald's app and the Pizza Hut app provide a built-in feedback form to gather feedback from the customers. On the other hand, the restaurant using the POS system also can gather feedback from the customers. However, it will be difficult because the customers have to make their feedback to the restaurant staff which is lack of anonymity. As a result, the feedback gathered from customers will be lesser than using the built-in feedback form due to the effect of communication apprehension. In conclusion, the McDonald's app and the Pizza Hut app are the winners because feedback from customers is important for a business to improve its services.

Furthermore, the POS system provides the function to generate sales reports for the business owner. The business owner is able to access the sales reports via the POS system to evaluate the performance of the restaurant. However, the McDonald's app and the Pizza Hut app are inaccessible from the client's side, hence, analysis or comparison cannot be done. In short, the POS system allows the business owner to evaluate the performance via the sales reports and this function will also be implemented in the proposed app.

Last but not least, the number of staff required by the POS system to run the business smoothly is higher compared to the number of staff required by the McDonald's app and the Pizza Hut app. This is because the POS system is only used by the staff. If the customers wish to place their orders, they have to line up and wait for their turn in front of the counter. The staff are required to place the orders of the customers, collect payments from customers, prepare and serve the food to customers as well as respond to any order cancellation or order tracking from the customers. While the McDonald's app and the Pizza Hut app require the customers to place their orders on their own without involving the staff. The staff is only responsible to prepare the food according to the orders and serving the food as well as collecting payment from the customers. Correspondingly, the proposed app will also require a minimum number of staff as it has similar properties to the McDonald's app and the Pizza Hut app.

To conclude, the McDonald's app and the Pizza Hut app are providing similar features to the customers. Both systems are focusing on simplifying ordering process and running the business smoothly with the minimum number of workers. Hence, the proposed app will inherit their strengths and provide a better service.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter discusses the approaches implemented in the development of the Order Management System for Restaurant (OMSR) on mobile devices. A good software development approach is required to make sure the system can be developed within the budget and schedule. There are many different types of software development approaches in the industry, each with its own set of pros and cons. For this project, the approach chosen to be used in the development is the Rapid Application Development (RAD) approach. A detailed explanation will be presented later in this chapter

3.2 Project Management Framework

The project management framework used in this project is Rapid Application Development (RAD). RAD is a type of agile project management strategy that is widely used in software development. (Lucidchart Content Team, 2018) The reason why RAD is used in this project is that RAD aims to minimize the planning stage and maximize the development of the prototype. (Lucidchart Content Team, 2018) This enables RAD can complete the projects quickly. According to turorialspoint, the functional modules are built-in parallel as prototypes in RAD and then combined to create the whole product for faster delivery. Since the planning is done at a minimum level, hence, it is easier to accommodate the changes that occur throughout the development process. (tutorialspoint, 2019)

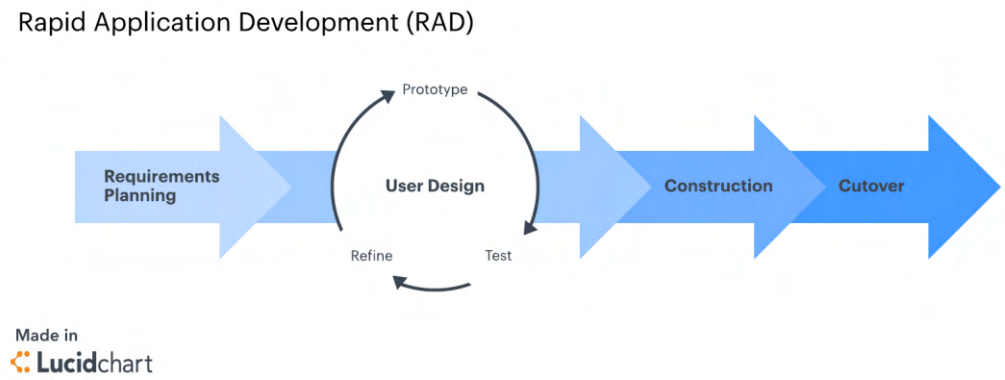

Figure 3.1 Main Phases in Rapid Application Development (RAD)
Source: Lucidchart Content Team (2018)

Figure 3.1 depicts the four main phases involved in Rapid Application Development (RAD). RAD starts with requirements planning, user design, construction and lastly cutover.

In the requirements planning phase, the developers, system users, and team members will communicate to define the goals and expectations of the project, as well as any present or future difficulties which will need to be handled throughout the completion of the project. Even though the planning phase in RAD is shorter than in other project management approaches, it is still an important step in ensuring the success of the project. In short, the requirement planning phase can be broken down into three components which are investigating the present issue, specifying the project's requirements, and getting each stakeholder's agreement on the requirements.

Once the requirement planning phase is completed, the project will move to on next phase which is the user design phase. In this phase, the user design will be built via various prototype iterations. The developers will develop the prototype according to the requirements while the system user will test the prototype and give feedback to refine the prototype. The prototype will be refined until a satisfactory design is reached. Moreover, all the faults found in this phase will be fixed. This is to ensure the prototype produced at the end of the phase will be able to meet the system user's needs and expectations.

After that, the prototypes produced during the design phase will be converted into a functional model in the construction phase. As the system users were satisfied with the prototype design in the user design phase, the developers can proceed directly to build

the final system faster than in traditional project management methodology. This phase involves the steps of preparing for the rapid construction, development of the system and testing the system at different levels such as component testing, integration testing and system testing. This is to ensure the system is able to work smoothly and the final product meets the expectations and goals of the system users.

Lastly, in the cutover phase, the completed system will be ready to be delivered to the market. This phase will include user training, data conversion and switching to the new system.

3.3 Project Requirements

Table 3.1 Functional Requirements of The Proposed System

No.	Functional Requirements
1.	The system shall allow the customers to place their order via the internet.
2.	The system shall allow the customers to view the order status and order history.
3.	The system shall allow the customers to cancel their order with a single click as long as the order has not been prepared.
4.	The system shall allow the customers to reorder their orders.
5.	The system shall allow the customers to give feedback to the restaurant.
6.	The system shall allow the restaurant staff to view and update the status of customers' orders.
7.	The system shall allow the business owner to view customers' feedback.
8.	The system shall be able to generate sales reports for the business owner.

Table 3.2 Non-Functional Requirements of The Proposed System

Quality Attribute	Non-Functional Requirements
Security	Users' information shall be protected and secured.
	The account registered shall be verified via email.
Availability	The system should be available according to the business hour of the restaurant.
Usability	The time taken for the user to become familiar with the system should not be more than 5 minutes.
Accuracy	The sales reports must be generated according to the data in the system database.
Performance	The order status shall be updated in the database within 1 second after the restaurant staff updated the order status.
	The updated order status shall be displayed to the relevant restaurant staff within 1 second.
Interoperability	The system shall be able to run smoothly on android mobile devices.

Table 3.3 Constraints of The Proposed System

No.	Constraints
1.	Internet connection is required for the system to retrieve data from and save data to the system database.
2.	The system must be completed before the author has completed the course "BCC3024 Undergraduate Project II".
3.	The system is only developed by the author and the time given to the author is also limited. Hence, the functionality of the prototype is limited.
4.	This project is for education purposes. Hence, all tools and software used in the development of the prototype are under free tier subscriptions.

Table 3.4 Limitations of The Proposed System

No.	Limitations
1.	The system does not support the function of modification of staff data and menu data.
2.	The system currently only supports cash payment at the counter.
3.	The system currently only supports android mobile devices.

3.4 Proof of Initial Concept (Storyboard)

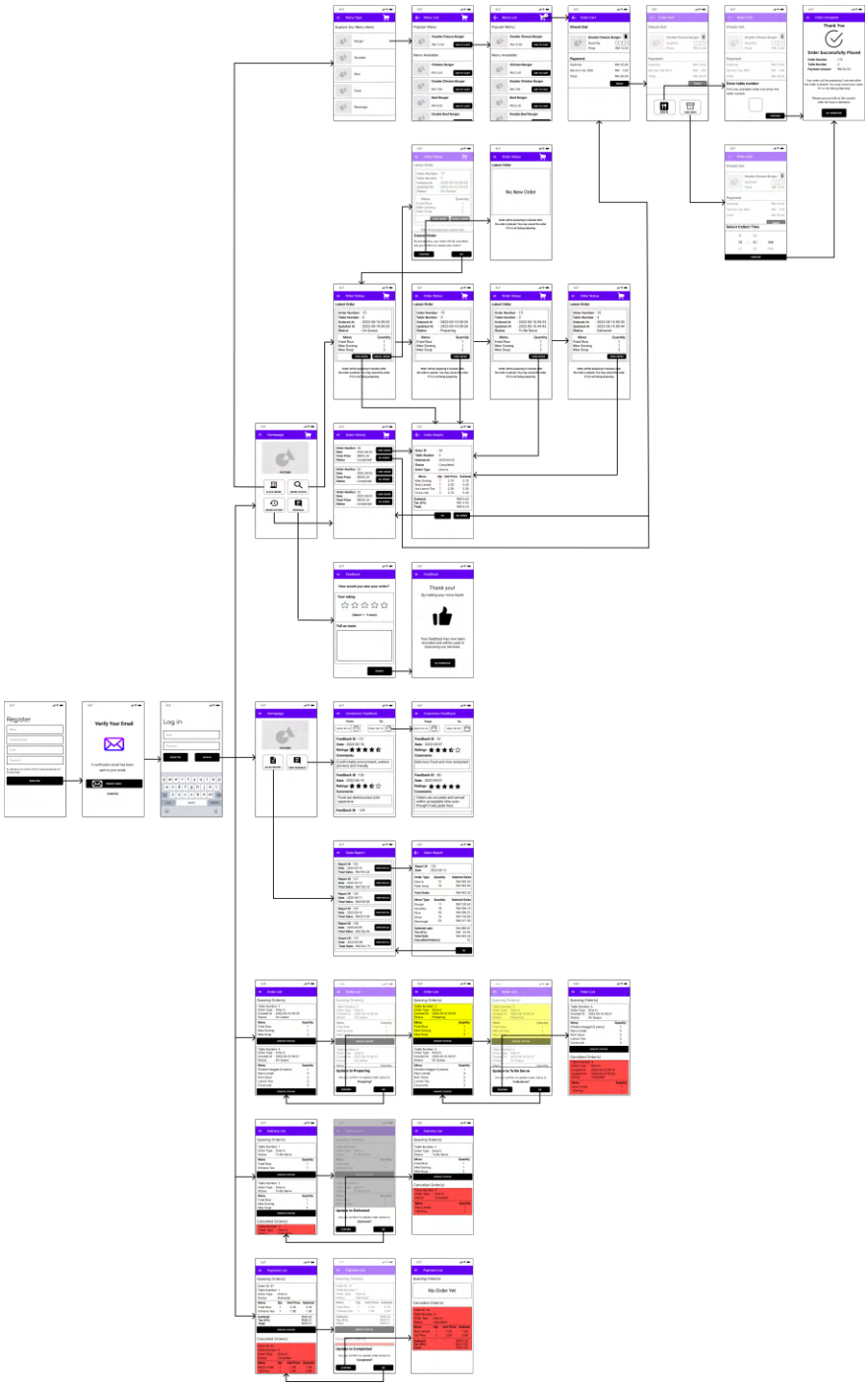

Figure 3.2 Storyboard of Proposed System

Figure 3.2 shows the storyboard of the proposed system which depicts the flow of interfaces of all modules. The proposed system consists of seven modules which are place order module, view order status module, view order history module, provide feedback module, view feedback module, view sales report module and update status module.

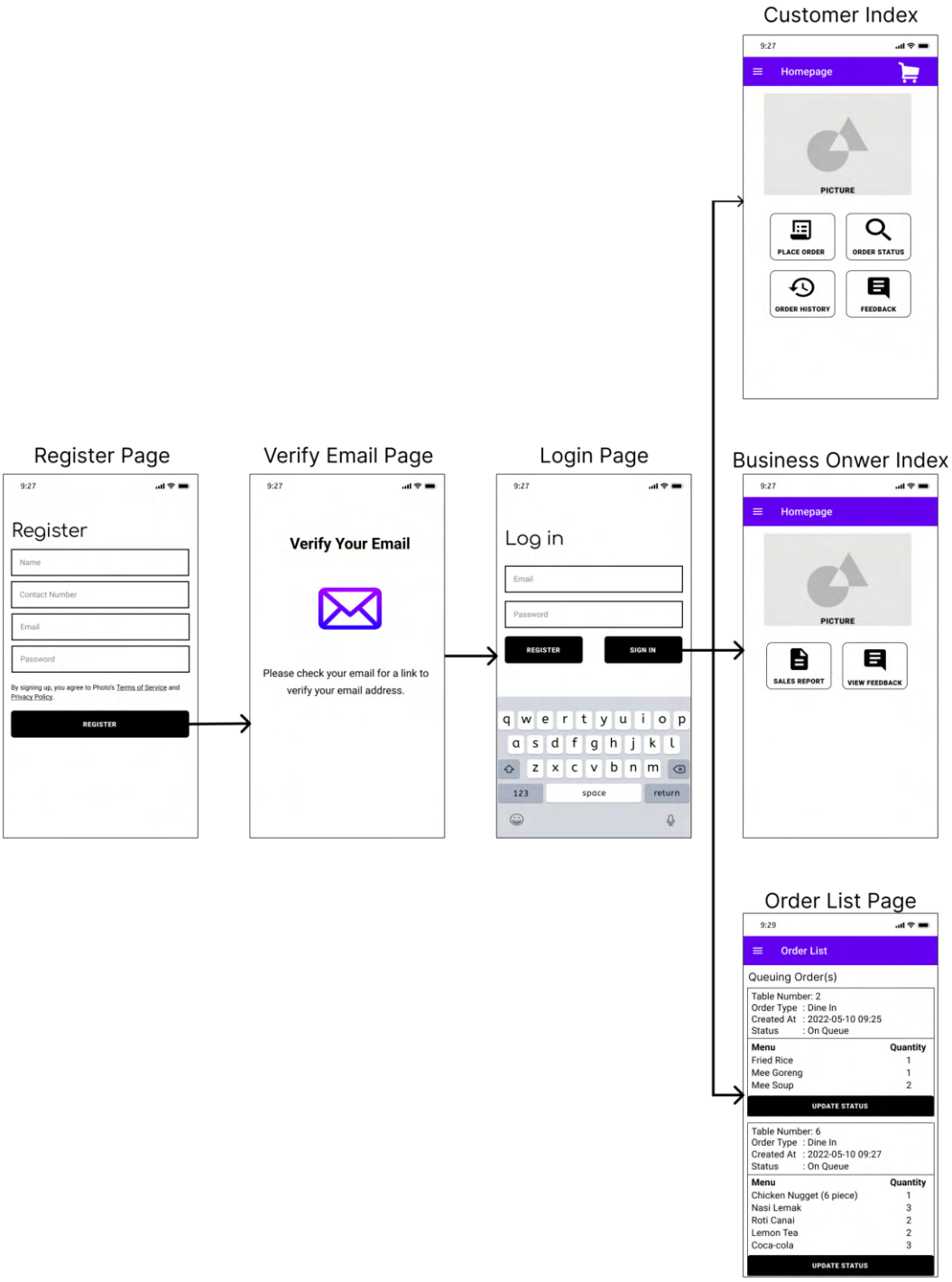

Figure 3.3 Flow of Interfaces for Register and Login

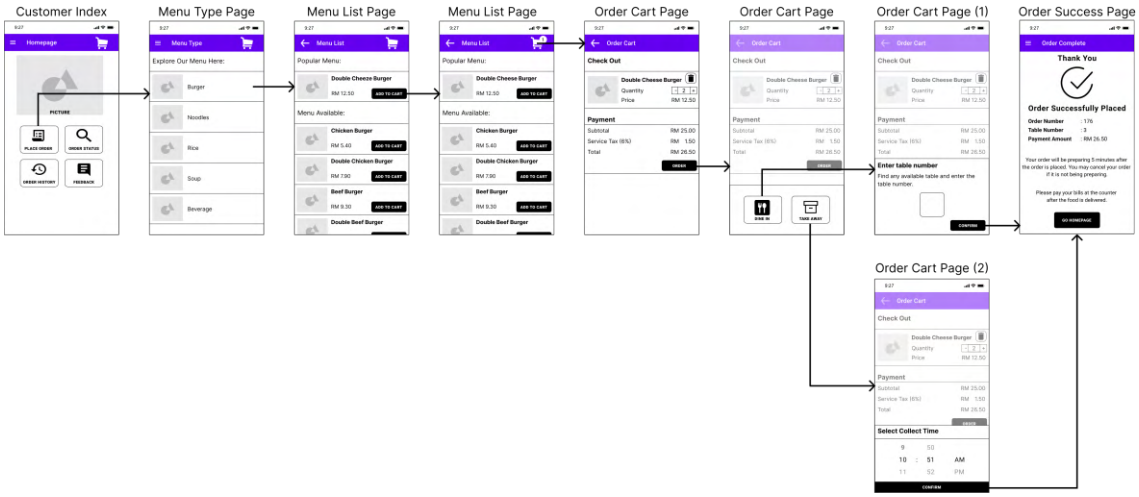

Figure 3.4 Flow of Interfaces for Place Order Module

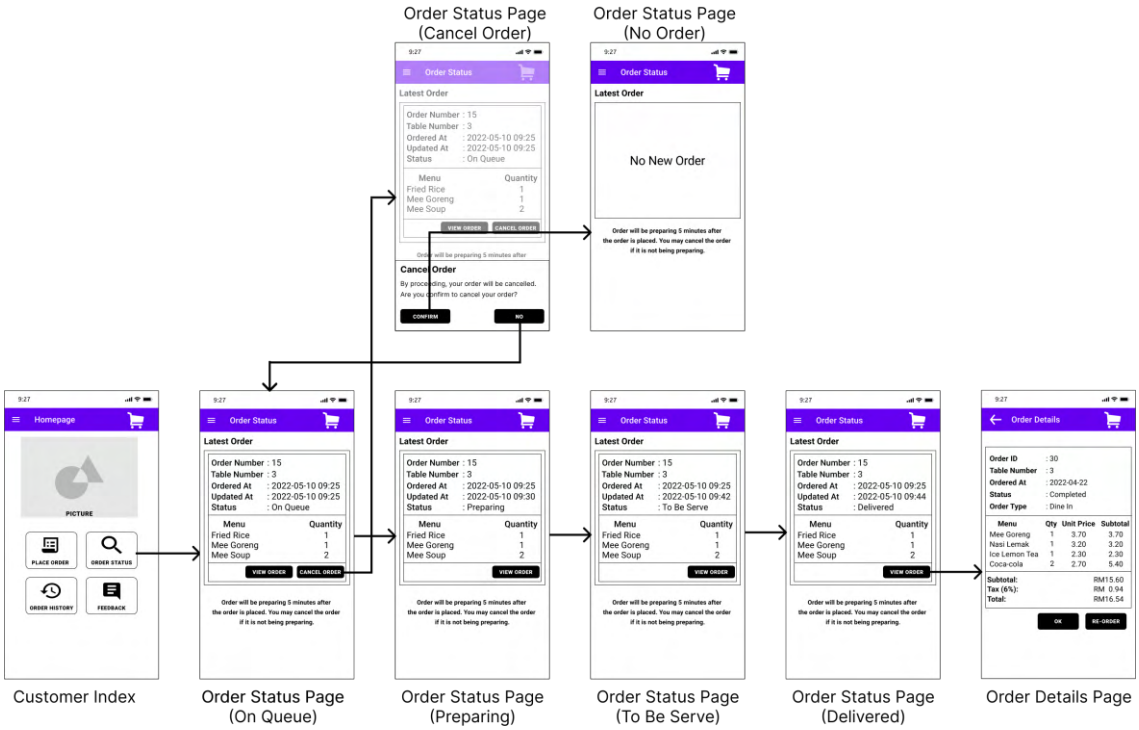

Figure 3.5 Flow of Interfaces for View Order Status Module

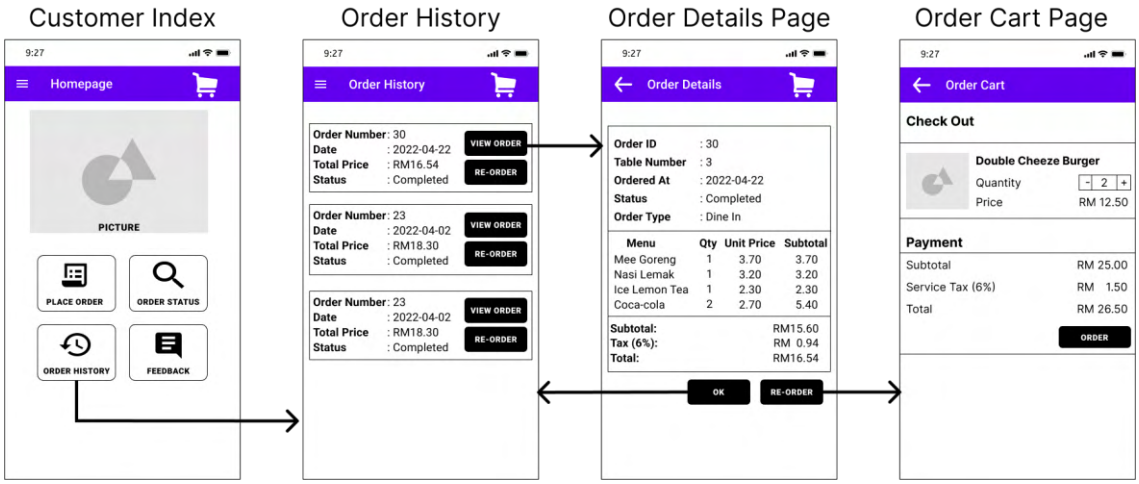

Figure 3.6 Flow of Interfaces for View Order History Module

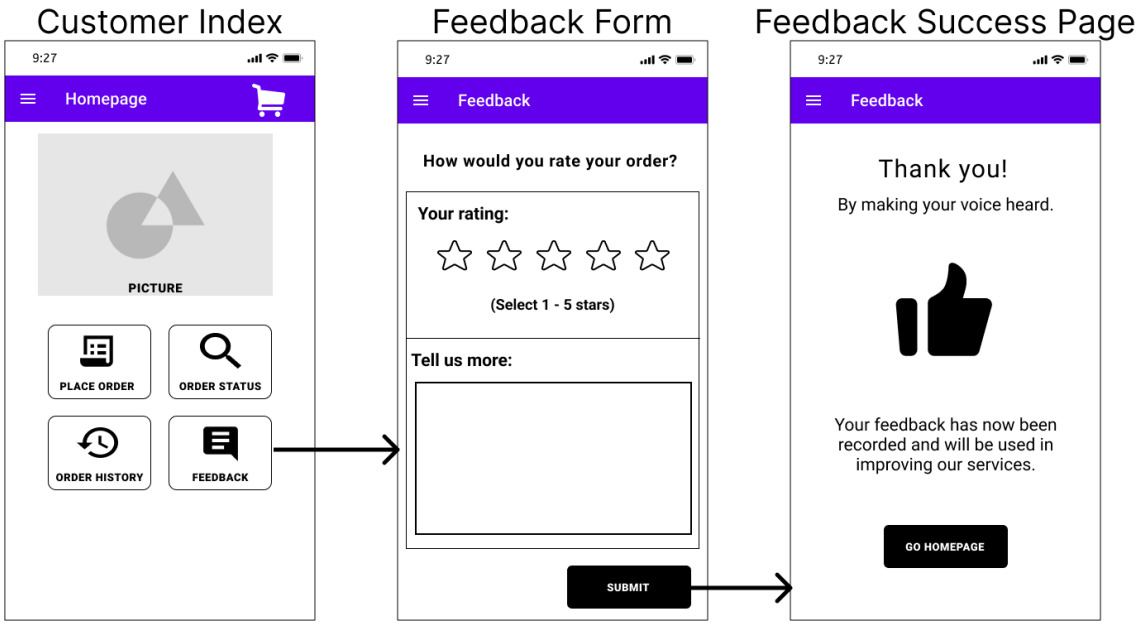

Figure 3.7 Flow of Interfaces for Provide Feedback Module

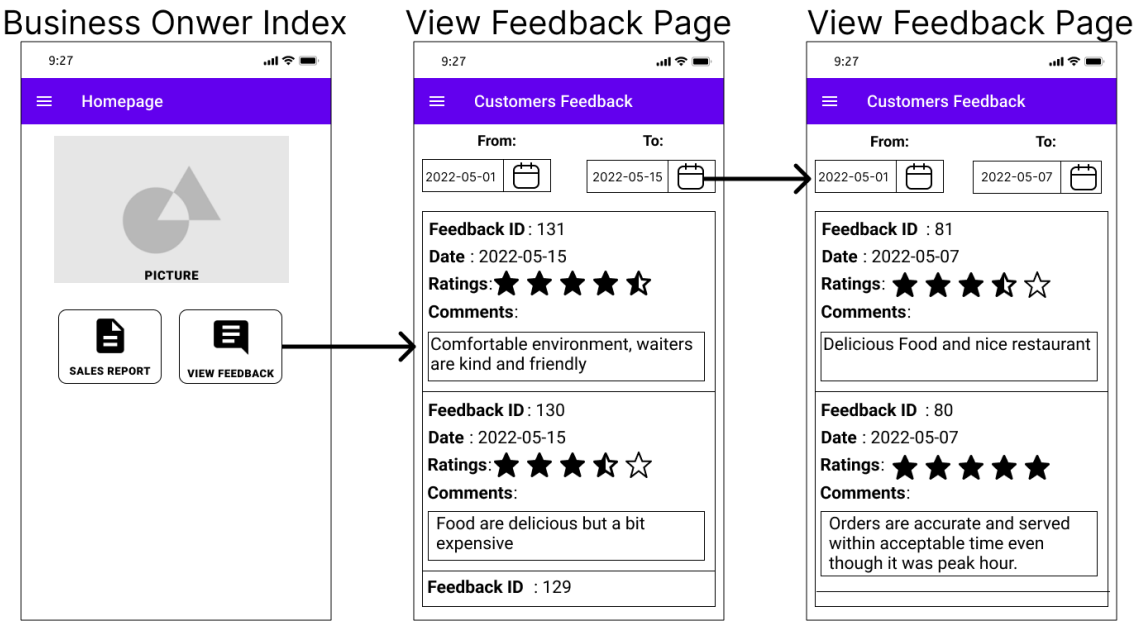

Figure 3.8 Flow of Interfaces for View Feedback Module

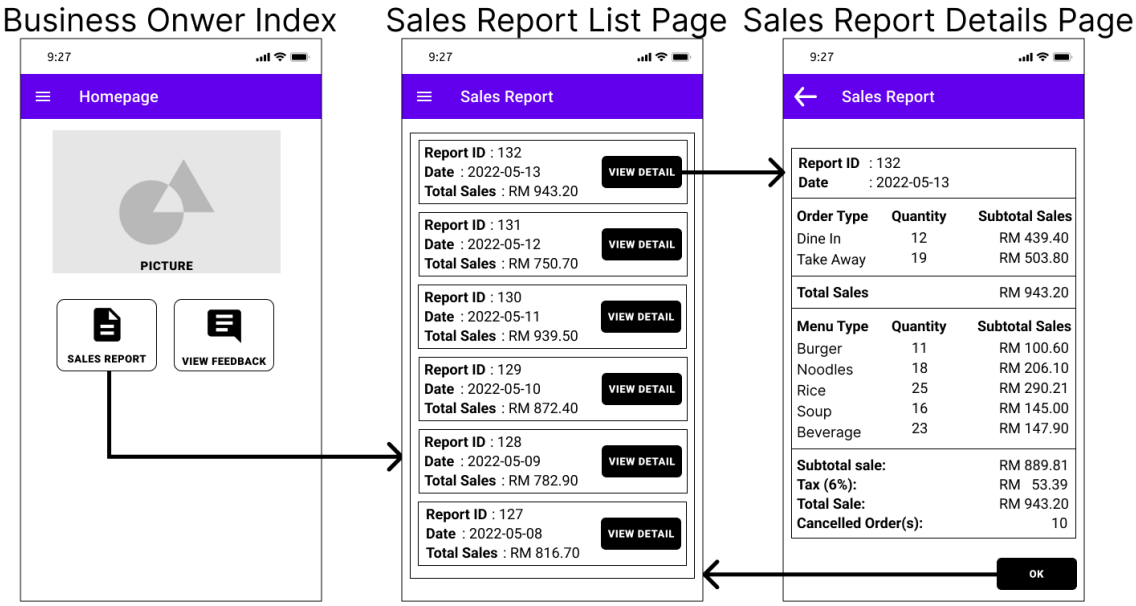

Figure 3.9 Flow of Interfaces for View Sales Report Module

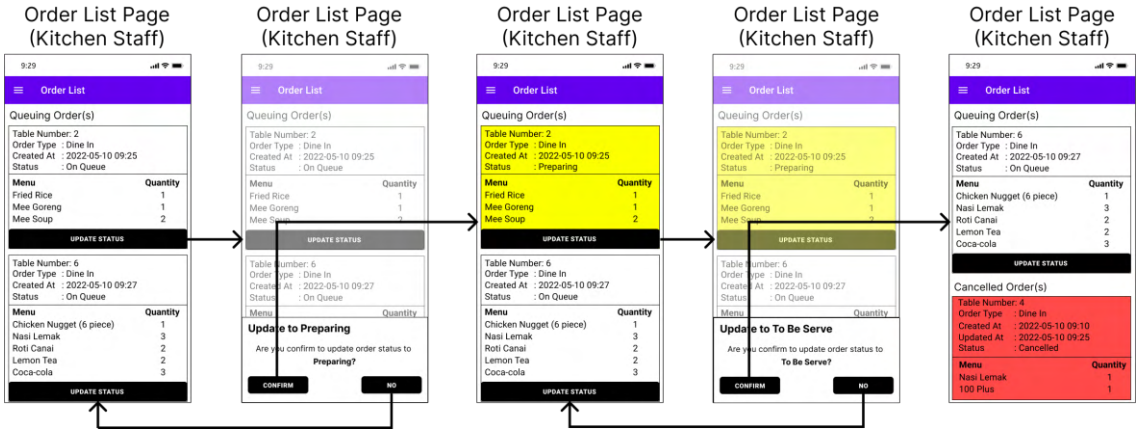

Figure 3.10 Flow of Interfaces for Update Order Status Module (Kitchen Staff)

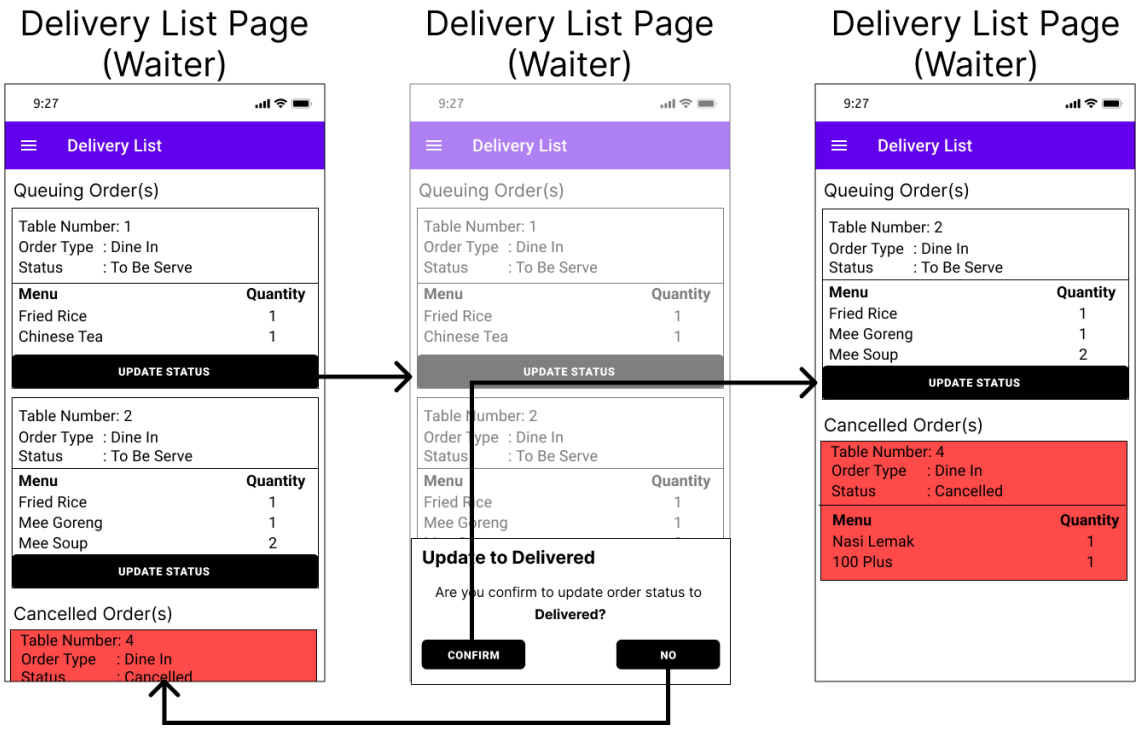

Figure 3.11 Flow of Interfaces for Update Order Status Module (Waiter)

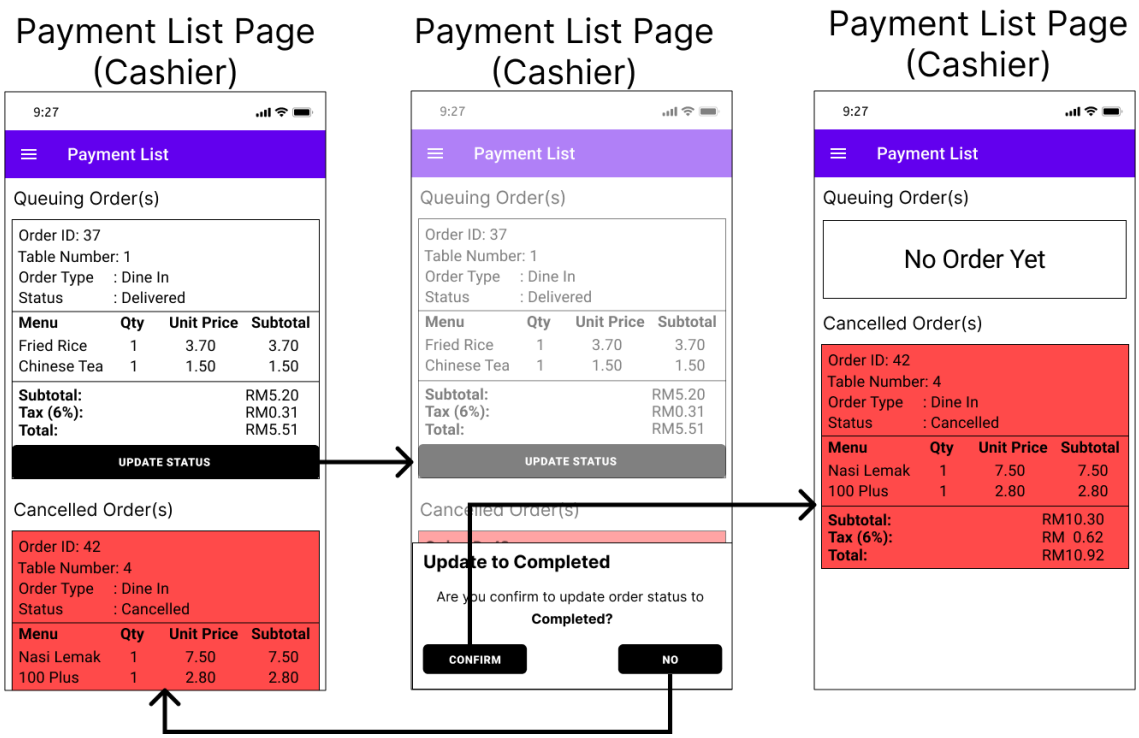

Figure 3.12 Flow of Interfaces for Update Order Status Module (Cashier)

3.5 Proposed Design

3.5.1 Context Diagram

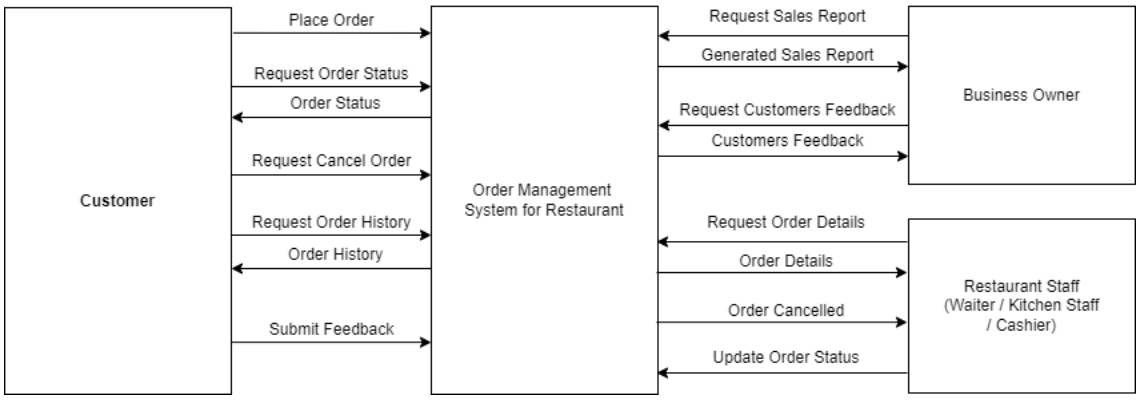

Figure 3.13 Context Diagram of Proposed System

3.5.2 Use Case Diagram and Description

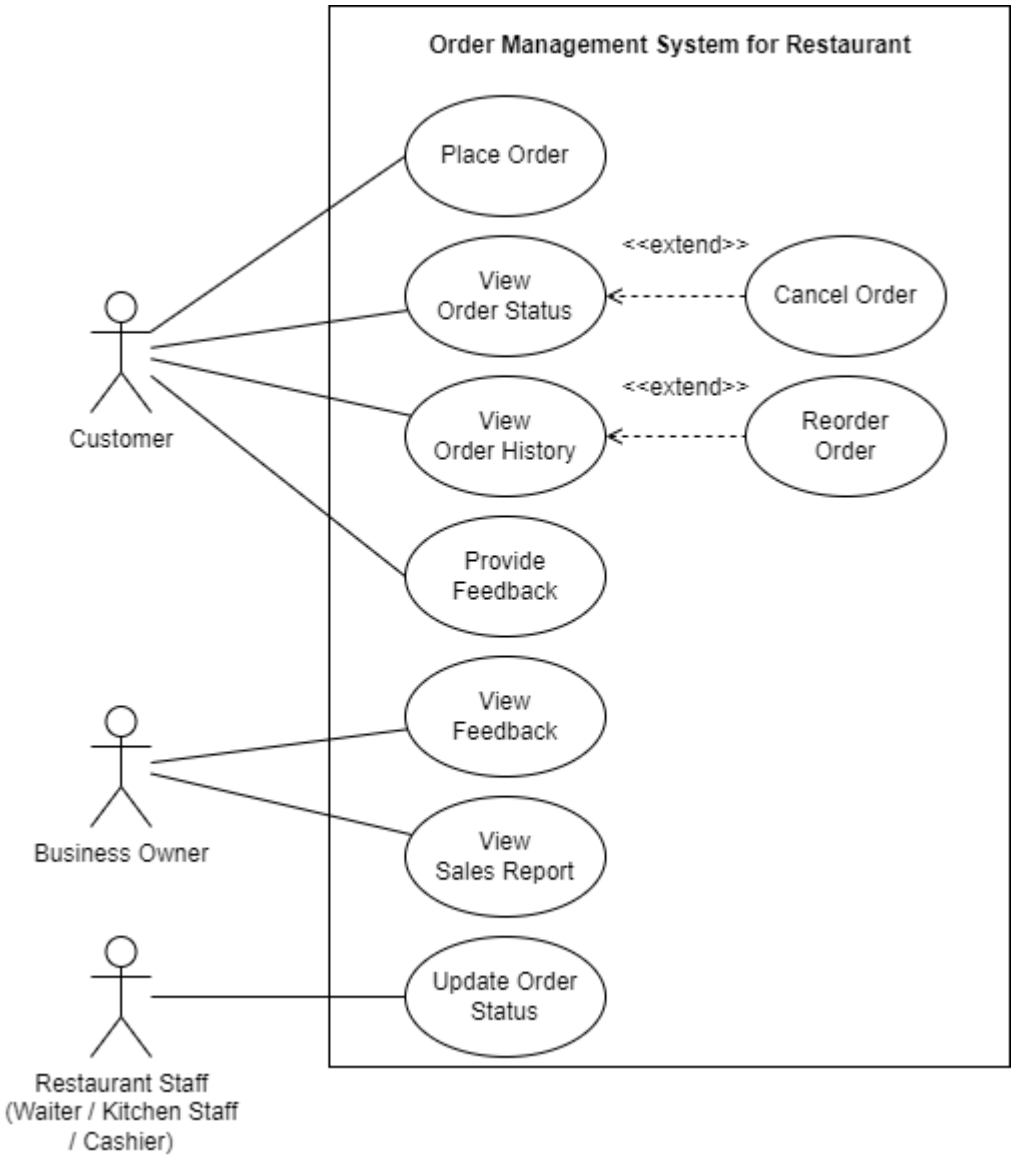

Figure 3.14 Use Case Diagram of Proposed System

Figure 3.14 shows the use case diagram of the order management system for restaurant. According to the figure, there are three types of users using the system. The first user is the customer, who can place an order, view order status, cancel an order under certain conditions, view order history and provide feedback. The next user is the business owner who can view the feedback provided by the customer and view the sales report generated by the system. Lastly, restaurant staff are the third user which consists of waiters, kitchen staff and cashiers. The restaurant staff can view and update the order status of the customer.

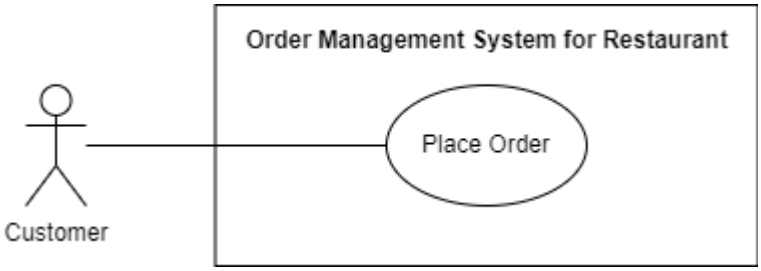

Figure 3.15 Use Case Diagram of Place Order Module

Table 3.5 Use Case Description of Place Order Module

Use Case ID	UC001
Brief Description	This use case describes the process of the customers placing an order.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet.
Basic Flow	**[B1: Place Order]** 1. Use case starts when the customer presses the "Place Order" button. 2. System retrieves menu data from the menu database. 3. System displays menu data in the "Menu" page. 4. Customer presses intended menu type. 5. System displays available menu under menu type selected. 6. Customer presses the "Add to Cart" button to add an intended menu to the order cart. 7. System retrieves data of the intended menu and adds it to the order cart. 8. Customer presses the "Order Cart" button to view their intended order. 9. System retrieves order cart data from order cart database. 10. System displays order cart data in the "Order Cart" page. 11. Customer presses the "Order" button to place their order. 12. System prompts the customer to select an order type. **[A1: Dine In] [A2: Take Away]** 13. System saves data into order database. 14. Use case ends.
Alternative Flow	**[A1: Dine In]** 1. Use case starts when the customer presses the "Dine In" button. 2. System prompts the customer to input their table number. 3. Customer inputs table number and presses the "Confirm" button. 4. Back to basic flow step 13. **[A2: Take Away]** 1. Use case starts when the customer presses the "Take Away" button. 2. System prompts the customer to select a collection time. 3. Customer selects collection time and presses the "Confirm" button. 4. Back to basic flow step 13.
Exception Flow	NONE
Post-Conditions	Customer will be redirected to the home page.

Rules	**[R1: Table Number]** 1. Customer can only input the table number where they sit on. **[R2: Collection Time]** 1. Customer can only select a time within the range of restaurant business hours.
Constraints	Only the registered customer will be able to place an order.

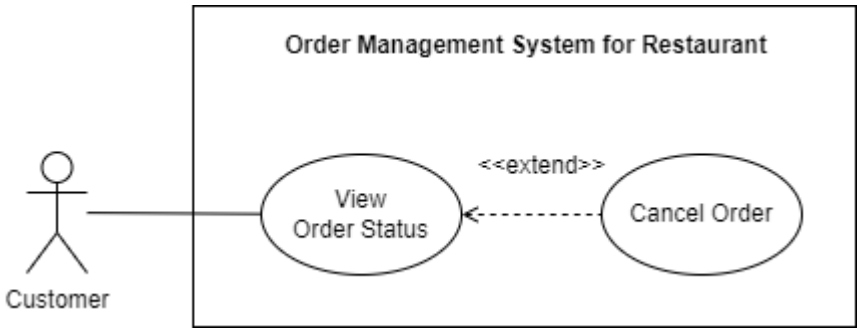

Figure 3.16 Use Case Diagram of View Order Status Module

Table 3.6 Use Case Description of View Order Status Module

Use Case ID	UC002
Brief Description	This use case describes the process of the customers viewing order status and cancelling their order if they want to.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet. 3. Customer had placed an order and the order status is not "Completed" or "Cancelled".
Basic Flow	**[B1: View Order Status]** 1. Use case starts when the customer presses the "Order Status" button. 2. System retrieves customer's order data from the order database. **[A1: No Order Data]** 3. System displays order data in the "Order Status" page. 4. Use case ends. **[B2: Cancel Order]** 1. Use case starts when the customer is at the "Order Status" page. 2. Customer presses the "Cancel Order" button to request to cancel an order. 3. System prompts confirmation message. **[A2: Press No]** 4. Customer presses the "Confirm" button to cancel the order. 5. System updates order status to "Cancelled" in the order database. 6. Use case ends.
Alternative Flow	**[A1: No Order Data]** 1. System retrieves customer's order data from order database. 2. System unable to retrieve data as the customer does not have an order with a status other than "Completed" or "Cancelled". 3. System displays "No New Order" in the "Order Status" page.

	4. Use case ends. **[A2: Press No]** 1. Alternative flow continues at Use Case of View Order Status Module, Basic Flow, B1: View Order Status, Step 3. 2. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	**[R1: Cancel Order]** 1. Order can only be cancelled when its status is "On Queue".
Constraints	Only the customer will be able to perform this use case.

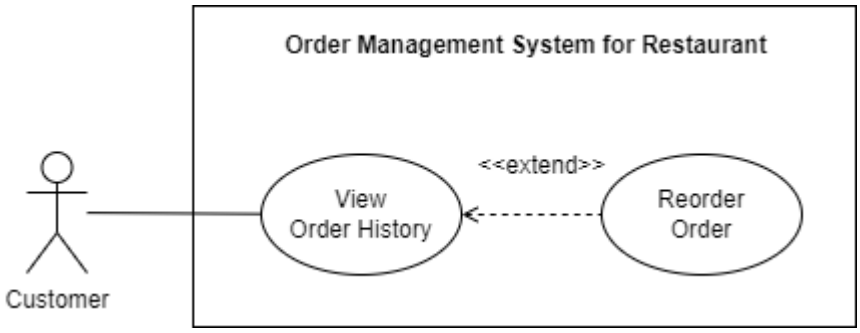

Figure 3.17 Use Case Diagram of View Order History Module

Table 3.7 Use Case Description of View Order History Module

Use Case ID	UC003
Brief Description	This use case describes the process of the customers viewing order history and reordering their previous order.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet. 3. Customer had placed an order.
Basic Flow	**[B1: View Order History]** 1. Use case starts when the customer presses the "Order History" button. 2. System retrieves customer's order data from the order database. **[A1: No Order History]** 3. System displays order data in the "Order History" page. 4. Customer presses the "View Order" button to view order details. 5. System retrieves order details data from the order database. 6. System displays order details data in the "Order Details" page. 7. Use case ends. **[B2: Reorder Order]** 1. Use case starts when the customer is at the "Order History" page. 2. Customer presses the "Reorder" button to reorder an order. 3. System retrieves order data and adds it to the order cart. 4. System redirects the customer to the "Order Cart" page.

	5. Basic flow continues at Use Case of Place Order Module, Basic Flow, B1: Place Order, Step 9. 6. Use case ends.
Alternative Flow	**[A1: No Order History]** 1. System unable to retrieve data as the customer does not place any order since they registered. 2. System displays "No Order History" in the "Order History" page. 3. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	NONE
Constraints	Only the customer will be able to perform this use case.

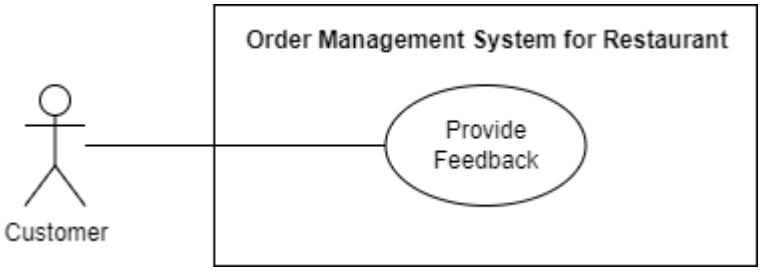

Figure 3.18 Use Case Diagram of Provide Feedback Module

Table 3.8 Use Case Description of Provide Feedback Module

Use Case ID	UC004
Brief Description	This use case describes the process of the customers providing feedback to the restaurant.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet.
Basic Flow	**[B1: Provide Feedback]** 1. Use case starts when the customer presses the "Feedback" button. 2. System retrieves feedback form. 3. System displays feedback form in the "Feedback" page. 4. Customer selects rating and fills their comment. 5. Customer presses the "Submit" button to submit the feedback. 6. System saves data into feedback database. 7. Use case ends.
Alternative Flow	NONE
Exception Flow	NONE
Post-Conditions	Customer will be redirected to the home page.
Rules	NONE
Constraints	Only the customer will be able to provide feedback to the restaurant.

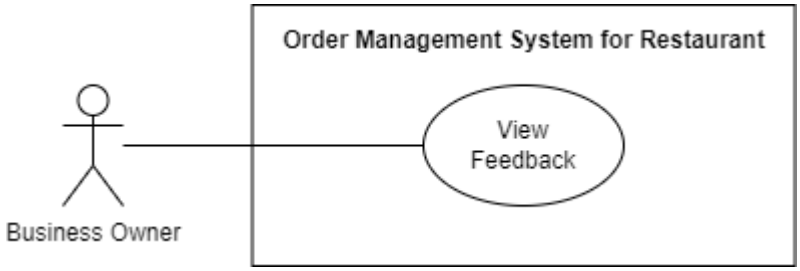

Figure 3.19 Use Case Diagram of View Feedback Module

Table 3.9 Use Case Description of View Feedback Module

Use Case ID	UC005
Brief Description	This use case describes the process of the business owner to view customers' feedback.
Actor	Business Owners
Pre-Conditions	1. Business owners signed in to the system. 2. Business owners' device is connected to the internet.
Basic Flow	**[B1: View Feedback]** 1. Use case starts when the business owners press the "View Feedback" button. 2. System retrieves feedback data from the feedback database. **[A1: No Feedback Data]** 3. System displays feedback data in the "View Feedback" page. **[A2: Filter Feedback Data]** 4. Use case ends.
Alternative Flow	**[A1: No Feedback Data]** 1. System unable to retrieve customers feedback data. 2. System displays "No Feedback Submitted" in "View Feedback" page. 3. Use case ends. **[A2: Filter Feedback Data]** 1. Business owners select date input to filter feedback data. 2. System retrieves and displays filtered feedback data. **[A1: No Feedback Data]** 3. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	NONE
Constraints	Only the business owners will be able to view customers' feedback.

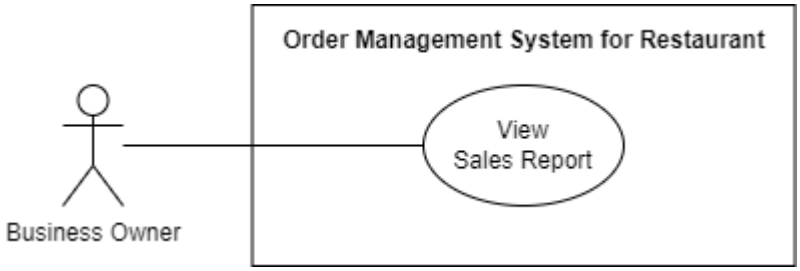

Figure 3.20 Use Case Diagram of View Sales Report Module

Table 3.10 Use Case Description of View Sales Report Module

Use Case ID	UC006
Brief Description	This use case describes the process of the business owner viewing a sales report.
Actor	Business Owners
Pre-Conditions	1. Business owners signed in to the system. 2. Business owners' device is connected to the internet.
Basic Flow	**[B1: View Sales Report]** 1. Use case starts when the business owners press the "Sales Report" button. 2. System retrieves sales report data from the report database. **[A1: No Report Data]** 3. System displays sales report data in the "Sales Report" page. 4. Business owners press the "View Detail" button to view a particular sales report. 5. System retrieves sales report details data from report detail database. 6. System displays sales report details data in the "Sales Report Details" page. 7. Use case ends.
Alternative Flow	**[A1: No Report Data]** 1. System unable to retrieve sales report data. 2. System displays "No Sales Report" in "Sales Report" page. 3. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	NONE
Constraints	Only the business owners will be able to view the sales report.

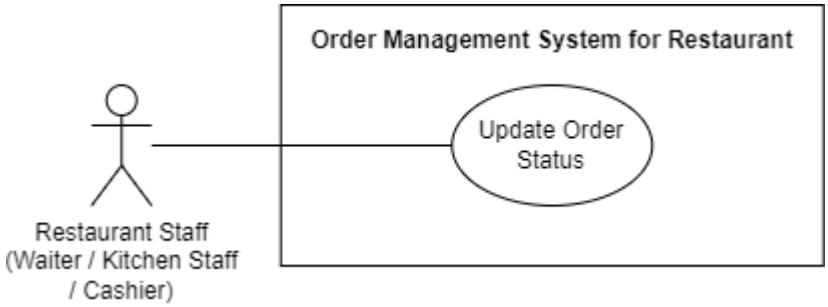

Figure 3.21 Use Case Diagram of Update Order Status Module

Table 3.11 Use Case Description of Update Order Status Module

Use Case ID	UC007
Brief Description	This use case describes the process of the restaurant staff updating order status.
Actor	Restaurant Staff (Waiter, Kitchen Staff, Cashier)
Pre-Conditions	1. Restaurant staff signed in to the system. 2. Restaurant staff's device is connected to the internet.
Basic Flow	**[B1: Update Order Status]** 1. Use case starts when restaurant staff signed in to the system. 2. System retrieves order data from the order database. 3. System displays order data in the "View Order" page. 4. Restaurant staff press the "Update Status" button to update the order status. 5. System prompts confirmation message. **[A1: Press No]** 6. Restaurant staff press the "Confirm" button to update order status. 7. System updates order status in order database. 8. Use case ends.
Alternative Flow	**[A1: Press No]** 1. Alternative flow continues at Use Case of Update Order Status Module, Basic Flow, B1: Update Order Status, Step 3. 2. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	**[R1: Order Data Displayed]** 1. Only order data with the status "On Queue" or "Preparing" will be displayed to the kitchen staff. 2. Only order data with the status "To be Serve" will be displayed to the waiter. 3. Only order data with the status "Delivered" will be displayed to the cashier. **[R2: Order Status Updated]** 1. Kitchen staff will update the order with "On Queue" status to "Preparing" status. 2. Kitchen staff will update the order with "Preparing" status to "To be Serve" status. 3. Waiter will update the order with "To be Serve" status to "Delivered" status. 4. Cashier will update the order with "Delivered" status to "Completed" status.

Constraints	• Only the restaurant staff will be able to view orders and update order status. • The kitchen staff will update the order status from “On Queue” to “Preparing” 5 minutes after the order is placed.

3.5.3 Activity Diagram

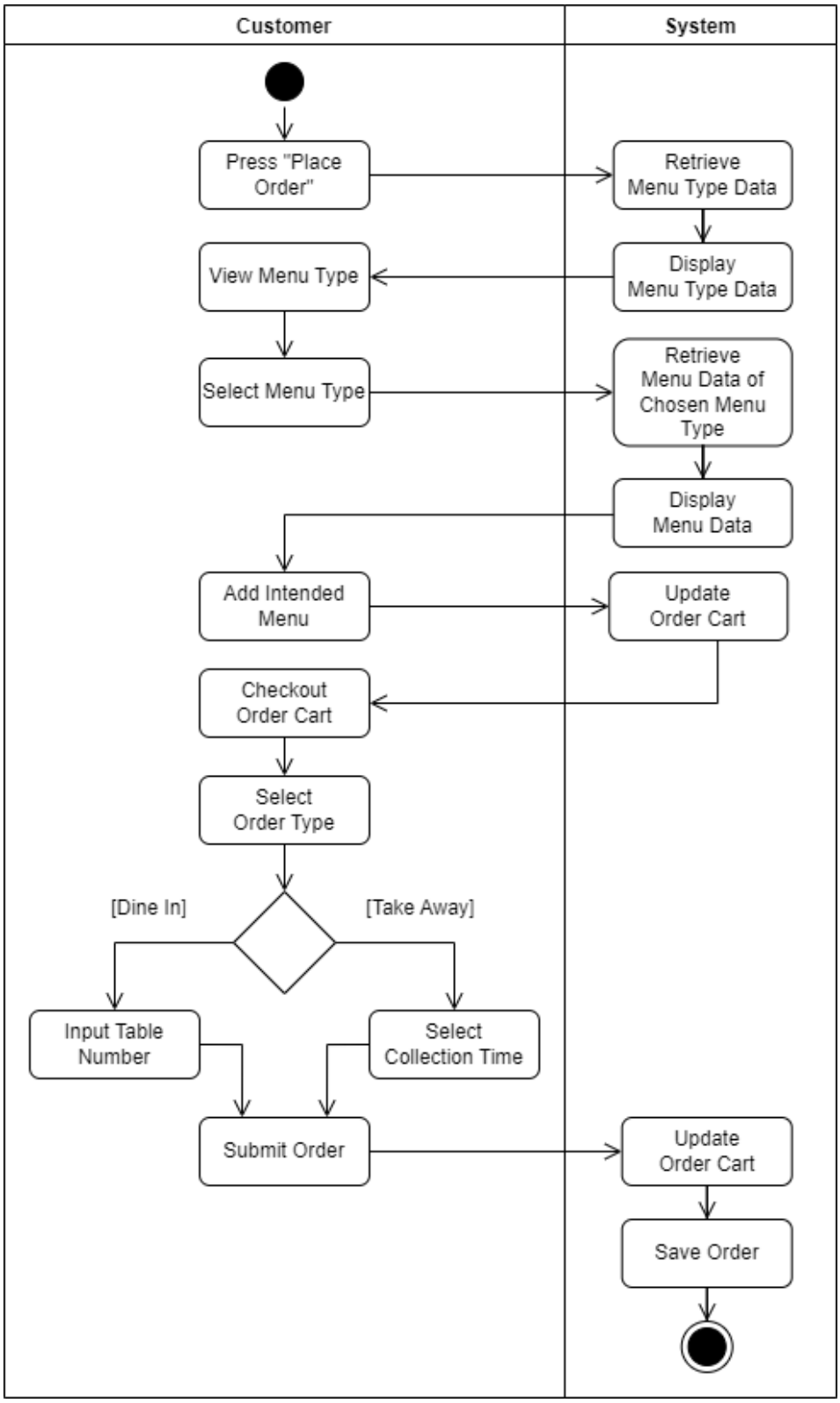

Figure 3.22 Activity Diagram of Place Order Module

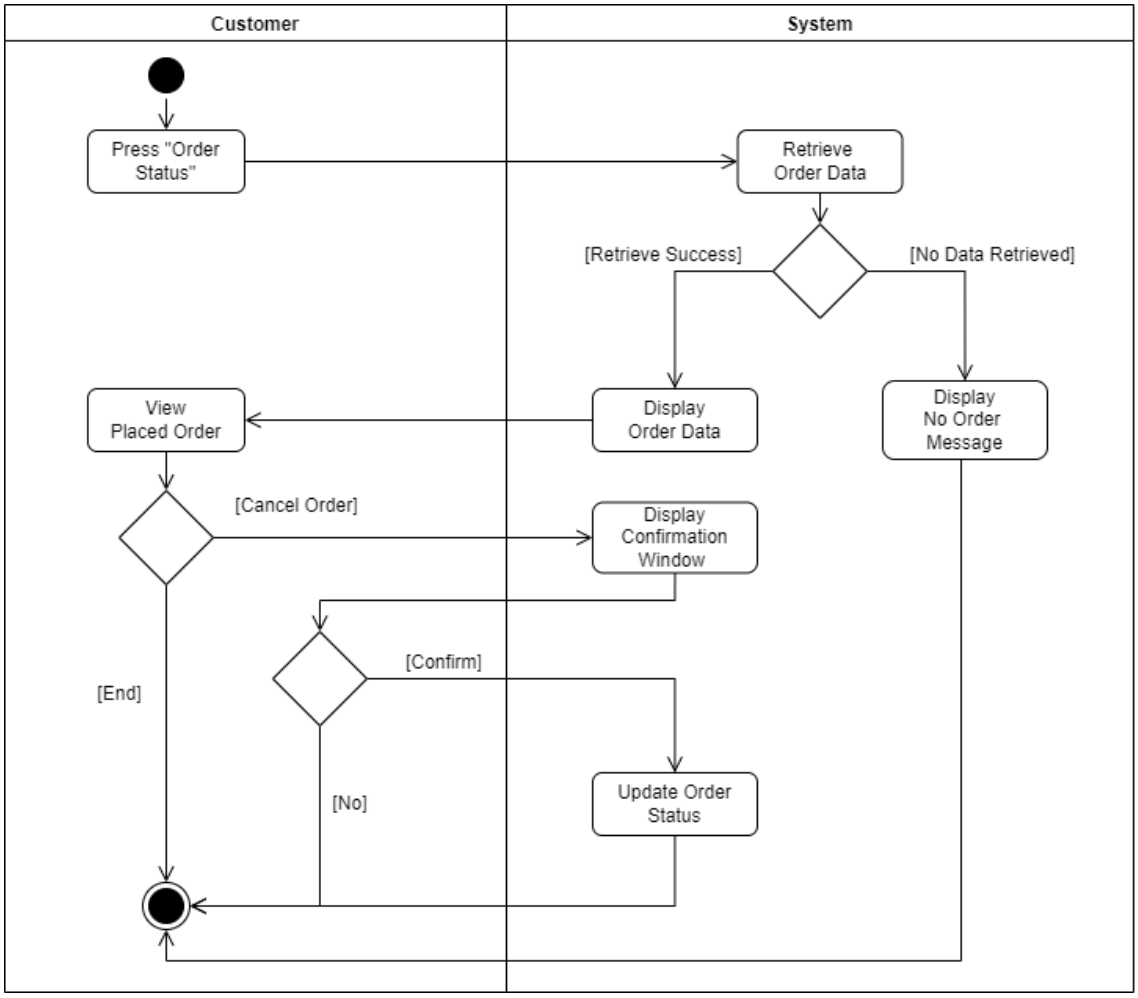

Figure 3.23 Activity Diagram of View Order Status Module

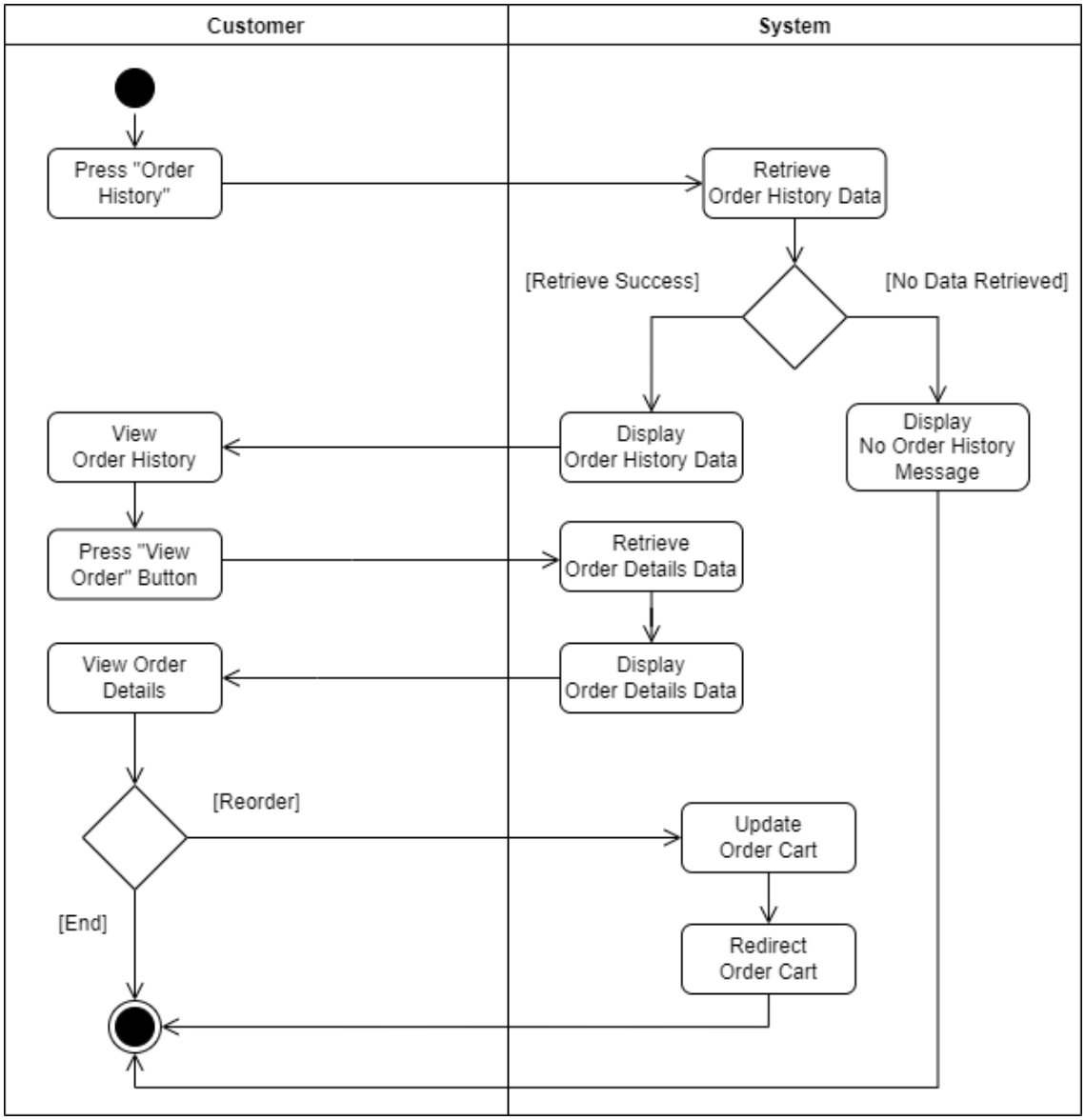

Figure 3.24 Activity Diagram of View Order History Module

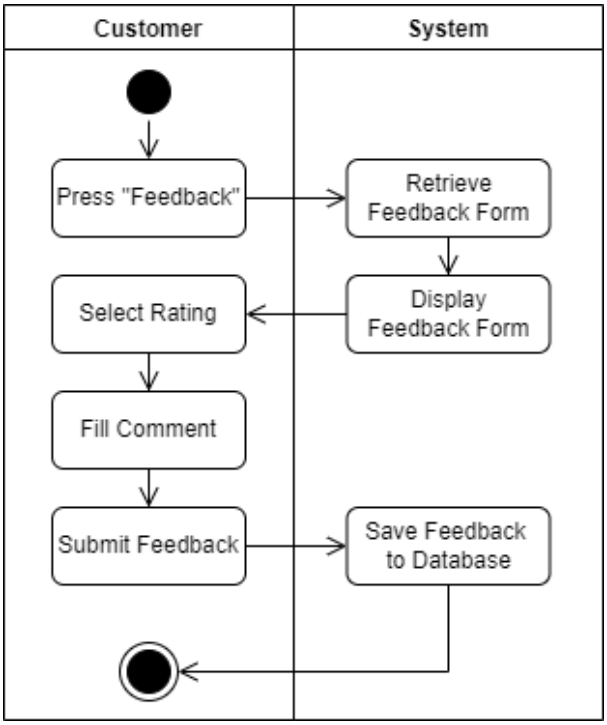

Figure 3.25 Activity Diagram of Provide Feedback Module

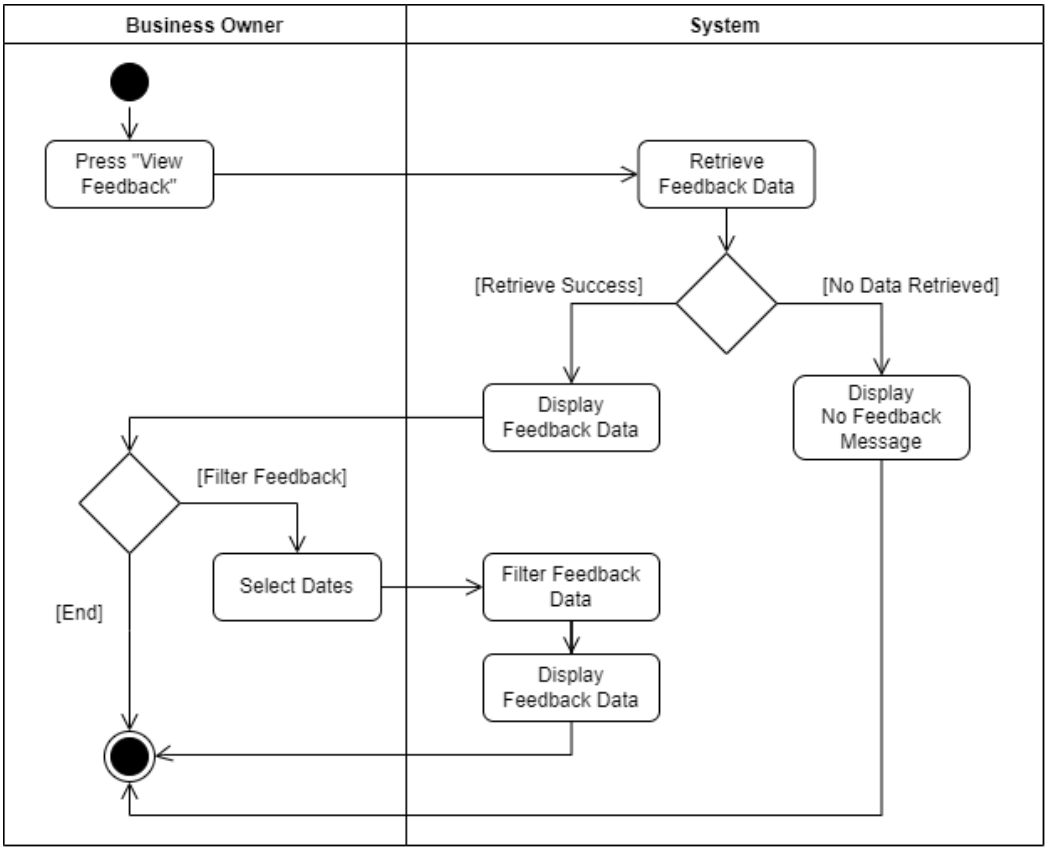

Figure 3.26 Activity Diagram of View Feedback Module

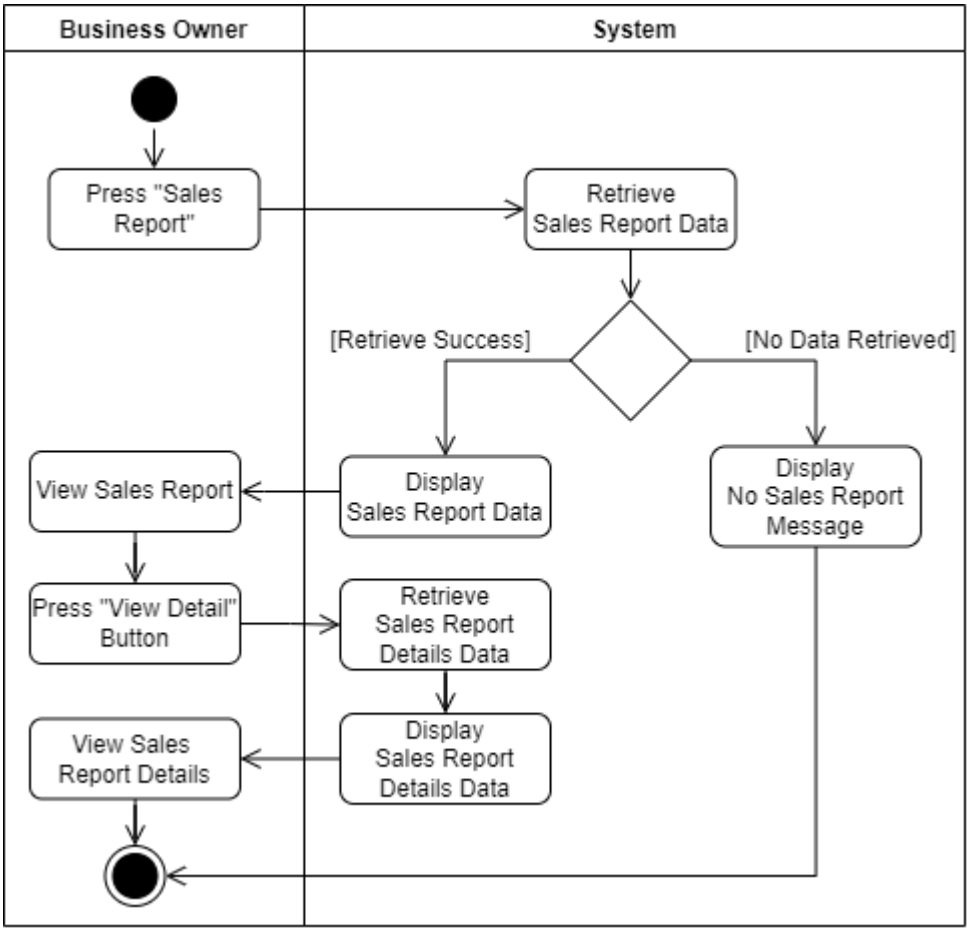

Figure 3.27 Activity Diagram of View Sales Report Module

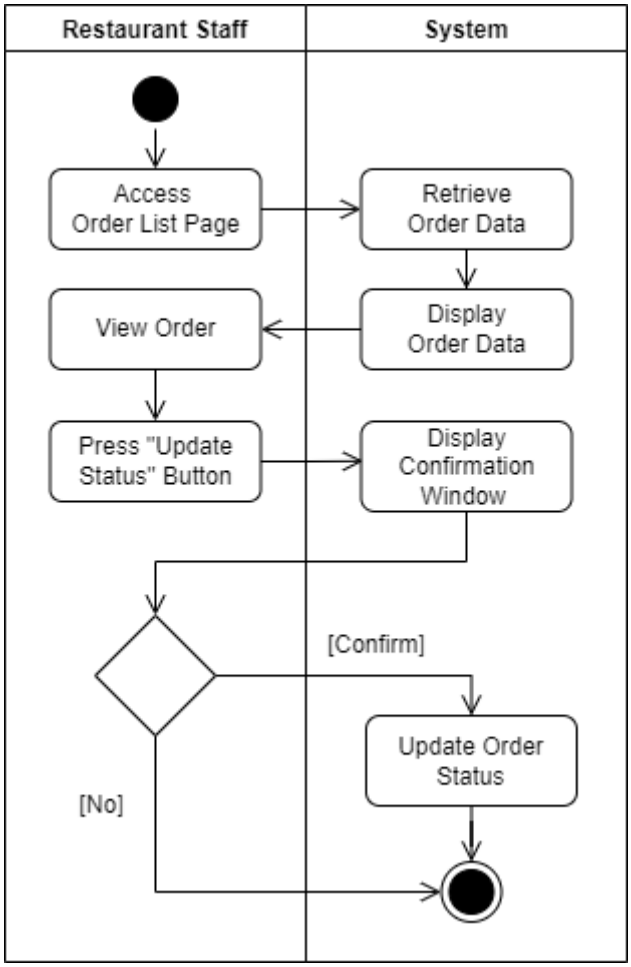

Figure 3.28 Activity Diagram of Update Order Status Module

3.6 Data Design

3.6.1 Entity Relationship Diagram (ERD)

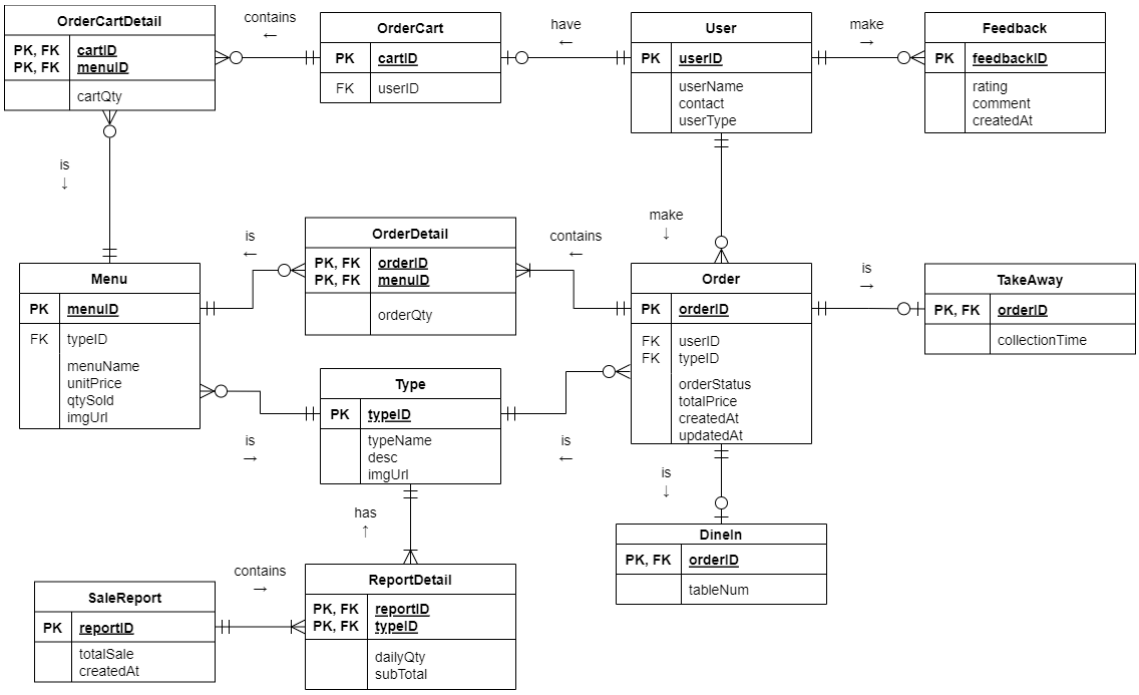

Figure 3.29 Entity Relation Diagram of Proposed System

Figure 3.29 shows the attributes of the entities in MySQL database. However, the database used for the proposed system is Firebase database which is a NoSQL database. There might be some entities combined in Firebase database, but the attributes used are still the same. Figures below shows how the data is stored in the collection in Firebase database and the table depicts the data dictionary of the entities in Figure 3.29 that involved in the collection in Firebase database.

3.6.2 Data Dictionary of Proposed System


```
[
    {
        "__id__": "UCtabgOJv0P8yQ6IMha2IxADkNX2",
        "userName": "Chong Kai Jie",
        "contact": "0167789136",
        "userType": "Owner"
    },
    {
        "__id__": "YawfIM1trmcyPIXDrTiZmvG7xcS2",
        "userName": "Wong See Hua",
        "contact": "0175439665",
        "userType": "Kitchen Staff"
    },
    {
        "__id__": "hJaJCd8x0zPXvSjNqmxhoy2qrHf1",
        "userName": "Chan Kin Keong",
        "contact": "0163289462",
        "userType": "Waiter"
    },
    {
        "__id__": "eMxDha1An3SxUVxSjQcQaG0nvRs3",
        "userName": "Cheong Kai Wei",
        "contact": "0124834452",
        "userType": "Cashier"
    },
    {
        "__id__": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "userName": "Lee Yong Jian",
        "contact": "0112368846",
        "userType": "Customer"
    }
]
```


Figure 3.30 User Collection in JSON

Table 3.12 Data Dictionary of User Entity

Attribute Name	Attribute Type	Description	Constraints
userID	String	ID of user.	PK
userName	String	Name of user.	
contact	String	User contact number.	
userType	String	Type of user	

```
[
    {
        "__id__": "YC1MWy7qFPeehB0aNlry",
        "typeID": 1,
        "typeName": "Dine In",
        "desc": "order"
    },
    {
        "__id__": "hP1Kw6OvN8kaslJ5LDQp",
        "typeID": 2,
        "typeName": "Take Away",
        "desc": "order"
    },
    {
        "__id__": "BBPYiHP73PMYZ1hqIehp",
        "typeID": 3,
        "typeName": "Burger",
        "desc": "menu",
        "imgUrl": "burger/00.jpg"
    },
    {
        "__id__": "VGKAyYfNJaFmGYzHTlem",
        "typeID": 4,
        "typeName": "Noodle",
        "desc": "menu",
        "imgUrl": "noodle/00.jpg"
    },
    {
        "__id__": "dDpABAuBkvncJKiUDBJf",
        "typeID": 5,
        "typeName": "Rice",
        "desc": "menu",
        "imgUrl": "rice/00.jpg"
    },
    {
        "__id__": "9TiK79hACTTl7crFuTIv",
        "typeID": 6,
        "typeName": "Soup",
        "desc": "menu",
        "imgUrl": "soup/00.jpg"
    },
    {
        "__id__": "RAvQbmzQX7pnSETvTbdn",
        "typeID": 7,
        "typeName": "Beverage",
        "desc": "menu",
        "imgUrl": "beverage/00.jpg"
    }
]
```


Figure 3.31 Type Collection in JSON

Table 3.13 Data Dictionary of Type Entity

Attribute Name	Attribute Type	Description	Constraints
typeID	Number	ID of type.	PK
typeName	String	Name of the type.	
desc	String	Description of the type.	
imgUrl	String	Path of image of the type located.	


```
[
    {
        "__id__": "ItIOhavD0TCArLz5wNc5",
        "feedbackID": 1,
        "rating": 3.5,
        "comment": "Food are delicious but waiting time are too long.",
        "createdAt": "__Timestamp__2022-03-13T07:07:47.429Z"
    },
    {
        "__id__": "XNr9ITaxRTa1eglq6aUv",
        "feedbackID": 1,
        "rating": 3.5,
        "comment": "Staff are kind and friendly. Worth of money.",
        "createdAt": "__Timestamp__2022-03-13T07:07:47.429Z"
    }
]
```


Figure 3.32 Feedback Collection in JSON

Table 3.14 Data Dictionary of Feedback Entity

Attribute Name	Attribute Type	Description	Constraints
feedbackID	Number	ID of user feedback.	PK
rating	Number	User rating on their ordering experience.	
comment	String	User comment on their ordering experience.	
createdAt	Timestamp	Date and time of feedback created.	

```
[
    {
        "__id__": "iHpbFHmKHCTPIIB5ZHRf",
        "menuID": 1,
        "typeID": 3,
        "menuName": "Cheese Burger",
        "unitPrice": 12.5,
        "qtySold": 22,
        "imgUrl": "burger/01.jpg"
    },
    {
        "__id__": "DXD63djMlQjrHPqMzuJE",
        "menuID": 2,
        "typeID": 3,
        "menuName": "Beef Burger",
        "unitPrice": 9.3,
        "qtySold": 7,
        "imgUrl": "burger/02.jpg"
    },
    {
        "__id__": "yvhhjyZp28rEqtVFNflM",
        "menuID": 3,
        "typeID": 4,
        "menuName": "Mee Goreng",
        "unitPrice": 5.6,
        "qtySold": 16,
        "imgUrl": "noodle/01.jpg"
    }
]
```

Figure 3.33 Menu Collection in JSON

Table 3.15 Data Dictionary of Menu Entity

Attribute Name	Attribute Type	Description	Constraints
menuID	Number	ID of menu.	PK
typeID	Number	ID of type of the menu.	FK
menuName	String	Name of menu.	
unitPrice	Number	Unit price of menu.	
qtySold	Number	Quantity of menu sold.	
imgUrl	String	Path of menu image located.	

```
[
    {
        "__id__": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "menuID": [
            2,
            4
        ],
        "cartQty": [
            1,
            3
        ]
    }
]
```

Figure 3.34 OrderCart Collection in JSON

Table 3.16 Data Dictionary of OrderCart Entity

Attribute Name	Attribute Type	Description	Constraints
cartID	String	ID of user order cart.	PK
userID	String	ID of user.	FK

Table 3.17 Data Dictionary of OrderCartDetail Entity

Attribute Name	Attribute Type	Description	Constraints
cartID	String	ID of user order cart.	PK, FK
menuID	Array (Number)	ID of menu.	PK, FK
cartQty	Array (Number)	Quantity of menu added.	

```
[
    {
        "__id__": "MqHdIKZc9uN89SYqbOI7",
        "orderID": 2,
        "userID": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "typeID": 1,
        "orderStatus": "On Queue",
        "totalPrice": 36.36,
        "createdAt": "__Timestamp__2022-03-13T07:13:59.133Z",
        "updatedAt": "__Timestamp__2022-03-13T07:22:38.909Z",
        "menuID": [
            3,
            7,
            10,
            1
        ],
        "orderQty": [
            1,
            1,
            1,
            1
        ],
        "tableNum": 5
    },
    {
        "__id__": "ibBOaoZWuHLxNSfmz9rb",
        "orderID": 1,
        "userID": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "typeID": 2,
        "orderStatus": "Cancelled",
        "totalPrice": 49.61,
        "createdAt": "__Timestamp__2022-03-13T07:12:51.714Z",
        "updatedAt": "__Timestamp__2022-03-13T07:13:40.412Z",
        "menuID": [
            3,
            7,
            10,
            1
        ],

        "orderQty": [
            1,
            1,
            1,
            2
        ],
        "collectionTime": "__Timestamp__2022-03-13T07:25:48.000Z"
    }
]
```


Figure 3.35 Order Collection in JSON

Table 3.18 Data Dictionary of Order Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK
userID	String	ID of user.	FK
typeID	Number	ID of type of the order.	FK
orderStatus	String	Status of user order.	
totalPrice	Number	Total price of user order.	
createdAt	Timestamp	Date and time of order created.	
updatedAt	Timestamp	Date and time of order updated.	

Table 3.19 Data Dictionary of TakeAway Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
collectionTime	Timestamp	Collection time of an order.	

Table 3.20 Data Dictionary of DineIn Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
tableID	Number	ID of the table for an order.	

Table 3.21 Data Dictionary of OrderDetail Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
menuID	Array (Number)	ID of menu ordered.	PK, FK
orderQty	Array (Number)	Quantity of menu ordered.	


```
[
    {
        "__id__": "1M4S5xRb9FqHSflBZNei",
        "reportID": 1,
        "totalSale": 72.72,
        "createdAt": "__Timestamp__2022-03-13T07:21:13.344Z"
    }
]
```


Figure 3.36 SaleReport Collection in JSON

Table 3.22 Data Dictionary of SaleReport Entity

Attribute Name	Attribute Type	Description	Constraints
reportID	Number	ID of sale report.	PK
totalSale	Number	Total sale of the report.	
createdAt	Timestamp	Date and time of report created.	


```
[
    {
        "__id__": "EKKBFnHCJo8BA1XMPra9",
        "reportID": 1,
        "dailyQuantity": [
            2,
            0,
            2,
            2,
            0,
            2,
            2
        ],
        "subTotal": [
            72.72,
            0,
            25,
            11.2,
            0,
            25.2,
            7.2
        ]
    }
]
```


Figure 3.37 ReportDetail Collection in JSON

Table 3.23 Data Dictionary of ReportDetail Entity

Attribute Name	Attribute Type	Description	Constraints
reportID	Number	ID of report.	PK, FK
typeID	Number	ID of type.	PK, FK
dailyQuantity	Array (Number)	Quantity sold for each type.	
subTotal	Array (Number)	Subtotal of profit for each type.	

3.7 Testing Plan

The testing of the system will be initiated after the proposed system has completely developed. The developer will first test the developed system to ensure it is bug-free and able to work as expected. After that, a survey google form will be distributed to collect feedback from the potential users regarding the proposed system. This testing will involve the customers, business owners, kitchen staff, waiters, and cashiers. The potential users will be requested to install the proposed system on their mobile device and go through the system at least once before they respond to the survey. This testing is to ensure the developed system fulfils the expectations of the potential users and is accepted by them.

3.8 Potential Use of Proposed Solution

Despite technical advances, most of the restaurants still use a conventional ordering method. The proposed solution will be able to help these restaurants to simplify and facilitate the ordering process and the management process for the restaurant. Moreover, it also provides a platform for business owners to digitalize their business processes. By implementing the proposed solution, the business owners can reduce the operating expenses of the business as well as the workload of the restaurant staff and enable them to focus on their task. Besides, the proposed solution also improves customers' order experience and satisfaction.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Chapter 4 will discuss about the implementation and development of Order Management System for Restaurant (OMSR). This chapter aims to illustrate the implementation and development process of the application with greater detail. OMSR simplifies and facilitates both the ordering process and the management process. The interaction between customers and staff are minimized and the business owner can easily check on the business sales and customers' feedback to improve the service. Testing was done on the OMSR to uncover any potential errors and resolve them as soon as possible.

4.2 Development Tools

Table 4.1 shows the software used in completing the project from the beginning.

Table 4.1 Software Used for Project Development

No	Software	Purpose
1	Android Studio	The software used to develop the order management system for restaurant.
2	Firebase	The database used by the system.
3	Flutter	The framework used to develop the order management system for restaurant.
4	draw.io	The software used to create context diagram, use case diagram, activity diagram and entity relationship diagram.
5	Figma	The software used to create the prototype (storyboard).

4.3 Implementation Process

In this section, the process involved in implementing and developing OMSR is recorded. There are three elements of project implementation and development is discussed which is the Integrated Development Environment (IDE), Software Development Kit (SDK) and database.

4.3.1 Implementation of Software Development Kit (SDK)

The SDK used in this project is Flutter. Flutter is a SDK that supports both developers and designers in developing mobile applications for iOS and Android. It also falls under the category "Cross-Platform Mobile Development" due to its characteristics.

Flutter must be installed on the computer before the development process start. The Flutter installation guide can be found on official Flutter documentation website at https://docs.flutter.dev/get-started/install. Figure 4.1 shows the Flutter installation guide page for different Operating System (OS).

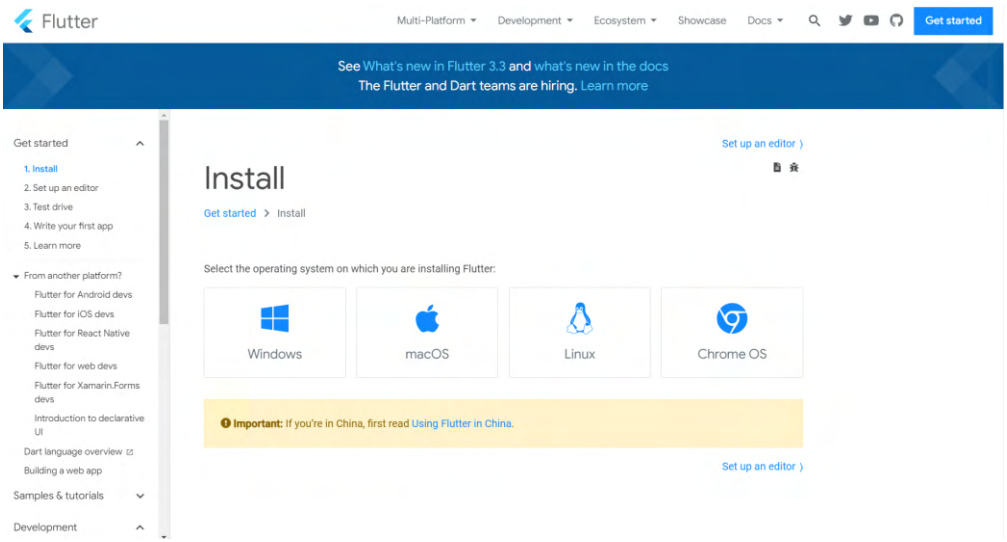

Figure 4.1 Flutter Installation Guide on Flutter Documentation Website

For Windows OS, first we need to download the Flutter SDK from the website by clicking the button labelled "flutter_windows_3.3.7-stable.zip" as shown in Figure 4.2.

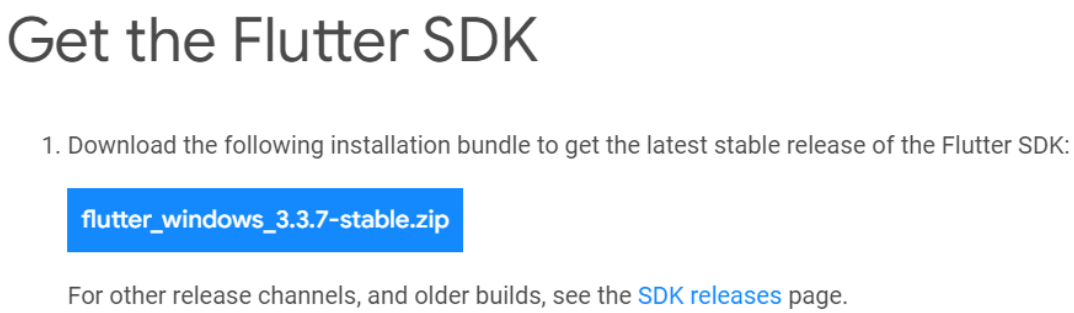

Figure 4.2 Flutter SDK Download

After finish downloading, the zip file needs to be placed the desired installation location and extract. Keep in mind that the path where the Flutter to be installed shall not contains any special characters or spaces and do not install it in the directory which requires elevated privileges such as “C:\Program Files\”. Besides, instead of using the installation bundle, we can also use the command “git clone https://github.com/flutter/flutter.git -b stable” in Command Prompt to get the source code from Flutter repository on GitHub.

Next, we need to update the environment variable to ensure the Flutter commands can run in the regular Windows console. To do so, first, we need to enter ‘env’ in Windows search box as shown in Figure 4.3. Click the “Edit environment variables for your account and a window like Figure 4.4 will pop-up.

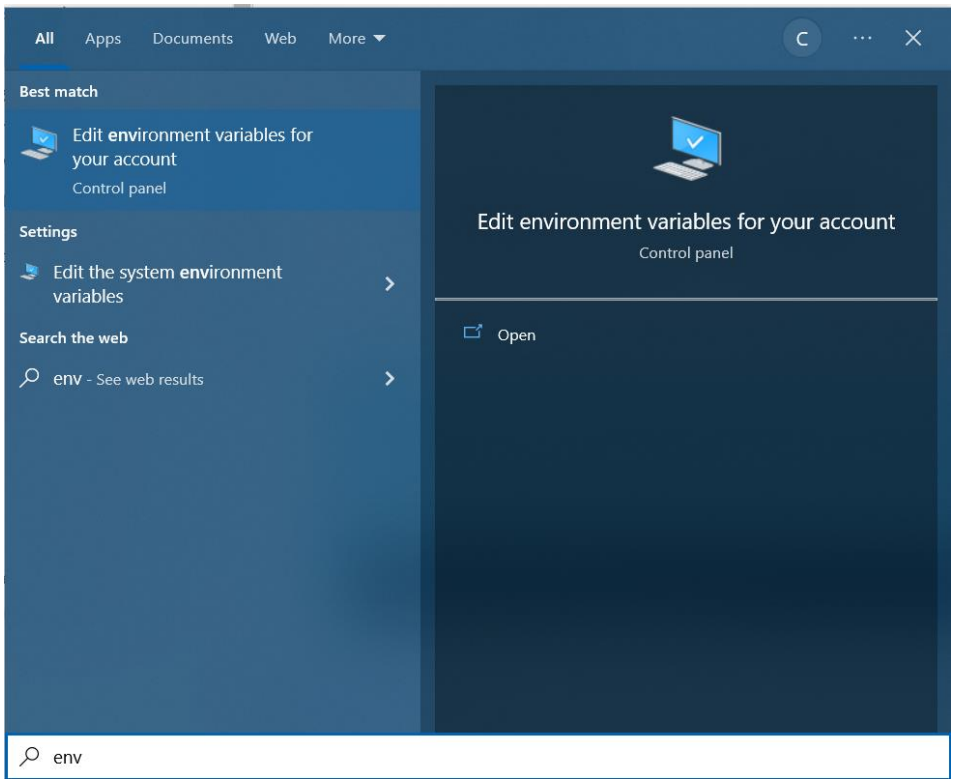

Figure 4.3 Edit Environment Variable

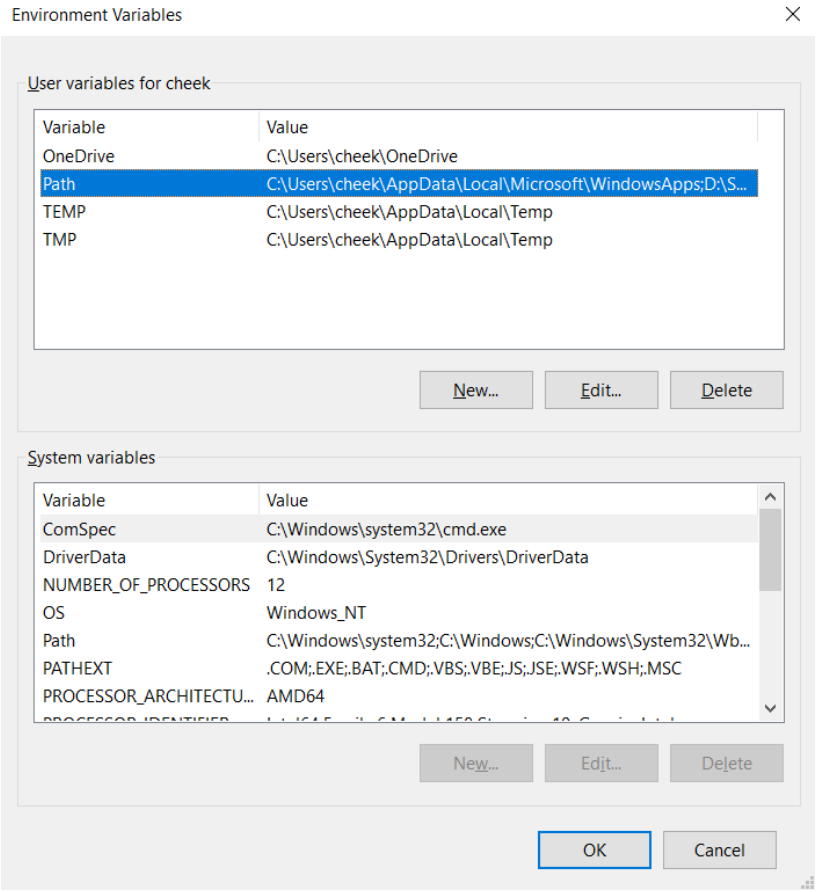

Figure 4.4 Environment Variables Window

We need to double click the entry named “Path” under user variables and click the “New” button then paste the full path to flutter\bin as shown in Figure 4.5. If there is no entry named “Path”, we need to click “New…” button in Figure 4.4 then use “Path” as the variable name and the full path to flutter\bin as the variable value.

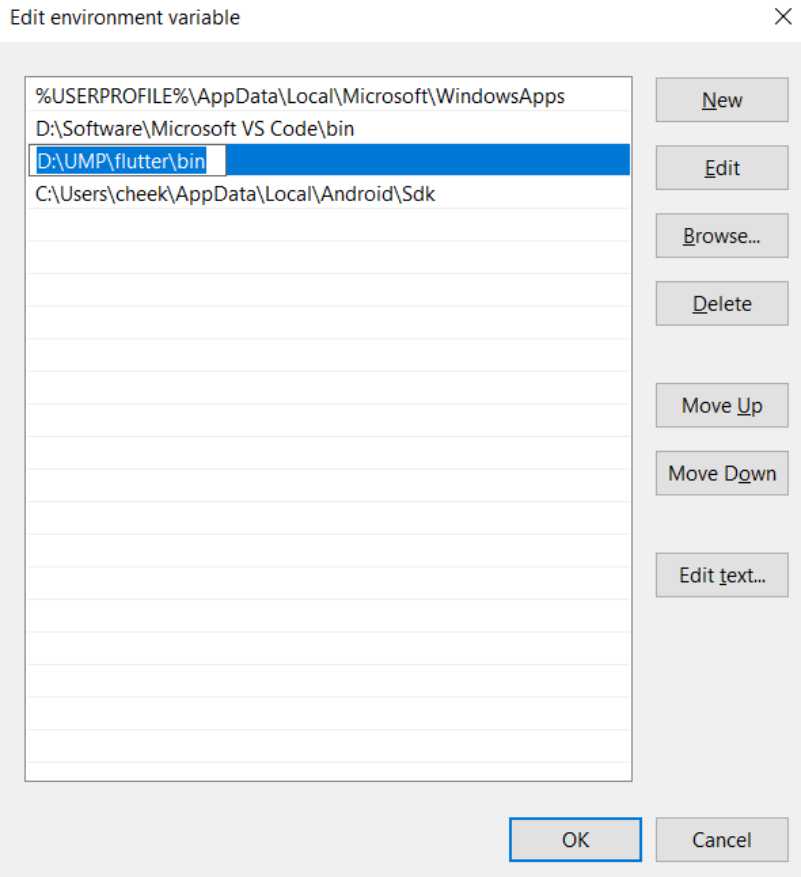

Figure 4.5 Add New Environment Variable

Now we can run flutter doctor in Command Prompt by using the command “flutter doctor” to check if there any platform dependencies that need to be set up. The result of running the command should be similar to Figure 4.6 where Flutter, Android toolchain and Android Studio must show a green tick at the front to ensure the development can be performed smoothly. If there is any red cross exist, it should be resolved before the development process start. However, Visual Studio is not used for this project, hence we can ignore the red cross.

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cheek>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.0.5, on Microsoft Windows [Version 10.0.19044.2130], locale en-US)
[√] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2021.2)
[√] Connected device (3 available)
[√] HTTP Host Availability

! Doctor found issues in 1 category.
```

Figure 4.6 Output of Running Flutter Doctor Command

4.3.2 Implementation of Integrated Development Environment (IDE)

IDE are the software that facilitates the development of other software. It is designed to integrates all programming activities into a single application and provides a central interface with all the tools a developer requires, such as code editor, compiler, debugger and build automation tools.

The IDE used in this project is Android Studio. Android Studio can be downloaded from the official website of Android Studio at https://developer.android.com/studio as shown in Figure 4.7.

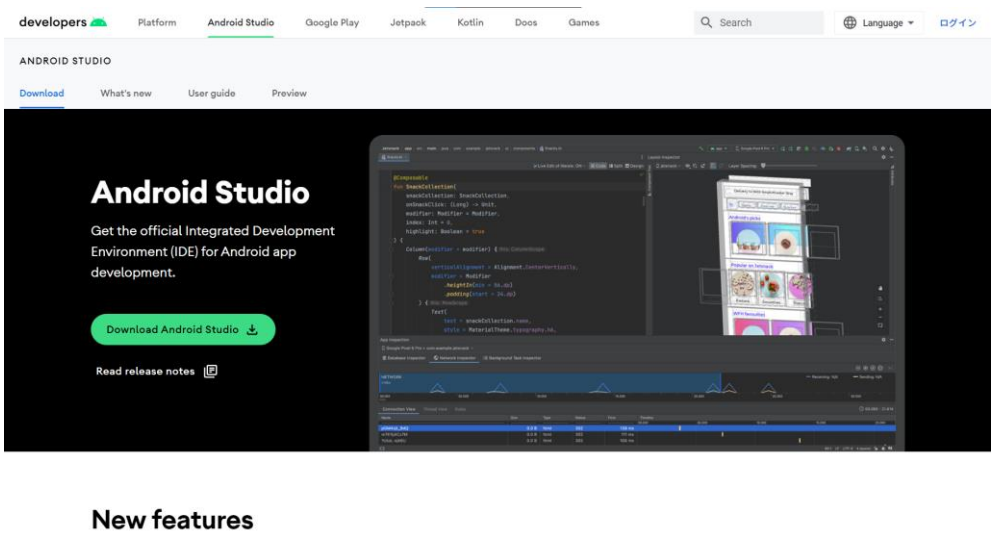

Figure 4.7 Android Studio Official Website

After installing the Android Studio, we need to install two plugins before we can start developing a Flutter project in Android Studio. First, open Android Studio and go to the “Plugins” tab on the left menu as shown in Figure 4.8. Next, enter “Flutter” and “Dart” at the search box respectively to search for the relevant plugins and install them

to Android Studio. After the installation is completed, we need to restart Android studio and now we can go to the “Project” tab on the left menu, and it should now have a button called “New Flutter Project” as shown in Figure 4.9.

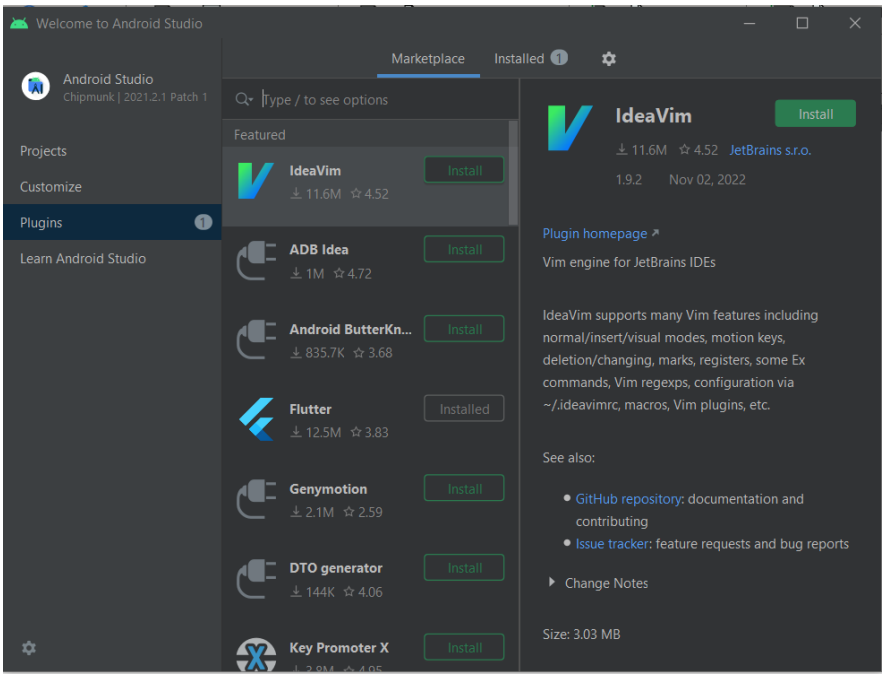

Figure 4.8 Android Studio Dashboard - Plugins Tab

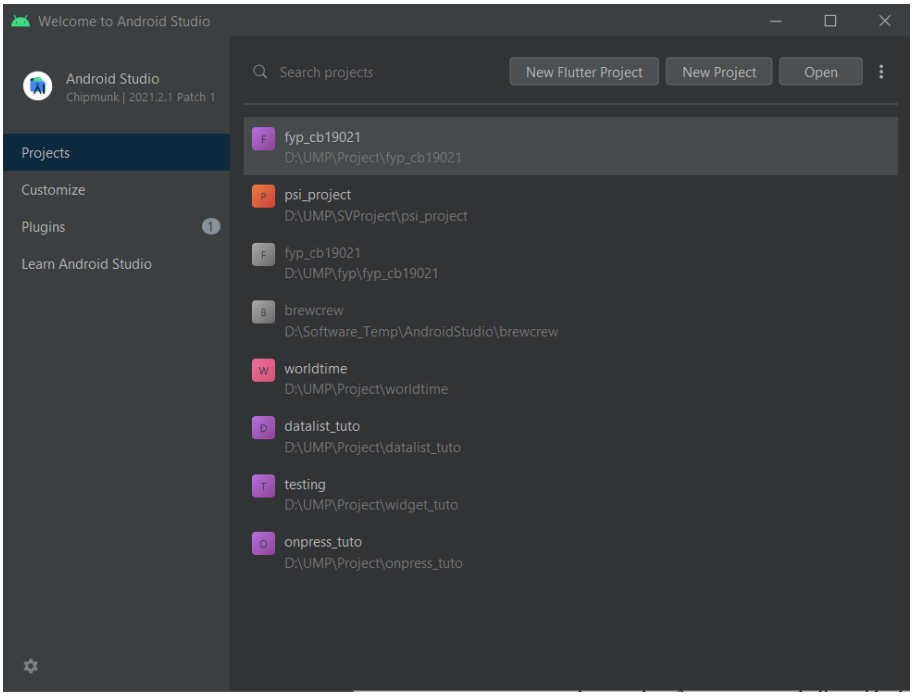

Figure 4.9 Android Studio Dashboard - Projects Tab

By clicking the “New Flutter Project”, it will prompt a new window as shown in Figure 4.10. We need to ensure the Flutter SDK path displayed is the same with the path where Flutter installed then we can click “Next” button. After that, we need to fill in the project information such as project name, project location, the language used for android platform and iOS platform as shown in Figure 4.11. Once it is completed, we can click on “Finish” button and Android Studio will start creating the Flutter project for us.

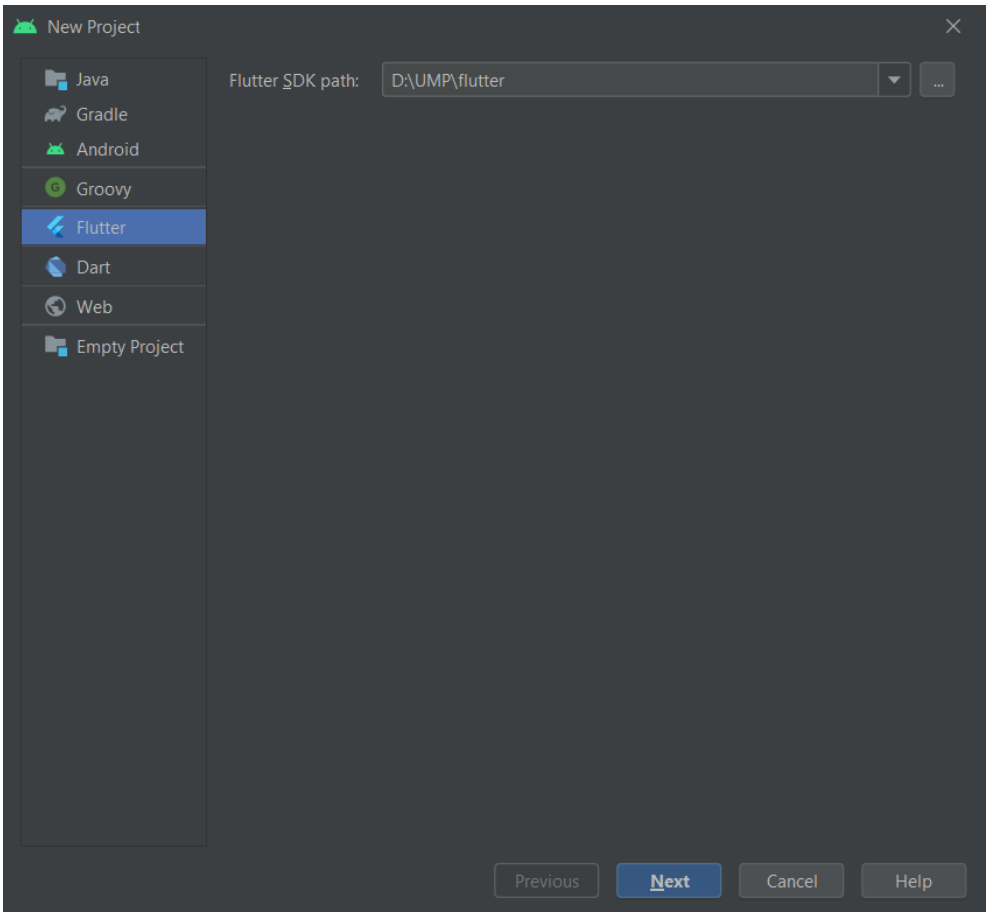

Figure 4.10 Android Studio - New Flutter Project

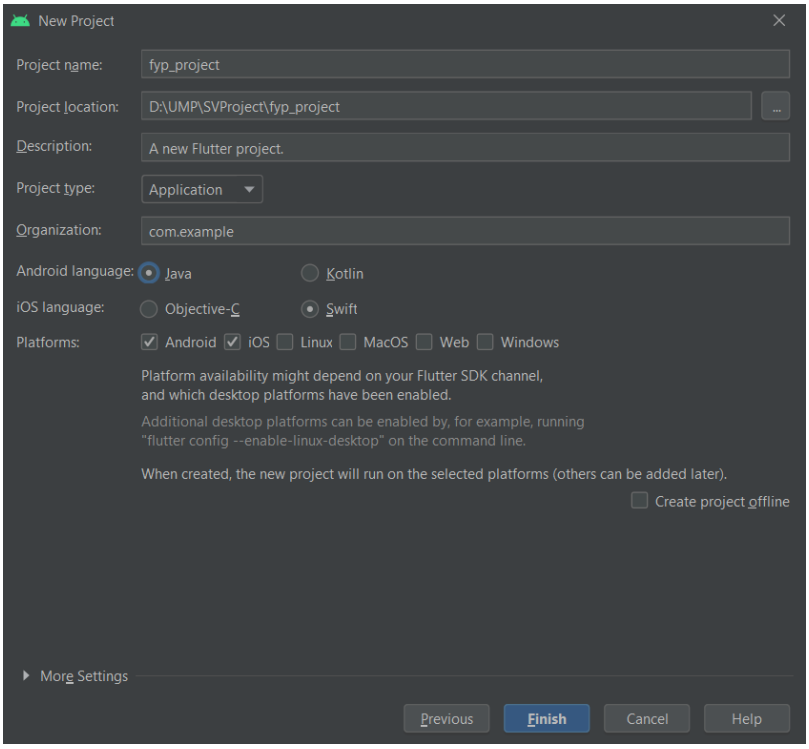

Figure 4.11 Android Studio - New Flutter Project Info

After the project is created, we need to set up the Android emulator such that we can run the application on our laptop for debug purposes. To set up Android emulator, first we need to go “Tools” tab on the top menu and click on “Device Manager” as shown in Figure 4.12.

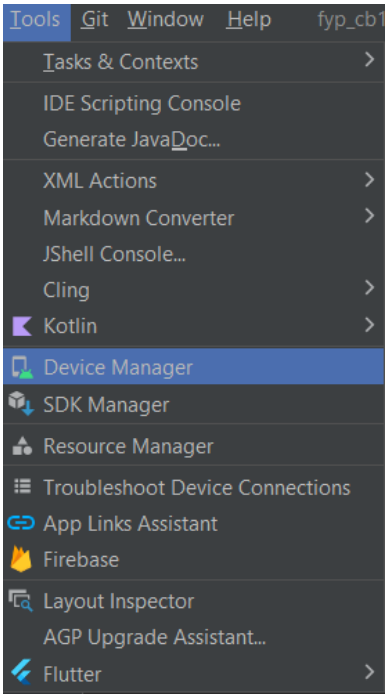

Figure 4.12 Android Tools Menu

After that, a tab like Figure 4.13 will pop-up and we need to click the “Create Device” button. Next, the software will prompt us to select hardware as shown in Figure 4.14. After clicking the “Next” button in Figure 4.14, Figure 4.15 will be displayed. We can select any desired system image and click the “Download” button to download the system image. Once the system image is downloaded, click “Next” button to proceed to the last step of virtual device configuration.

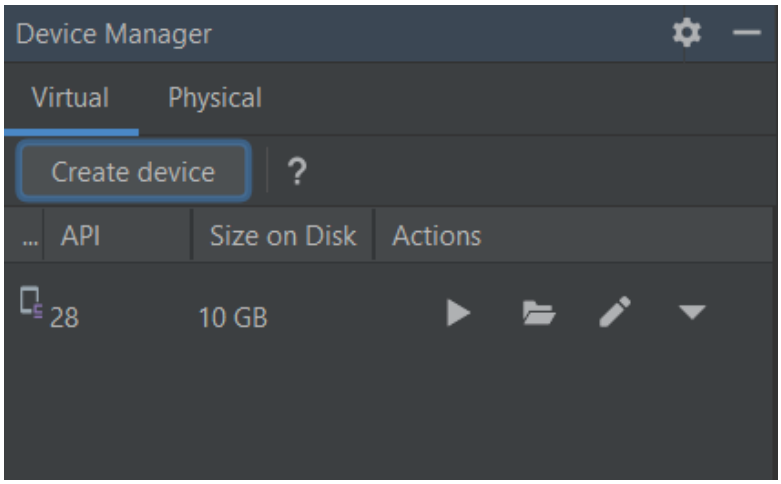

Figure 4.13 Android Studio Device Manager Tab

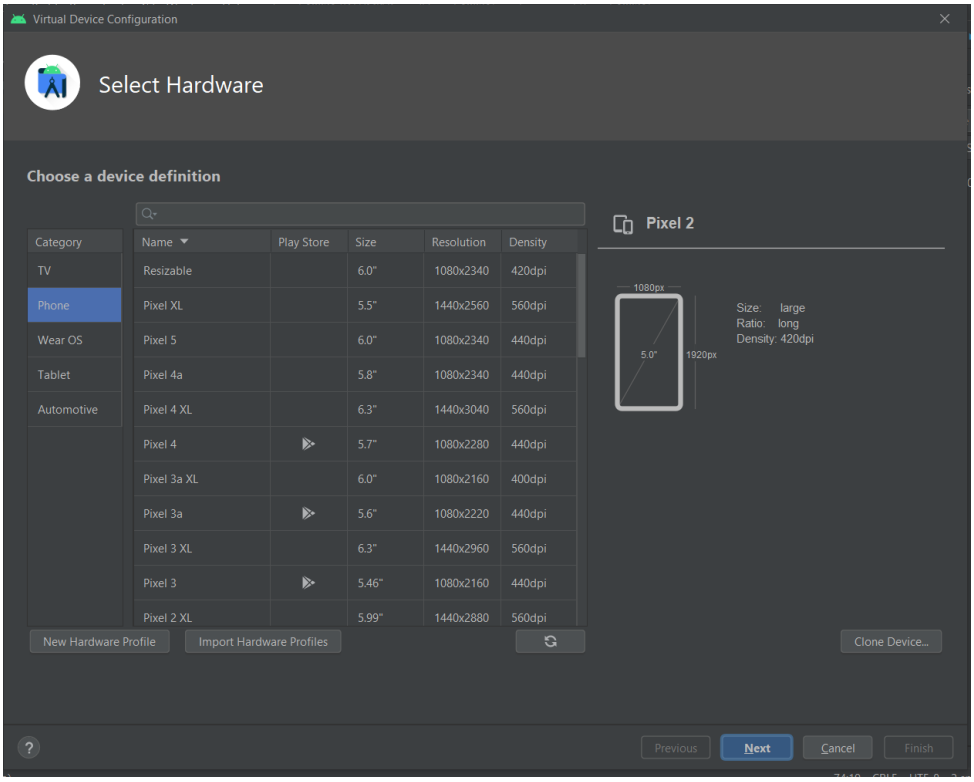

Figure 4.14 Android Studio Select Virtual Device Model

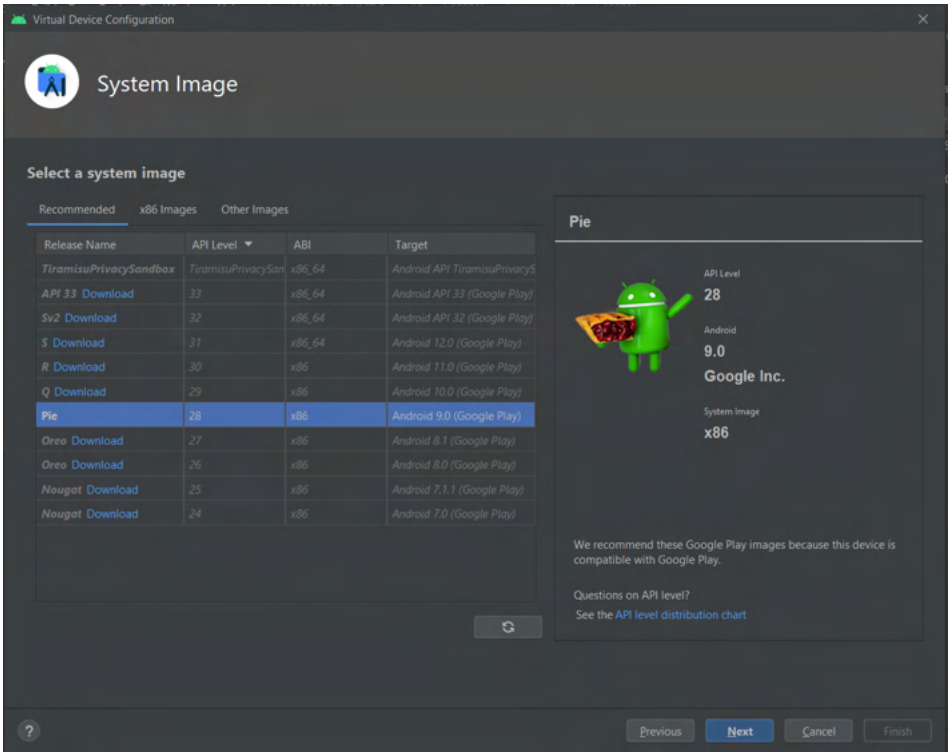

Figure 4.15 Android Studio Select Virtual Device System Image

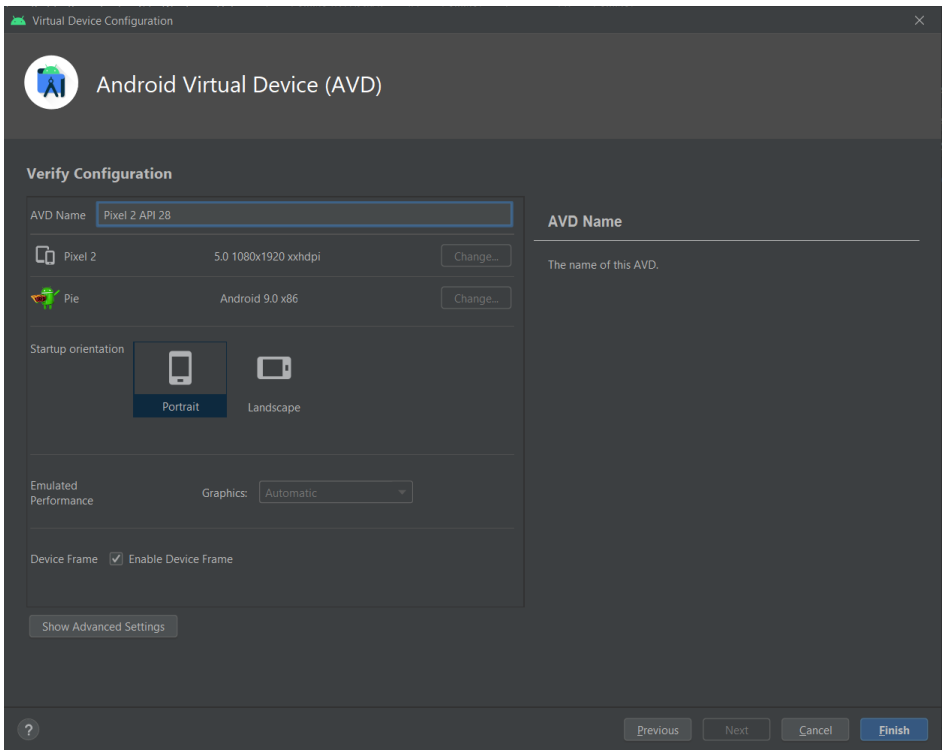

Figure 4.16 Android Studio Verify Virtual Device Configuration

Figure 4.16 shows the last step of virtual device configuration where we can give a name for this virtual device and verify its configurations. Now, the developed Flutter application is ready to run on the emulator.

4.3.3 Implementation of Cloud Firestore Database

Cloud Firestore Database is a flexible and scalable NoSQL cloud database that stores and syncs data from server- and client-side development. To implement Cloud Firestore Database, we need to go Firebase console page at https://console.firebase.google.com and click the “Add Project” button to create a new Firebase project as shown in Figure 4.17. After that, we need to complete the project set up by giving a name for the project and enable or disable some service provided.

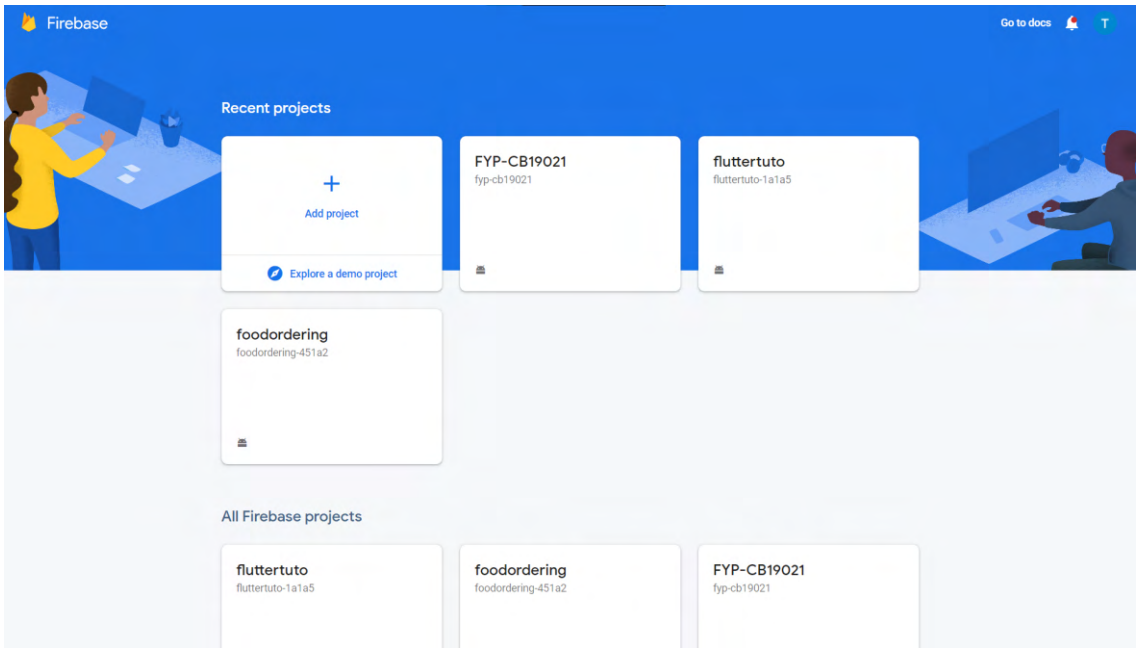

Figure 4.17 Firebase Console Page

After the setup is completed, we will be redirected to the Project Overview page as shown in Figure 4.18. Now we need to press the button with Android icon in Figure 4.18 because OMSR is an Android application.

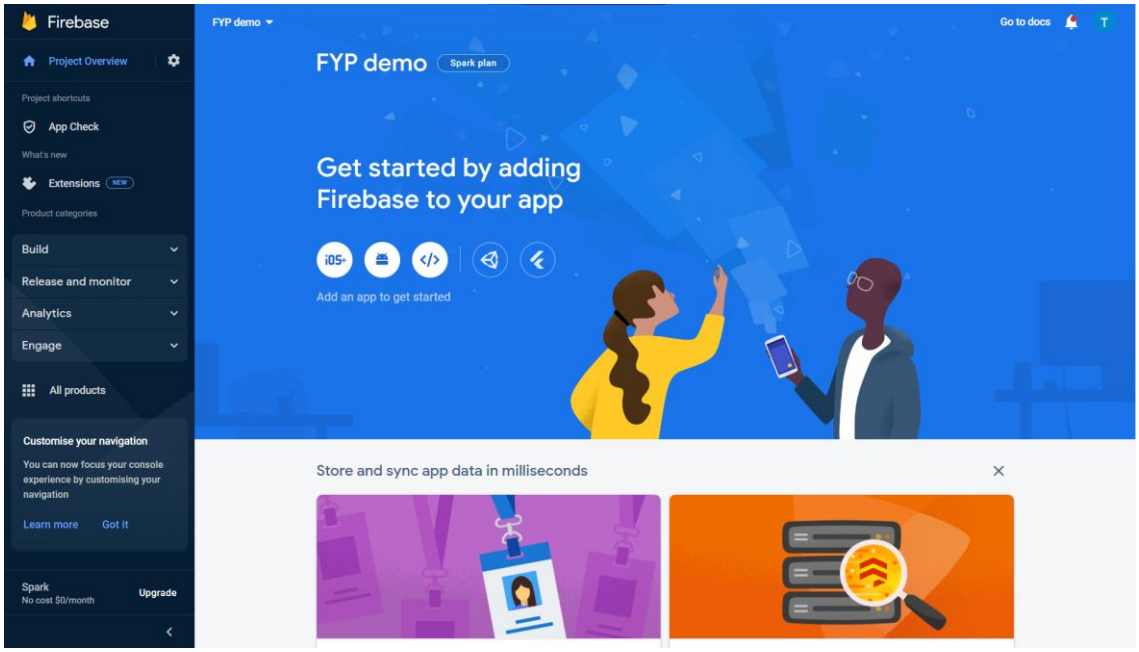

Figure 4.18 Project Overview in Firebase Console

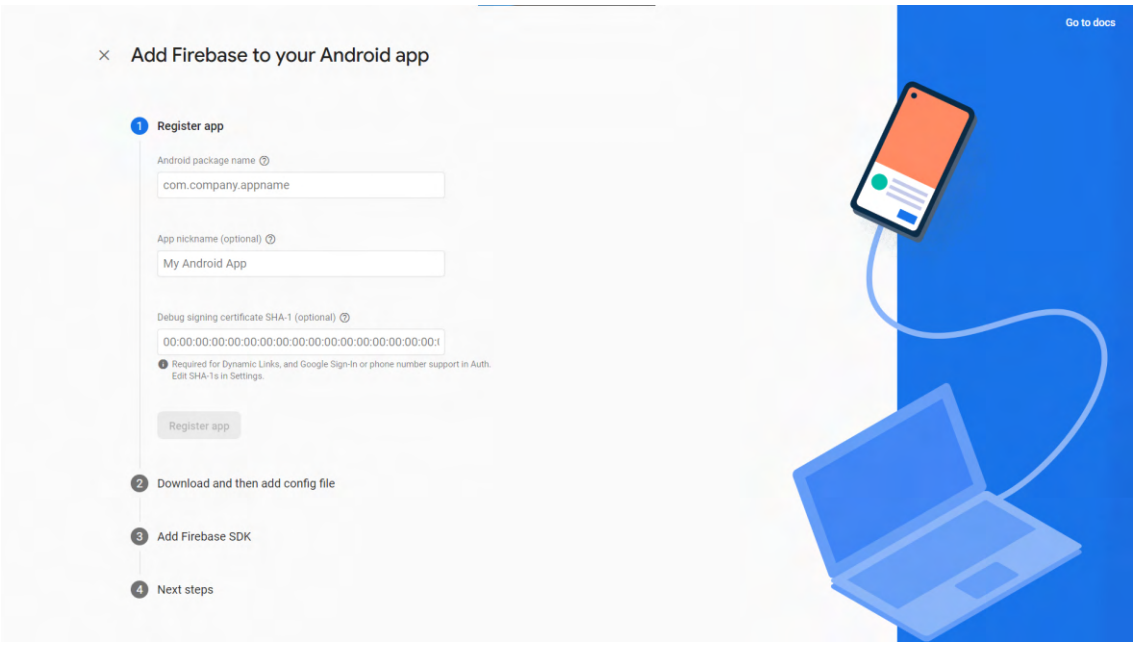

Figure 4.19 Add Firebase to Android App - Step 1

Moving on, we need to fill in the android package name and app nickname then click “Register App” button.

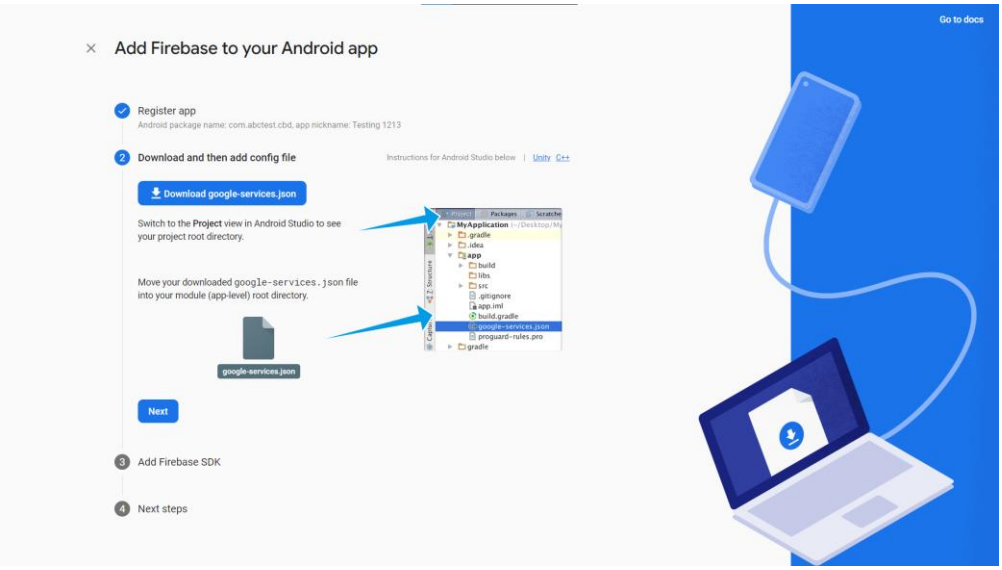

Figure 4.20 Add Firebase to Android App - Step 2

Next, we need to download the config file by pressing the “Download google-services.json” button. Once the download is completed, we need to move the downloaded file into “<project name>\android\app” as shown in Figure 4.7

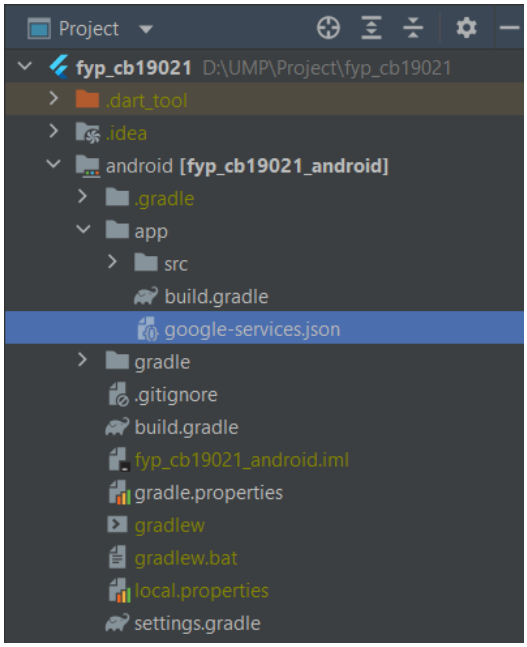

Figure 4.21 Add Firebase to Android App - Step 3

Lastly, we need add Firebase SDK to the project by adding “classpath 'com.google.gms:google-services:4.3.13'” at dependencies section in “<project name>\android\build.gradle” file as shown in Figure 4.22. We also need to add “apply plugin: 'com.google.gms.google-services'” to the “<project name>\android\app\build.gradle” as shown in Figure 4.23.

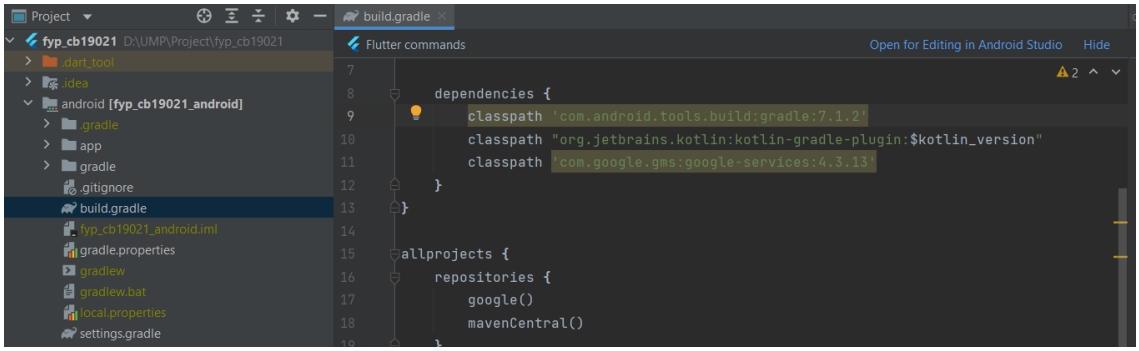

Figure 4.22 Add Firebase to Android App - Step 4

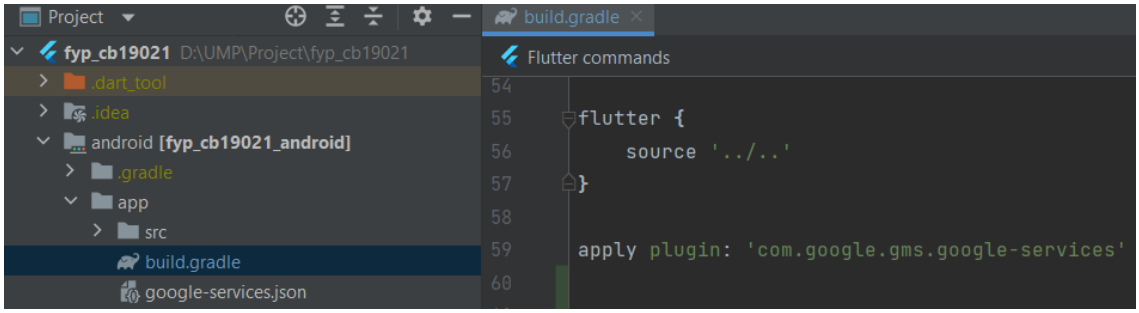

Figure 4.23 Add Firebase to Android App - Step 5

Now the Firestore database is ready to use. In addition, while the development is ongoing, we will need to create some collections and documents in Firestore database as shown in Figure 4.24.

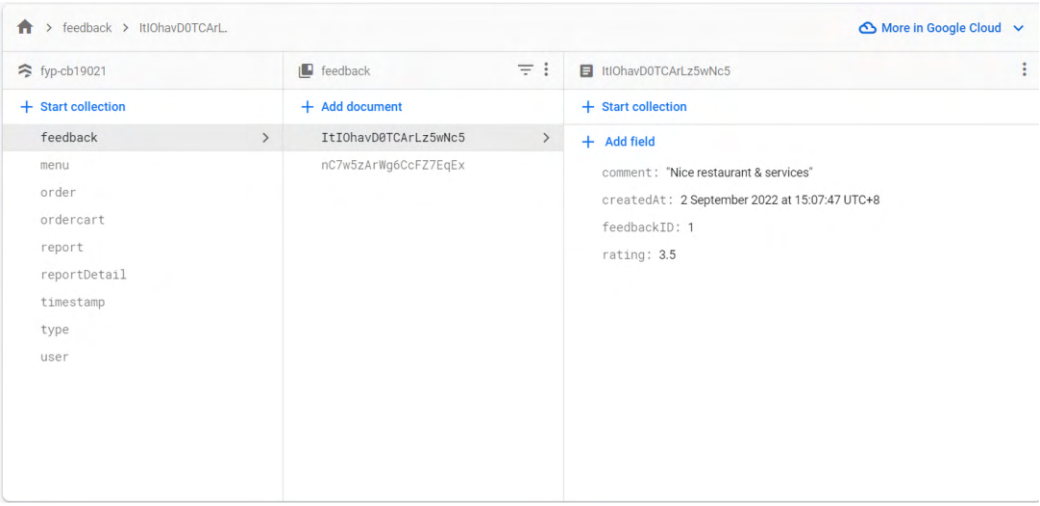

Figure 4.24 Firestore Database Interface

4.4 System User Interface

After the implementation process is completed, development of the proposed application can now be started. This project proposed an ordering application that simplify and facilitate the ordering and management process. Hence, this section will discuss the developed user interface and their functionality.

4.4.1 Login Page

Figure 4.25 shows the login page of OMSR. The users must login to the system with their registered email and password before they can perform any further action. If the users do not have an account, they need to register a new account by pressing the "Register" button to be redirected to registration page. However, the register function is only for customers. The restaurant staff will get their registered account from the business owner.

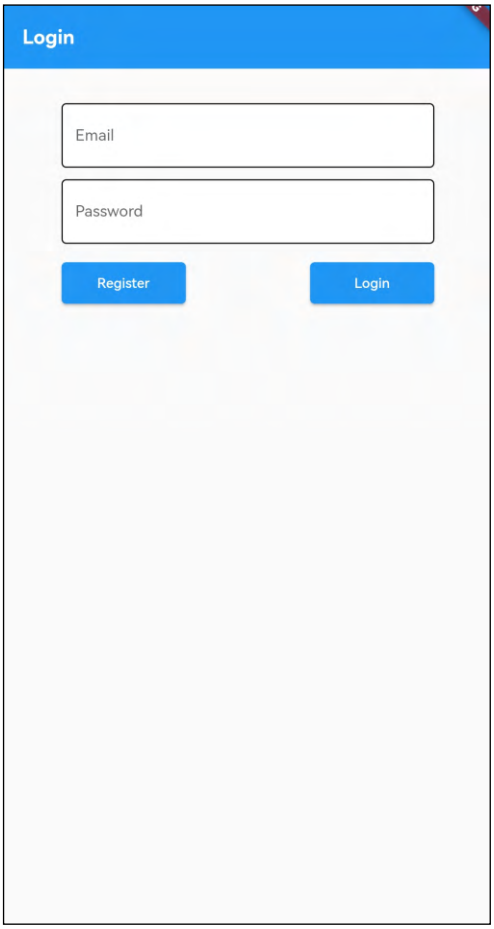

Figure 4.25 OMSR - Login Page

4.4.2 Registration Page

Figure 4.26 illustrates the registration page of OMSR where the customer is required to input their name, contact number, email address and password. After that, the customer can press the “Register” button to register their new account. If the user mistakenly gets into this page, they can return to login page by pressing the “Login” button on right top.

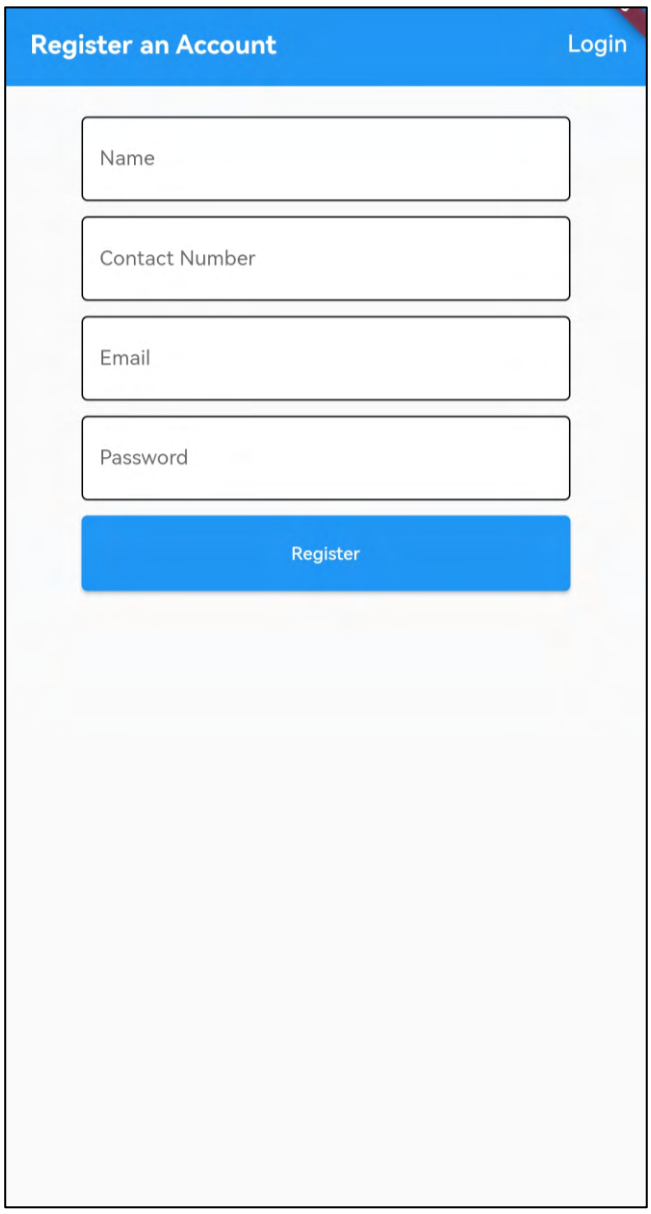

Figure 4.26 OMSR - Registration Page for Customer

4.4.3 Verify Email Page

Figure 4.27 depicts the verify email page of OMSR. Every user of OMSR is required to verify their email address after they register. The verification email will immediately send to the registered email address. The user is required to access the URL provided to verify their email. If the user did not receive any verification email, they can press on the “Resent Email” button to send again the verification email. This page will be displayed to the user until their email is verified. Moreover, if the user wants to logout or terminate the verify email process, they can press the “Cancel” button.

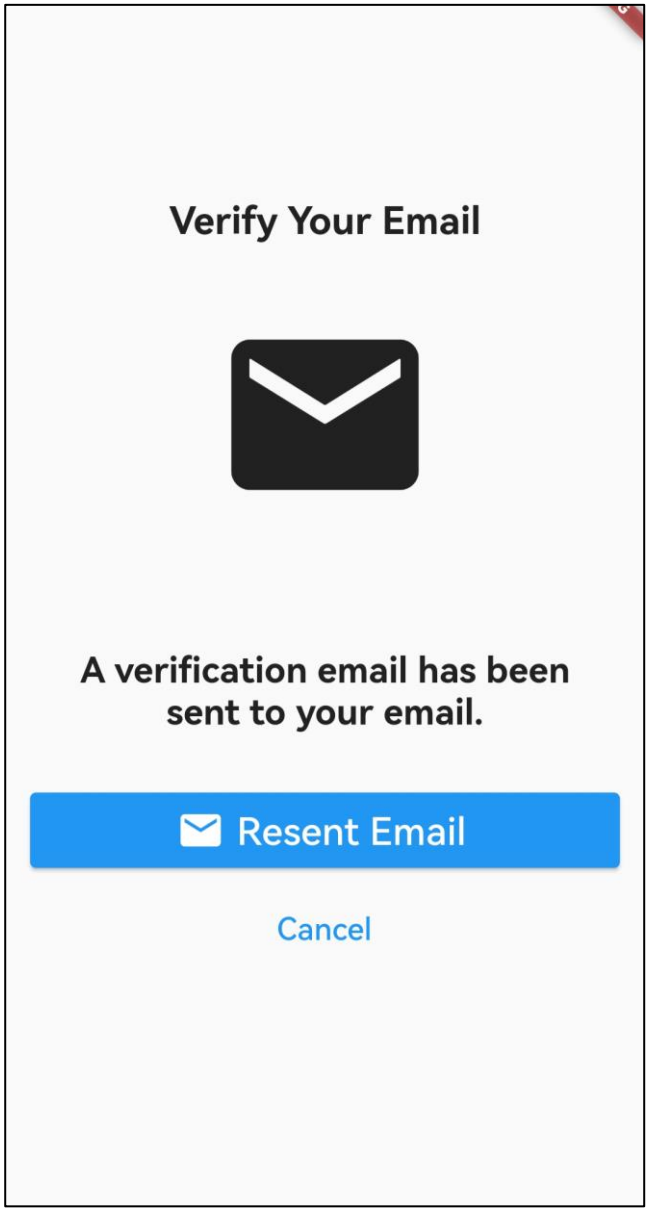

Figure 4.27 OMSR - Verify Email Page

4.4.4 Customer Homepage

Figure 4.28 shows the customer homepage of OMSR. Customer can have access to four modules by pressing the button displayed on this page, which is place order module, view order status module, view order history module and provide feedback module. Moreover, customer can access the four modules by pressing the menu icon on left top and the result is shown in Figure 4.29. Furthermore, they can also access the order cart page as well by pressing the cart icon on right top.

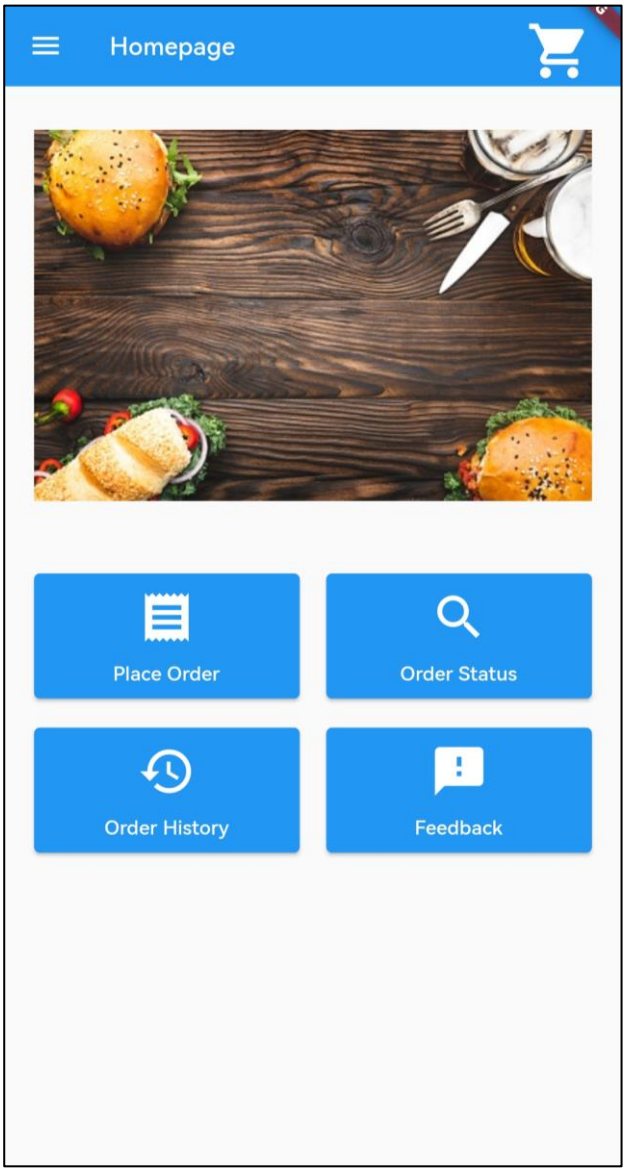

Figure 4.28 OMSR - Customer Homepage

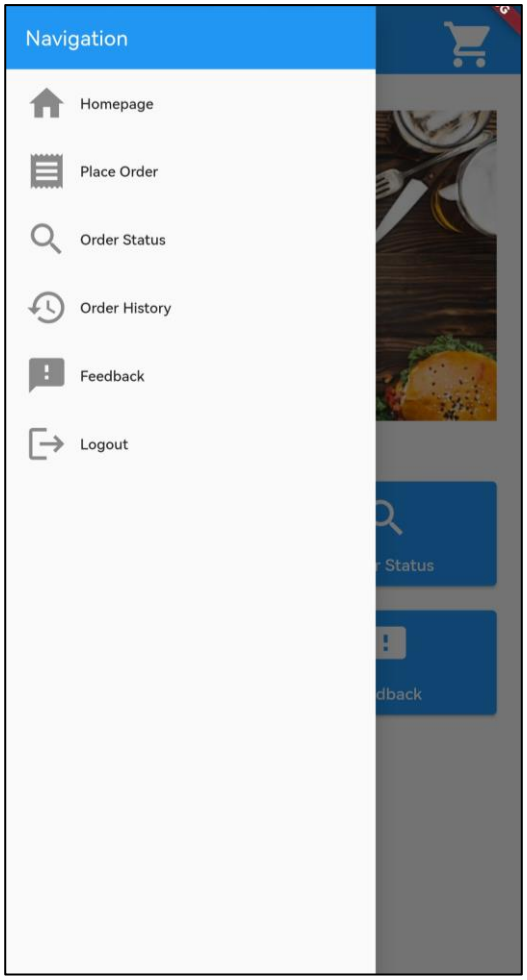

Figure 4.29 OMSR - Customer Navigation Menu

4.4.5 Customer Place Order Module Interfaces

Figure 4.30 illustrates the menu type page that will be displayed to the customer when they want to place order. The customer can press their desired menu type to check the available menu. Figure 4.31 shows the list of burger menus after the “Burger” in Figure 4.30 is pressed. Customer can add their desired menu to order cart by pressing the “Add to Cart” button in Figure 4.31. The cart icon on right top will be updated with a number “1” as shown in Figure 4.32 after the menu is successfully added. This number indicates the number of items in customer order cart. After that, customer can press the cart icon on the right top to access the order cart as shown in Figure 4.33. Customer can view their menus and the payment details. They can also delete the menu by pressing the trash button and increase or decrease the quantity of the menu before they checkout their order cart. Lastly, the customer can press the “Order” button to place their order.

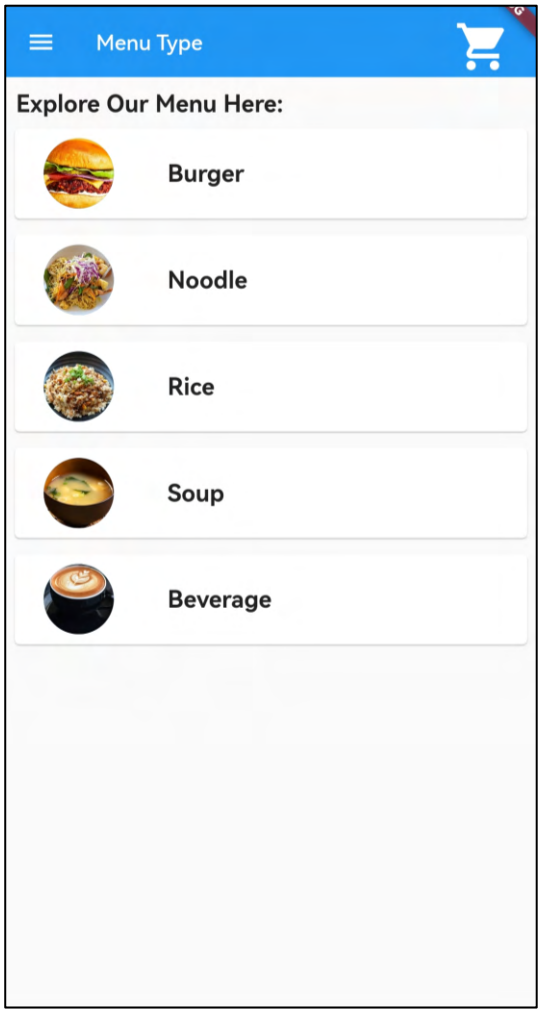

Figure 4.30 OMSR - Menu Type Page

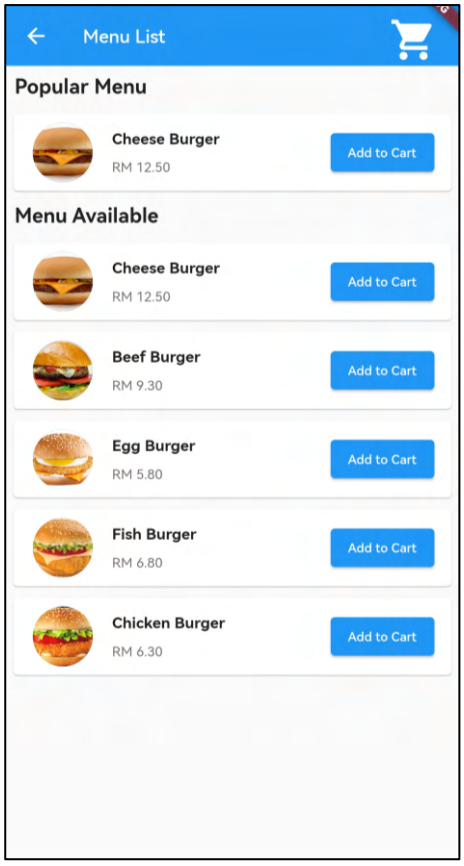

Figure 4.31 OMSR - Menu List Page Before Add to Cart Pressed

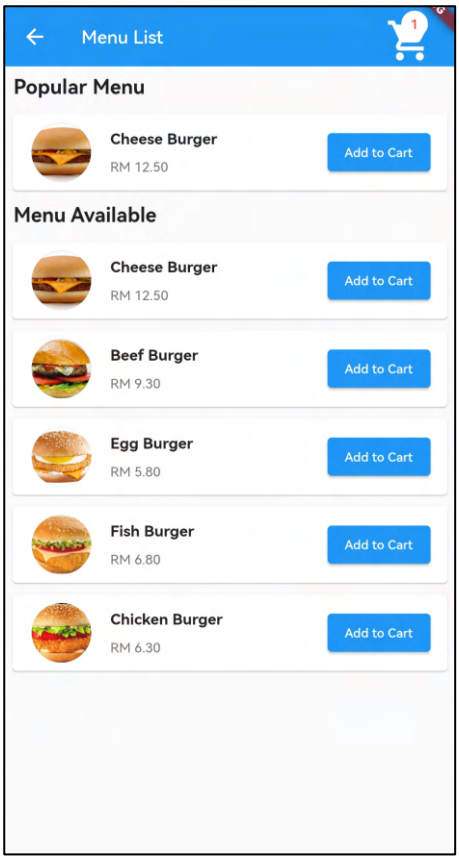

Figure 4.32 OMSR - Menu List Page After Add to Cart Pressed

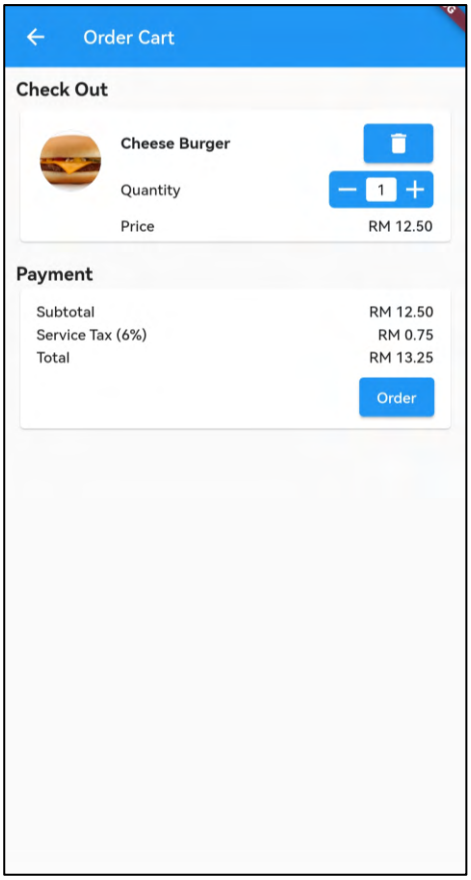

Figure 4.33 OMSR - Order Cart Page

After the customer pressed the "Order" button in Figure 4.33, a panel at the bottom will be displayed as shown in Figure 4.34. The customer will be required to select their order type to be "Dine In" or "Take Away". If "Dine In" is pressed, the system will request the customer to input their table number as shown in Figure 4.35. On the other hand, if "Take Away" is pressed, the customer needs to select the order collect time as shown in Figure 4.36. After they confirmed with their table number or order collect time, the system will record the order and redirect the customer to the order message page as shown in Figure 4.37 for order type "Dine In" and Figure 4.38 for order type "Take Away" respectively.

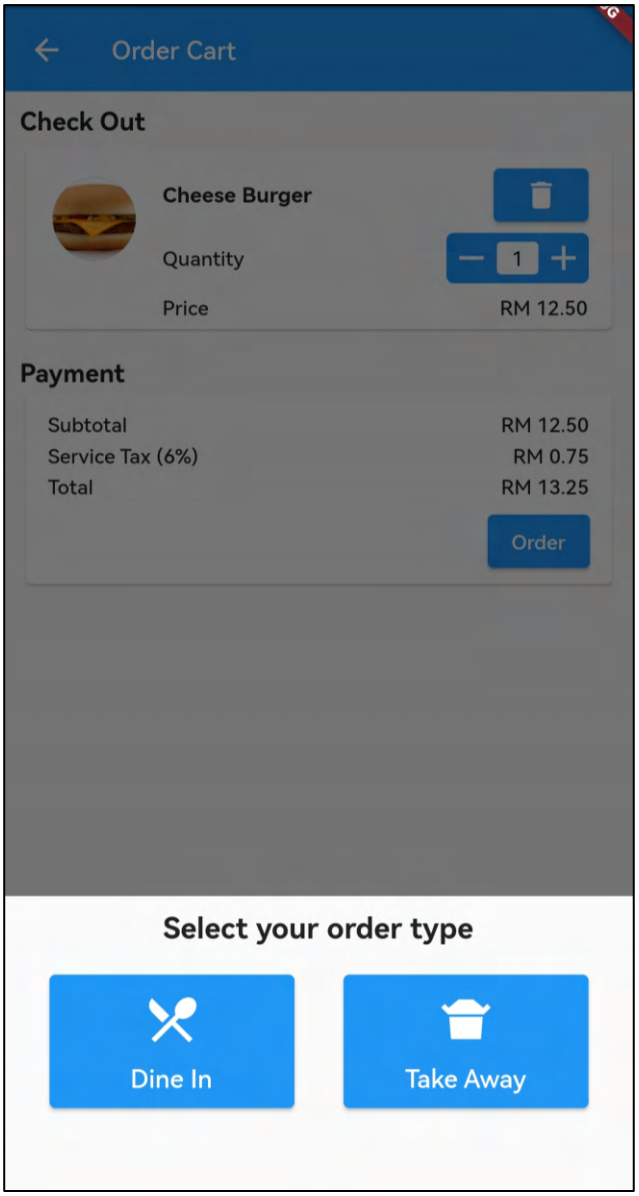

Figure 4.34 OMSR - Select Order Type Page

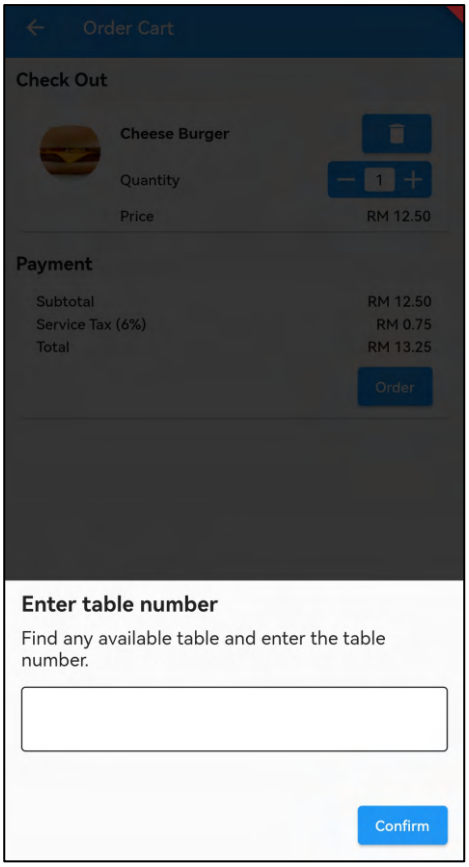

Figure 4.35 OMSR - Enter Table Number Page

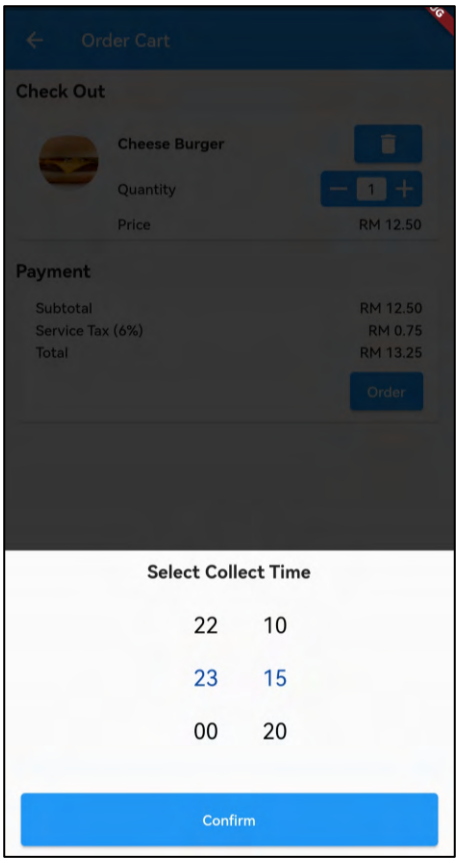

Figure 4.36 OMSR - Select Collect Time Page

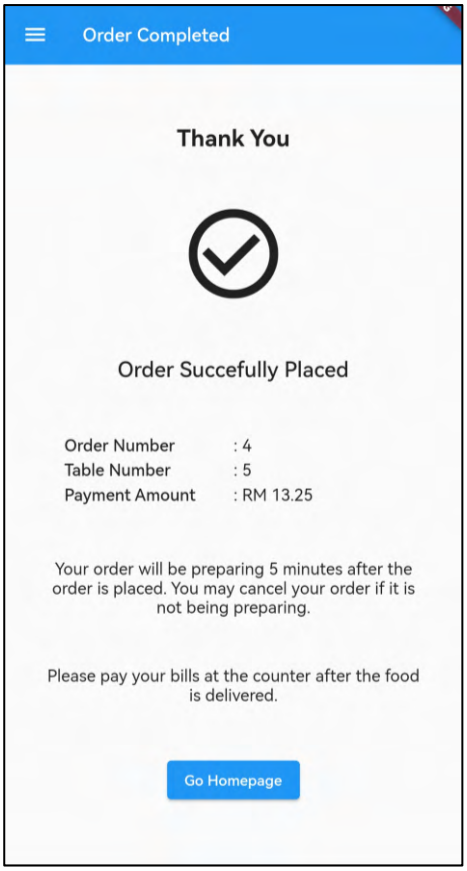

Figure 4.37 OMSR - Order Message Page for Dine In Order

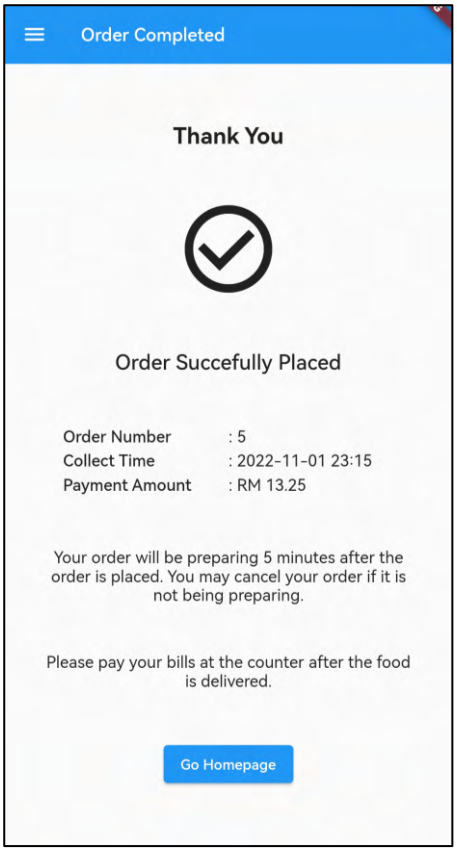

Figure 4.38 OMSR - Order Message Page for Take Away Order

4.4.6 Customer View Order Status Module Interfaces

Figure 4.39 shows the order status page with order status “On Queue”. The customer able to cancel their order by pressing the “Cancel Order” button. Customer can also press the “View Order” button to view the details of the order placed. Figure 4.40, Figure 4.41 and Figure 4.42 illustrates the order status page with order status “Preparing”, “To Be Serve” and “Delivered” respectively. As compared to Figure 4.39, there is no “Cancel Order” button in Figure 4.40, Figure 4.41 and Figure 4.42. This is because the customer is not allowed to cancel their order if their order status is not “On Queue”. Figure 4.43 depicts the order status page with no order which means the customer does not have any incomplete order. Figure 4.44 shows the order status page with cancel order panel. This panel will be displayed when the customer presses the “Cancel Order” button in Figure 4.39. If the customer confirmed to cancel their order, they need to press the “Confirm” button in the cancel order panel. If they press the “No” button, the cancel order panel will be closed and return to Figure 4.39.

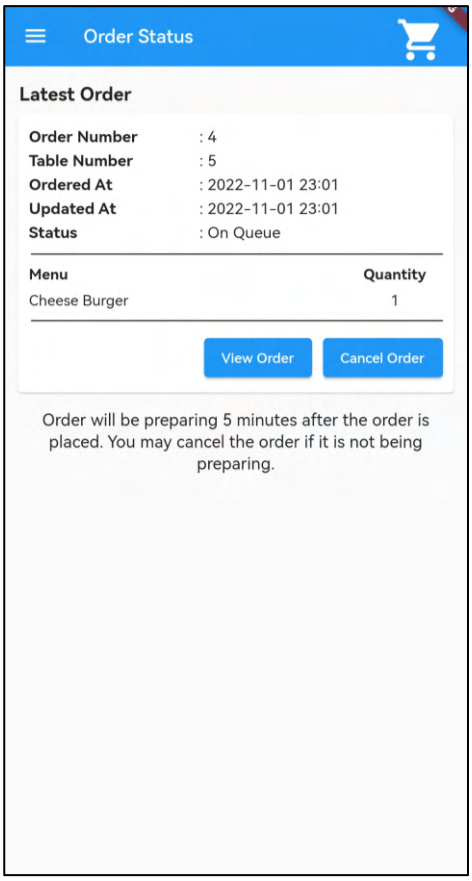

Figure 4.39 OMSR - Order Status Page - On Queue

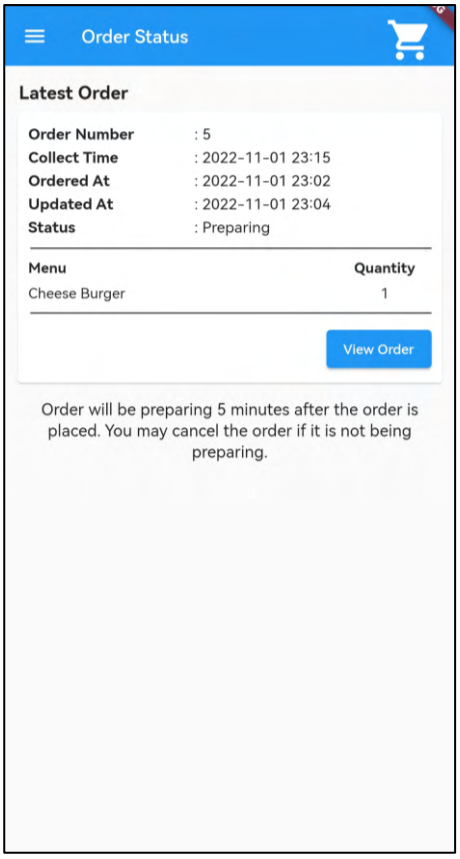

Figure 4.40 OMSR - Order Status Page - Preparing

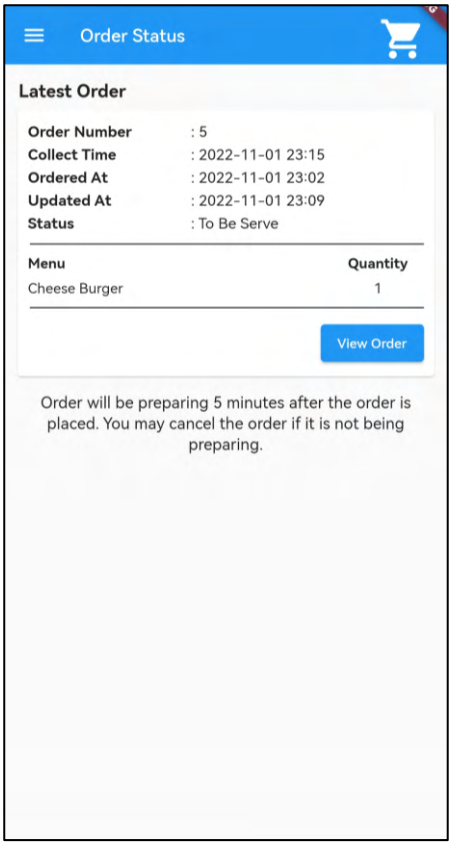

Figure 4.41 OMSR - Order Status Page - To Be Serve

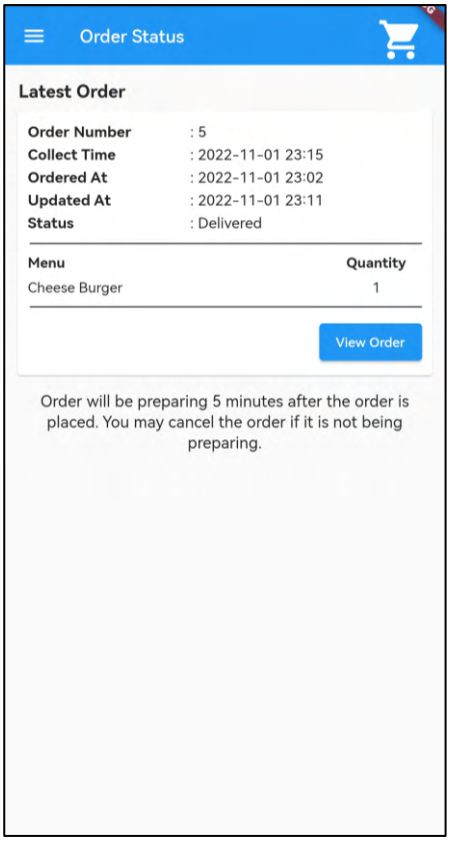

Figure 4.42 OMSR - Order Status Page - Delivered

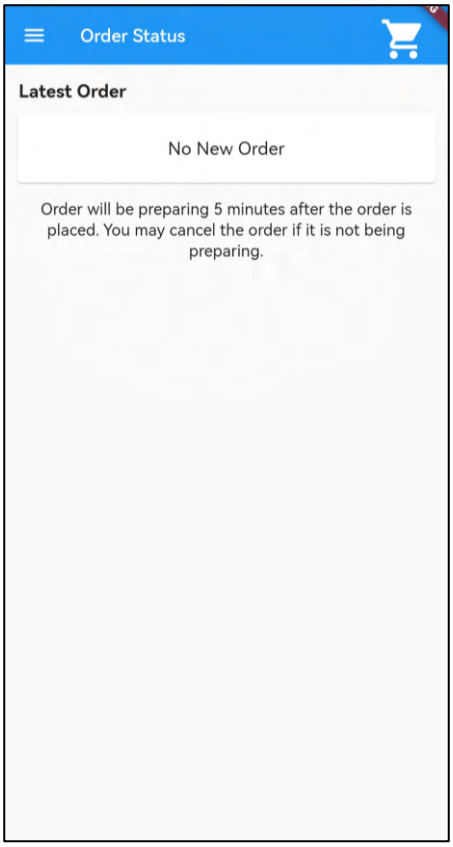

Figure 4.43 OMSR - Order Status Page - No Order

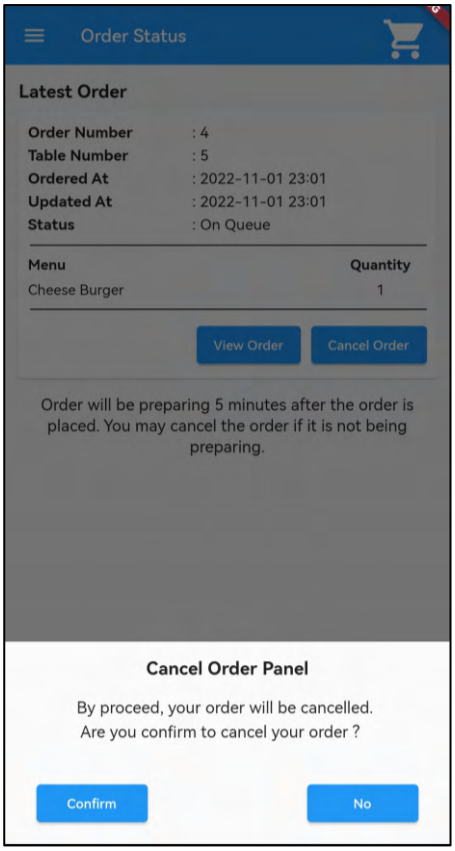

Figure 4.44 OMSR - Order Status Page - Cancel Order Panel

4.4.7 Customer View Order History Module Interfaces

Figure 4.45 shows the order history page of customer. Customer can view the details of the order by pressing the “View Order” button. After pressing, the customer will be redirected to Figure 4.46 if the order is incomplete and Figure 4.47 if the order is completed. Moreover, there will be a “Re-Order” button in order history page and order details page if the order is completed as shown in Figure 4.45 and Figure 4.47. Customer can press the button to order again the same menu. After pressing the “Re-Order” button, customer will be redirected to order cart page as shown in Figure 4.48.

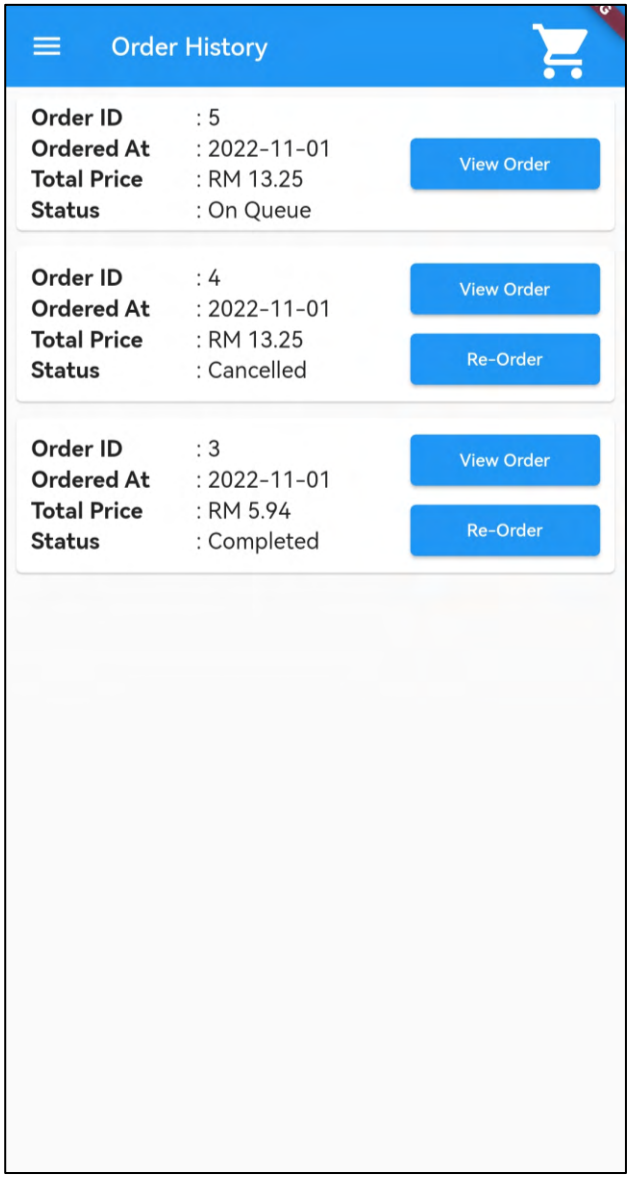

Figure 4.45 OMSR - Order History Page

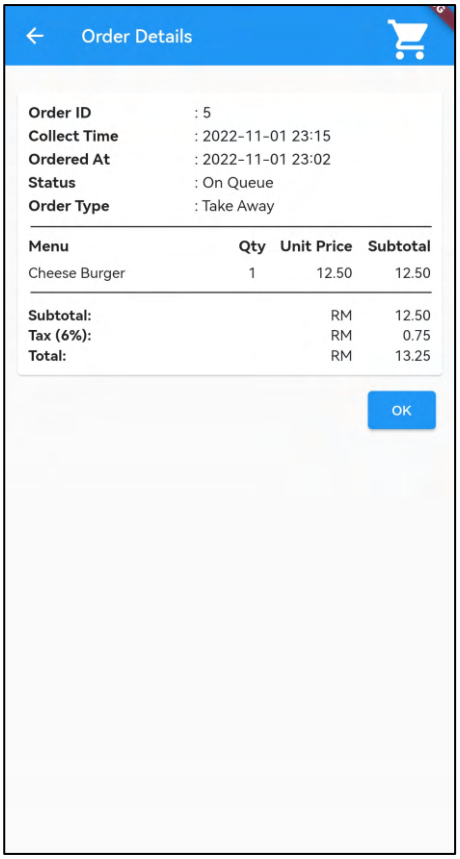

Figure 4.46 OMSR - Order Details Page for Incomplete Order

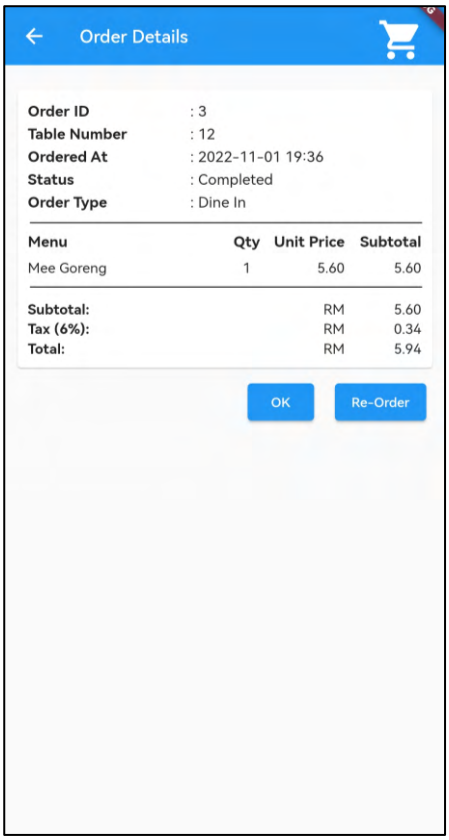

Figure 4.47 OMSR - Order Details Page for Completed Order

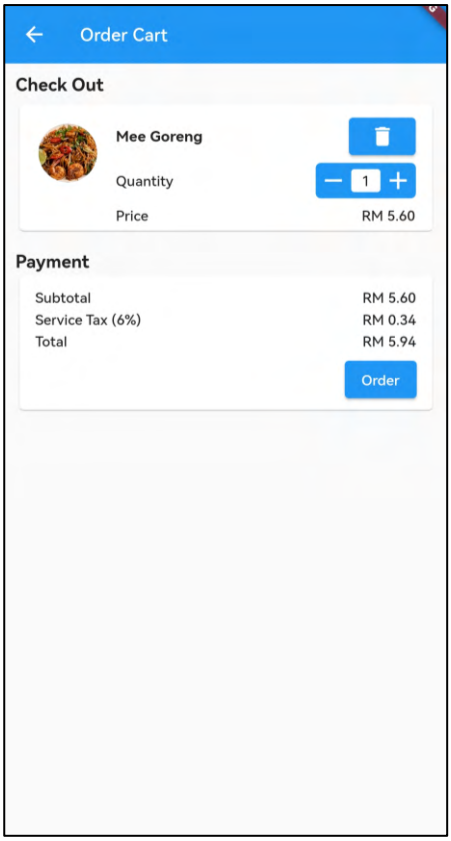

Figure 4.48 OMSR - Order Cart Page After Pressing Re-Order Button

4.4.8 Customer Provide Feedback Module Interfaces

Figure 4.49 illustrates the feedback form page. The customer is required to select their ratings and enter their comments to submit as feedback to the business owner. After they pressed the "Submit" button in Figure 4.49, the thank you message as shown in Figure 4.50 will be displayed to the customer.

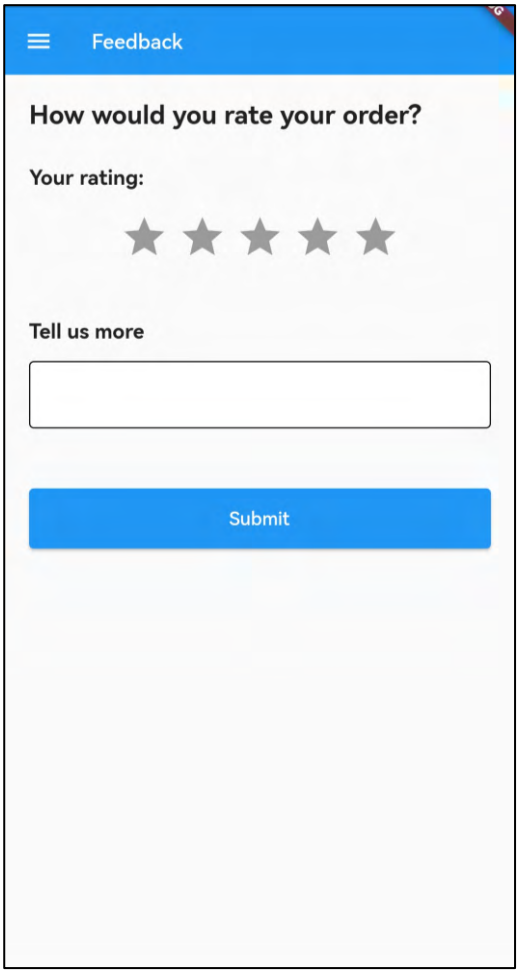

Figure 4.49 OMSR - Feedback Form Page

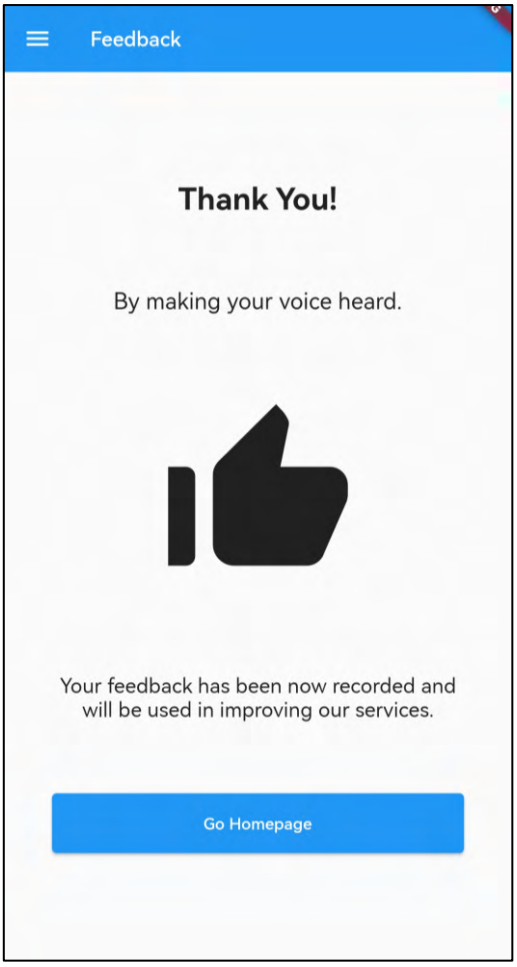

Figure 4.50 OMSR - Thank You Message Page

4.4.9 Business Owner Homepage

Figure 4.51 shows the homepage of business owner. The business owner can view sales report and customer feedback by pressing the related button. Other than pressing the button displayed on homepage, the business owner can also use the navigation menu to access the modules by pressing the menu button on left top. The result of pressing menu button is shown in Figure 4.52.

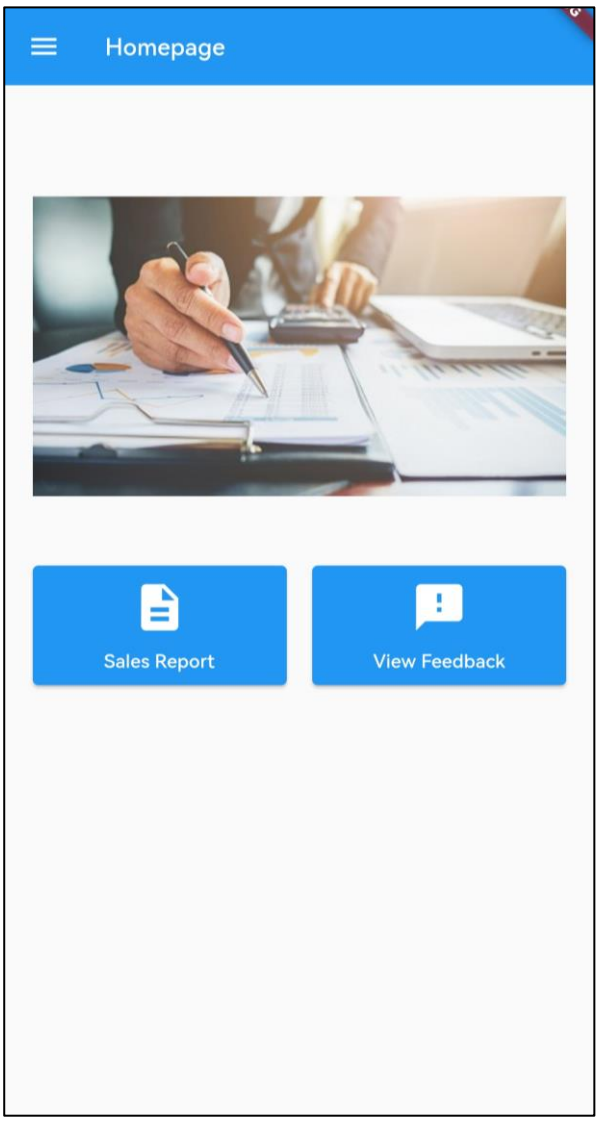

Figure 4.51 OMSR - Business Owner Homepage

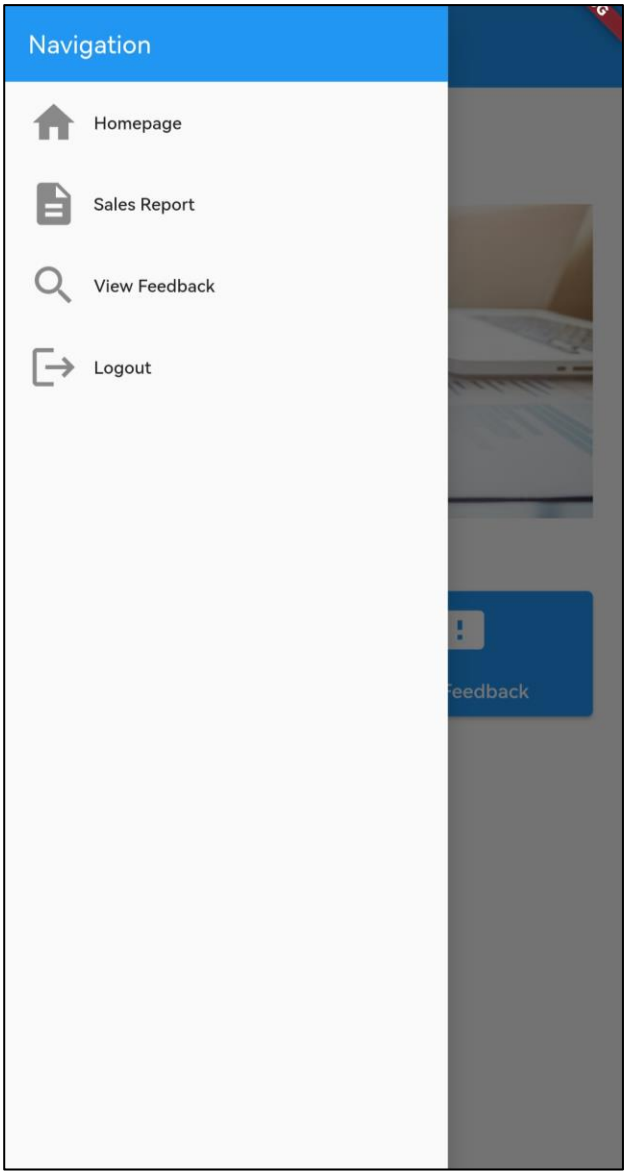

Figure 4.52 OMSR - Business Owner Navigation Menu

4.4.10 Business Owner View Sales Report Module Interfaces

Figure 4.53 illustrates the sales report page when the business owner accesses the view sales report module. All daily sales report will be displayed in this page with report ID, date of report generated and total sales. The business owner can view the details of report by pressing the "View Detail" button. After that, the system will display the details of sales report as shown in Figure 4.54.

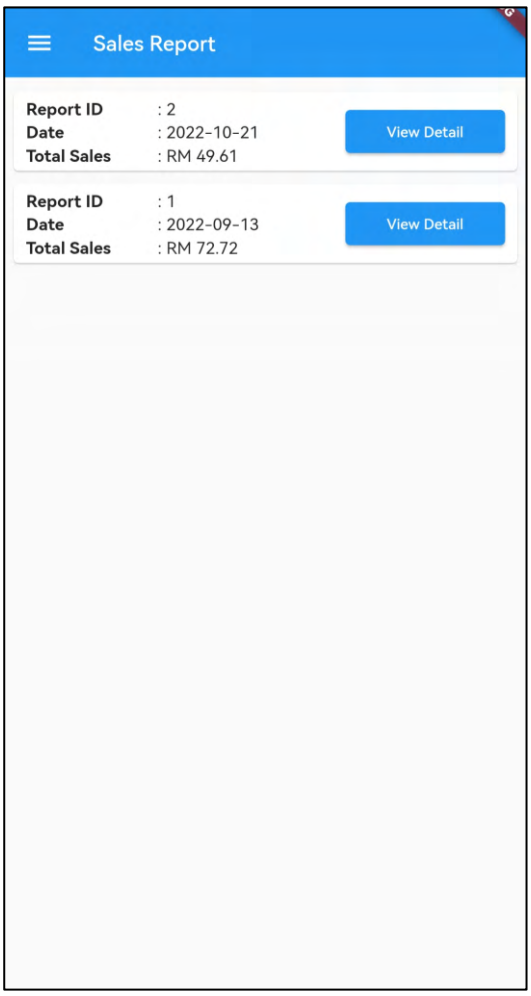

Figure 4.53 OMSR - Sales Report List Page

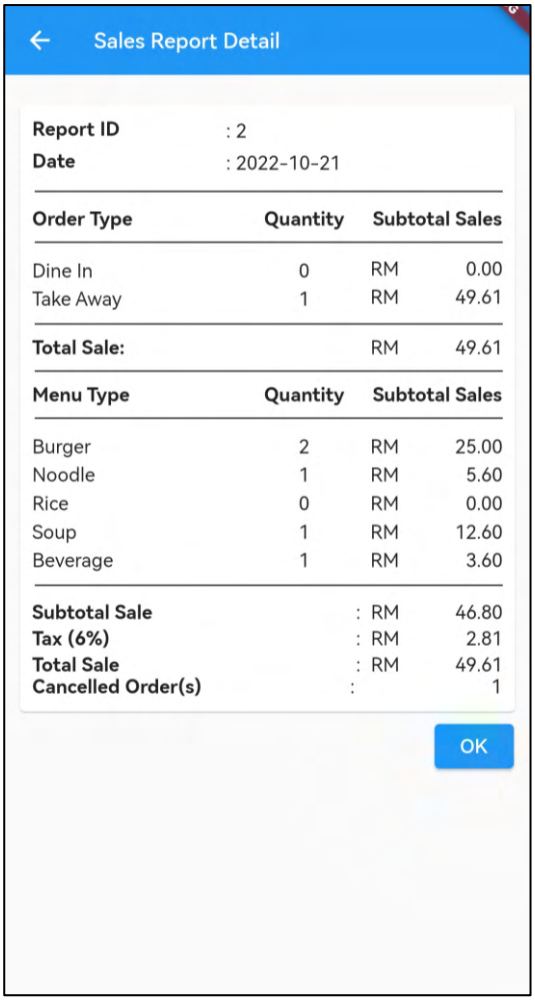

Figure 4.54 OMSR - Sales Report Details Page

4.4.11 Business Owner View Feedback Module Interfaces

Figure 4.55 depicts the feedback list when the business owner accesses the view feedback module. The ratings and comments given by the customer will be displayed to the business owner according to the date of feedback submitted in descending order. The business owner can also filter the feedback with date by pressing the date button displayed in Figure 4.55. Next, the business owner needs to select a range of desired date and press the "SAVE" button on right top as shown in Figure 4.56. If they wish to cancel the filter action, they can press on the cross icon on left top in Figure 4.56. Figure 4.57 shows the filtered customers' feedback.

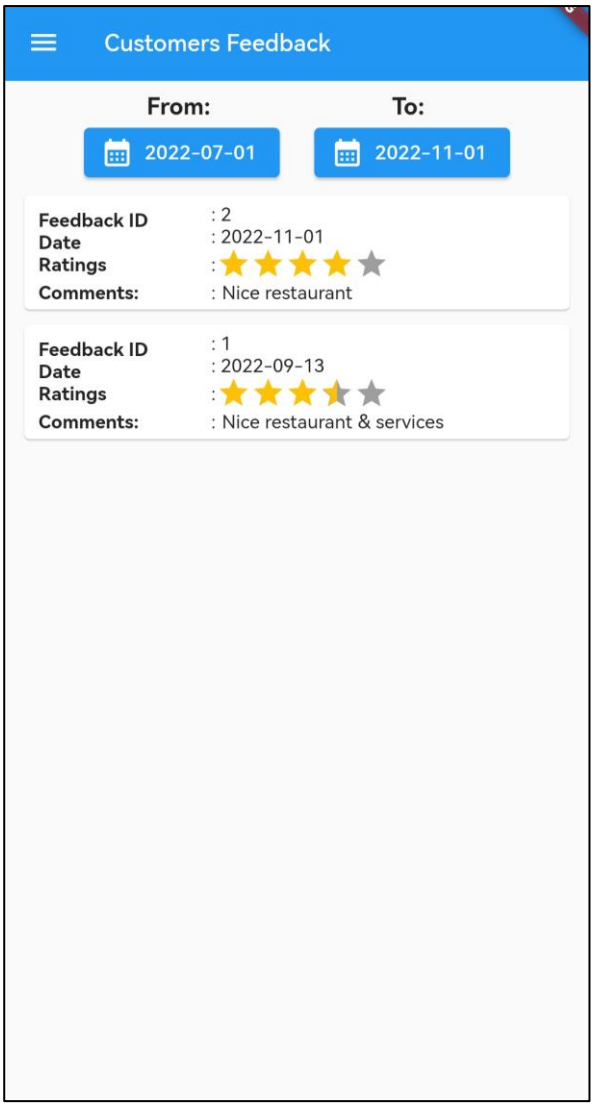

Figure 4.55 OMSR - Feedback List Page

Figure 4.56 OMSR - Select Date Range Page

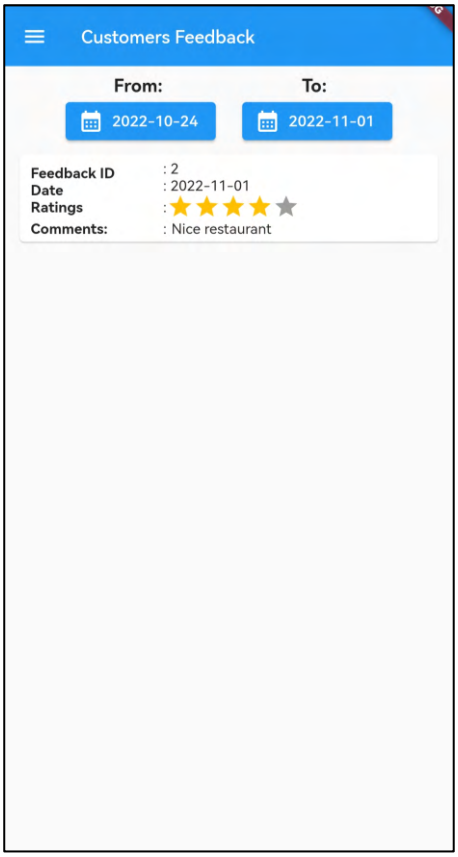

Figure 4.57 OMSR - Filtered Feedback List Page

4.4.12 Kitchen Staff Update Order Status Module Interfaces

Figure 4.58 depict the homepage of kitchen staff which is the order list page. Figure 4.59 shows the navigation menu for the kitchen staff when they press the menu button on left top. The kitchen staff will be able to view all the order with status “On Queue” and “Preparing” on top of the page while the order with status “Cancelled” on bottom of the page. Figure 4.60 shows the order list page with order that had status “On Queue” and Figure 4.62 shows the order list page with order that had status “Preparing”. By comparing Figure 4.60 and Figure 4.62, we can see that the order with status “Preparing” has an orange background. This is to help the kitchen staff easier to differentiate the order with different status. The kitchen staff need to update the order status from “On Queue” to “Preparing” or “Preparing” to “To Be Serve” by pressing the “Update Status” button if the prerequisite is satisfied. After the button is pressed, Figure 4.61 will be displayed for order with status “On Queue” and Figure 4.63 will be displayed for order with status “Preparing” then they need to press the “Confirm” button. Once the status is updated from “Preparing” to “To Be Serve”, that order will be removed from kitchen staff order list page.

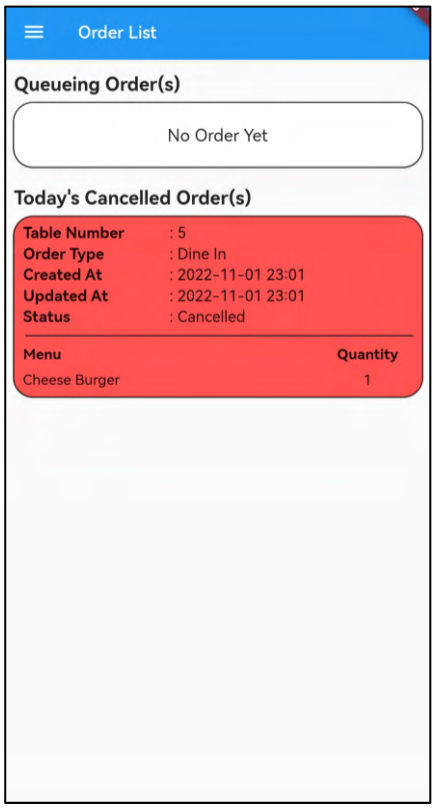

Figure 4.58 OMSR - Kitchen Staff Order List Page

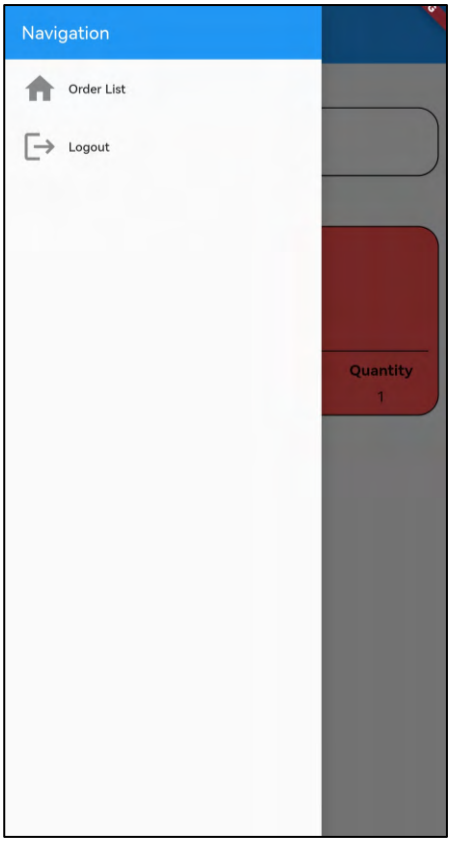

Figure 4.59 OMSR - Kitchen Staff Navigation Menu

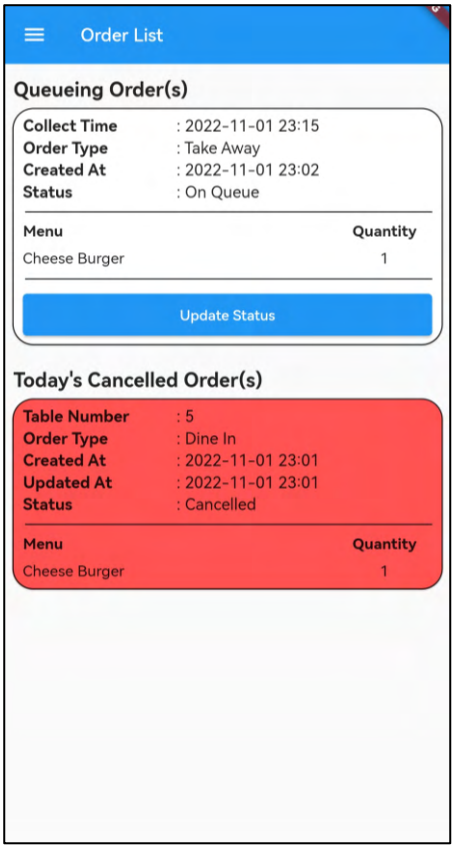

Figure 4.60 OMSR - Kitchen Staff Order List Page with Order Status On Queue

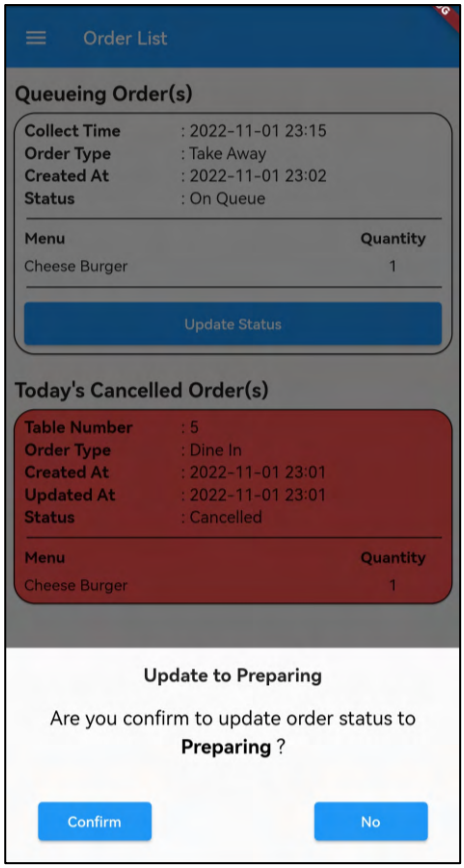

Figure 4.61 OMSR - Kitchen Staff Order List Page - Update On Queue Status Panel

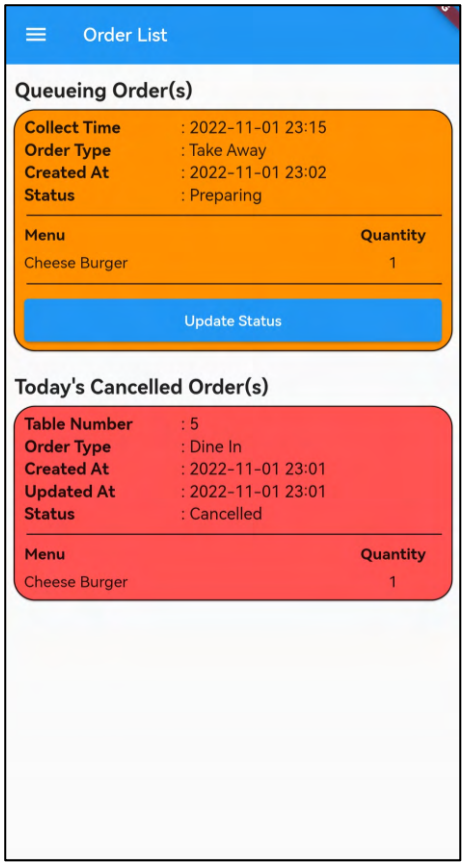

Figure 4.62 OMSR - Kitchen Staff Order List Page with Order Status Preparing

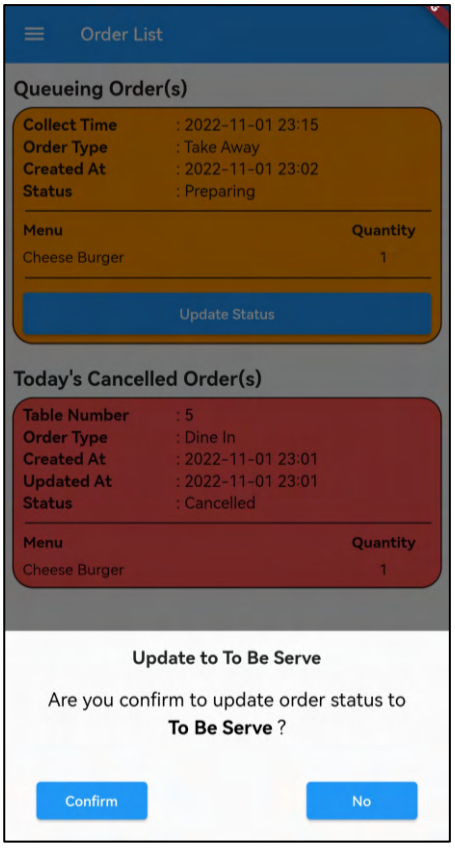

Figure 4.63 OMSR - Kitchen Staff Order List Page - Update Preparing Status Panel

4.4.13 Waiter Update Order Status Module Interfaces

Figure 4.64 illustrates the delivery list page of waiter and this is the homepage of waiter. Figure 4.65 shows the navigation menu when they press the menu button of left top in Figure 4.64. The waiter will be able to view all the order with status “To Be Serve” at the top of Figure 4.66 and order with status “Cancelled” at the bottom of Figure 4.66. Waiter must update the order status once the prerequisite is fulfilled. To do so, they need to press the “Update Status” button and a update confirmation panel will be displayed as shown in Figure 4.67. After that, the waiter needs to press the “Confirm” button to update the order status from “To Be Serve” to “Delivered”. Once the update is successful, the updated order will be removed from the delivery list.

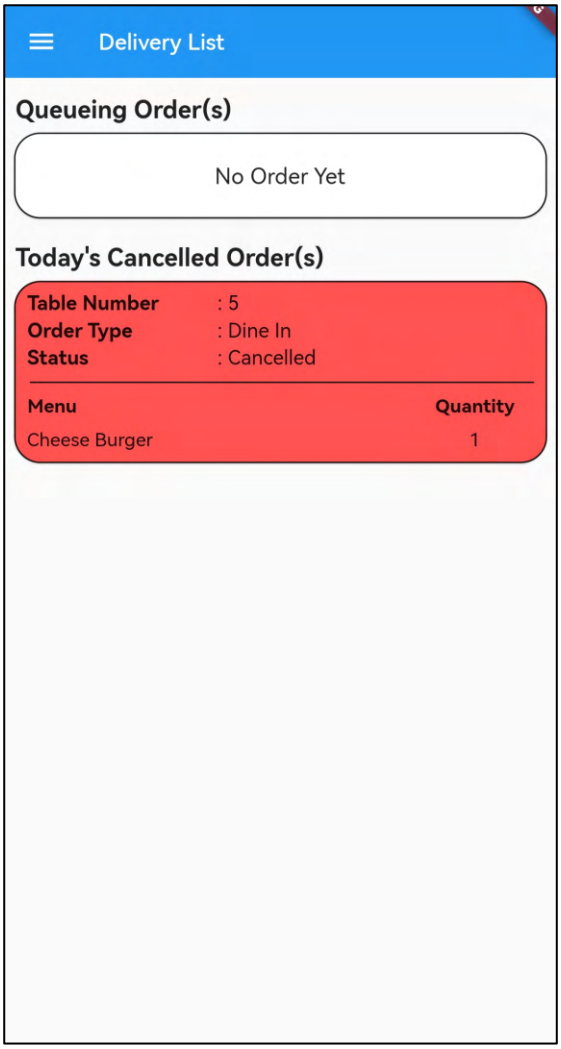

Figure 4.64 OMSR - Waiter Delivery List Page

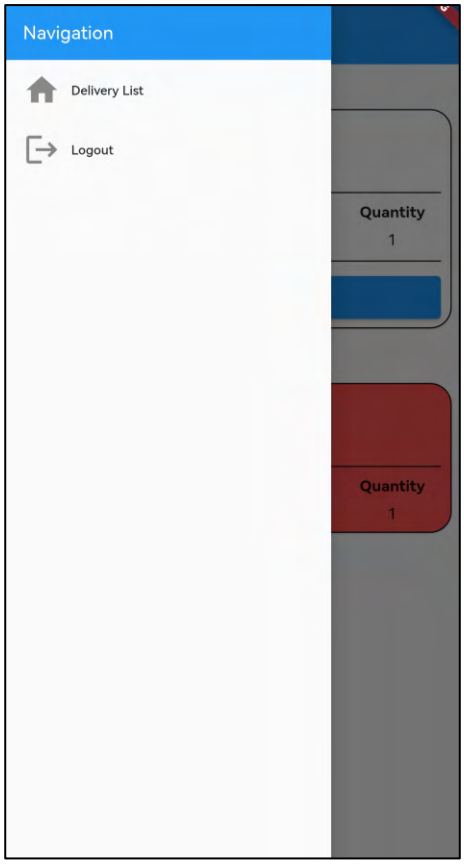

Figure 4.65 OMSR - Waiter Navigation Menu

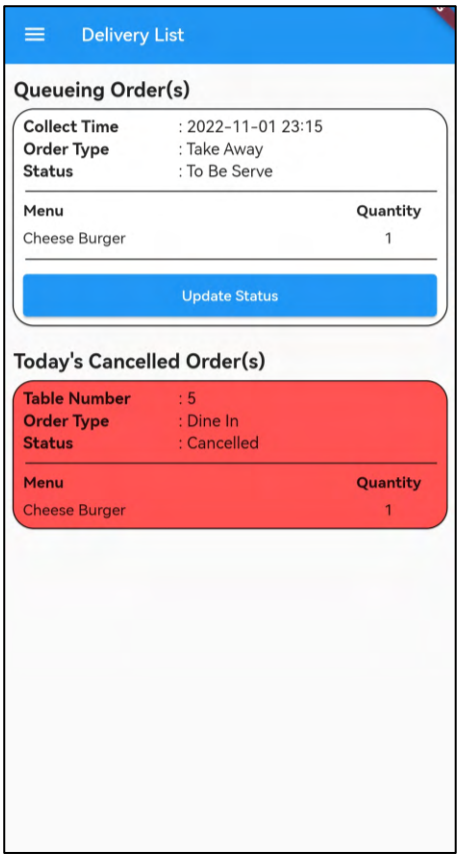

Figure 4.66 OMSR - Waiter Delivery List Page with Order Status To Be Serve

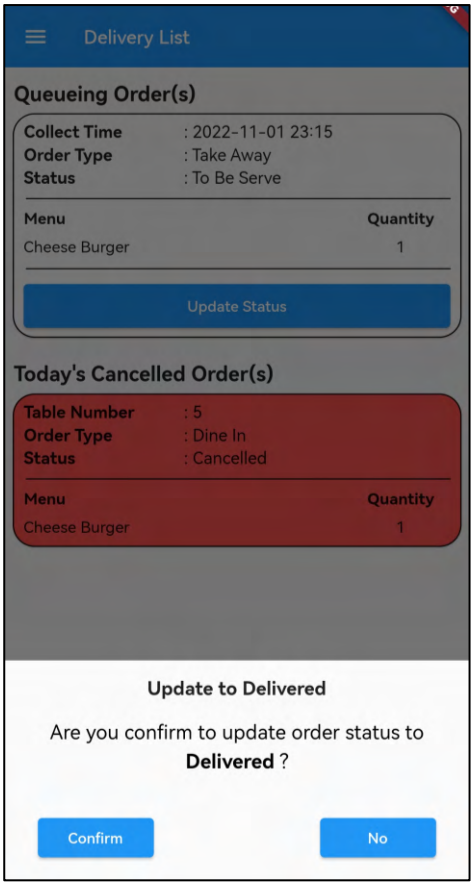

Figure 4.67 OMSR - Waiter Delivery List Page - Update To Be Serve Status Panel

4.4.14 Cashier Update Order Status Module Interfaces

Figure 4.68 shows the homepage of cashier which is the payment list page and Figure 4.69 depicts the navigation menu that will be displayed to the cashier when they press the menu button on left top of Figure 4.68. In this page, the order with status "Delivered" will be displayed under the "Queueing Order(s)" section while the order with status "Cancelled" will be displayed under the "Today's Cancelled Order(s)" section with a red background as shown in Figure 4.70. The cashier needs to update the order status from "Delivered" to "Completed" once they received the payment from the customer. Once the "Update Status" button is pressed, the update confirmation panel as shown in Figure 4.71 will be displayed and the cashier need to press the "Confirm" button.

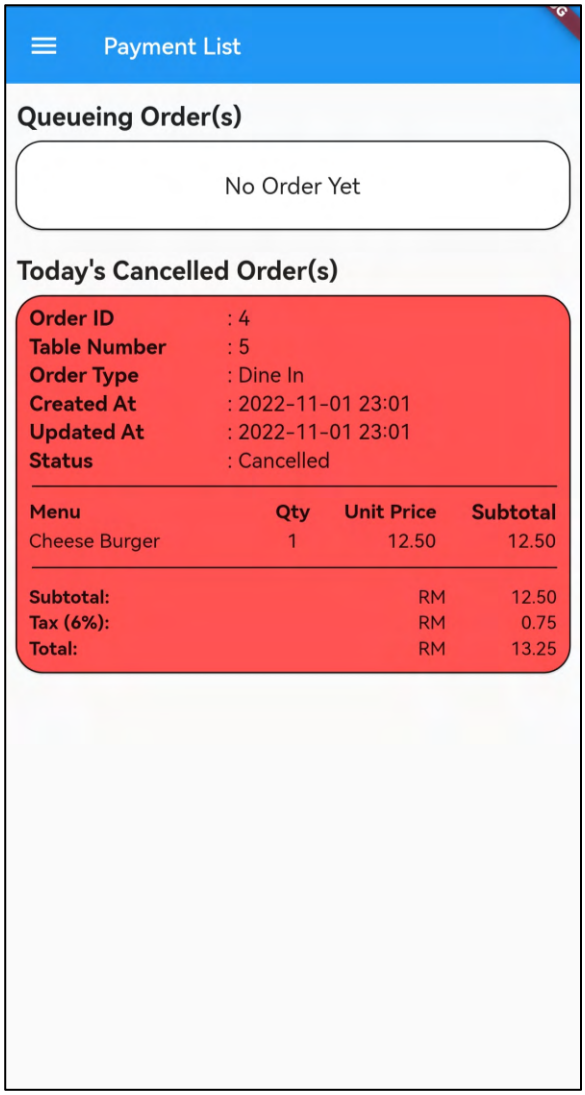

Figure 4.68 OMSR - Cashier Payment List Page

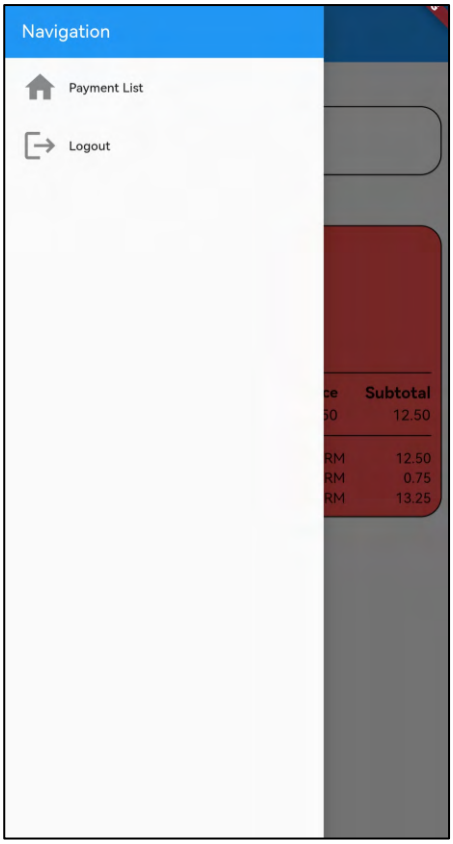

Figure 4.69 OMSR - Cashier Navigation Menu

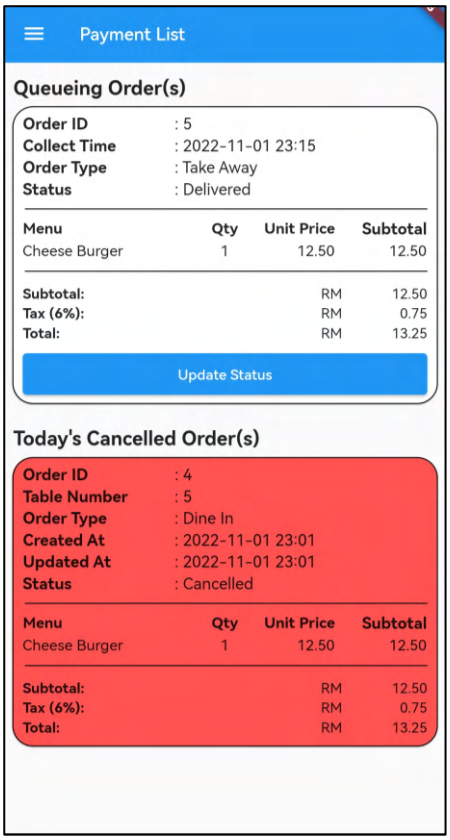

Figure 4.70 OMSR - Cashier Payment List Page with Order Status Delivered

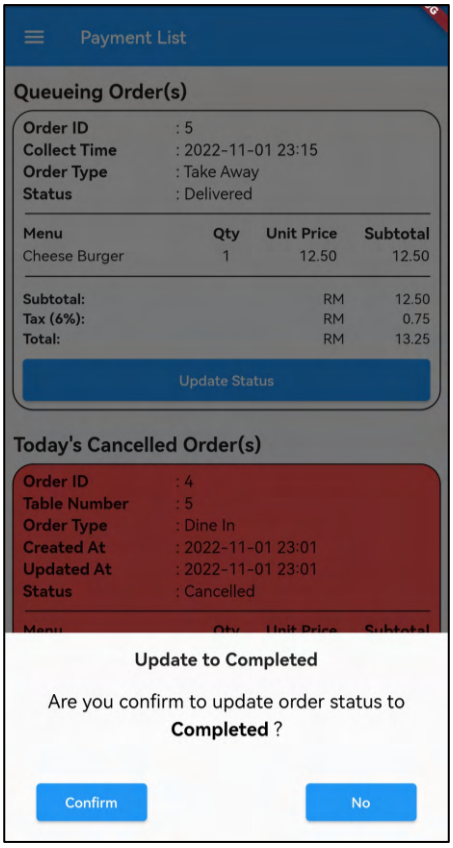

Figure 4.71 OMSR - Cashier Payment List Page - Update Completed Status Panel

4.5 User Manual

A user manual for Order Management System for Restaurant (OMSR) is produced to help the user in understanding the system. This user manual has included information such as description of the page, function of each button, and step-by-step tutorial on how to use a particular module. Refer **APPENDIX C** for the details of the application's user manual.

4.6 Testing and Result Discussion

Once the development of OMSR is completed, testing of the developed application will be tested to evaluate its usability and effectiveness. User Acceptance Test (UAT) form is used to test every single function provided by the system. Android mobile devices are used by the testers to test the application. There are 5 testers from different user type involved in the testing process. Feedback google form will also be given to them after the UAT is completed for collecting feedback regarding the developed application.

4.6.1 User Acceptance Testing (UAT)

Every function of the developed application was tested by 5 testers from different user type. The testers will compare the expected outcome and the actual outcome then mark the tested function as passed or failed. Moreover, any defects and failure will also be recorded in this form to help the developer in debugging. The result of UAT is shown in **APPENDIX D.**

4.6.2 Application Testing

After the testers completed UAT, feedback google form is distributed to the testers. The feedback google form consist of eight questions, the first question is ask about the role of the tester. Question 2 to question 7 is rating question where the tester needs to select either strongly agree, agree, neutral, disagree or strongly disagree. For the last question, the testers will give their feedback of using the application. The feedback google form is displayed in **APPENDIX E**

4.6.3 Result Discussion

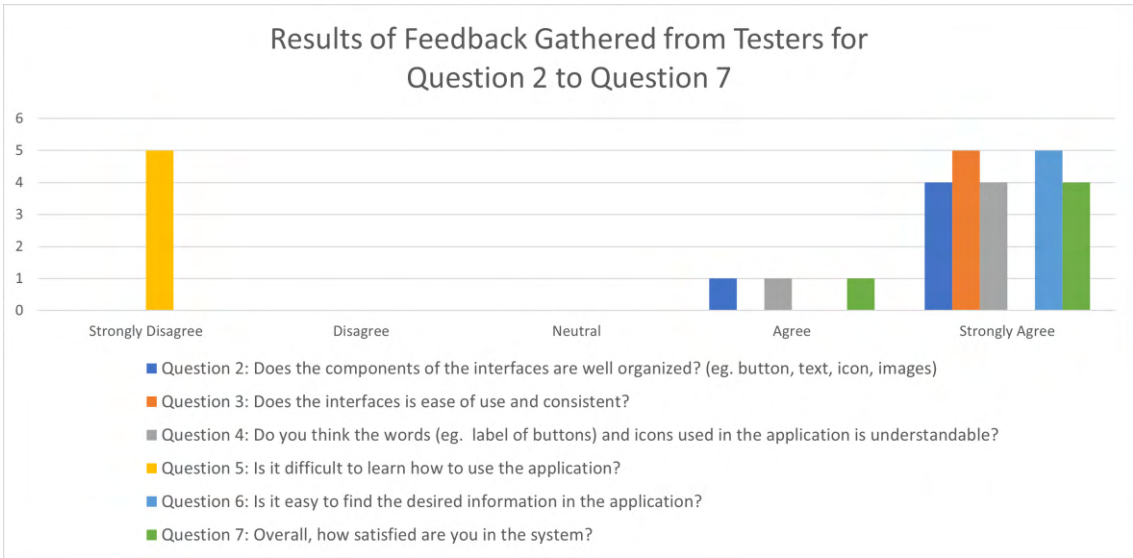

Figure 4.72 Results of Feedback Gathered from Testers for Question 2 to Question 7

Based on Figure 4.72, 80% of the testers answered question 2 with strongly agree where the components of the interfaces such as buttons, text, icons and images are well organized and 20% of the testers answered agree for this statement. Moreover, the interfaces are ease of use and consistent as all testers strongly agree to question 3. Next, 80% of the testers strongly agree that the words and icons used in the application understandable and 20% of them answered agree for question 4. Furthermore, all testers agreed that they can easily learn how to use the application and find their desired information in the application. Lastly, for question 7, 80% of the testers rated very satisfied with the system while 20% of them rated satisfied with the system. Based on the analysis, all testers answered the feedback google form with strongly agree and agree. Hence, we can conclude that the developed application received positive feedbacks from the testers.

There are also some comments on the developed application given by the testers while they filling the feedback google form. Table 4.2 shows the summary of the comments. These comments can be used to enhance the developed application for future works.

Table 4.2 Summary of Testers Comments Gathered

No	Comments
1	The interface can improve more.
2	Very detailed and complete order list, similar to the ones seen in actual restaurants.
3	Can add in payment type and printing of order at kitchen.
4	No.
5	Nope, quite a good functioning application.

CHAPTER 5

CONCLUSION

5.1 Introduction

Chapter 5 will discuss the summary of the development process of Order Management System for Restaurant (OMSR) application to achieve the stated objectives and problem statement as discussed in Chapter 1 of this thesis. The traditional method that still being used by most of the restaurant are highly dependent to the staff where the waiter needs to serve the customer, get order from them deliver meals as well as response to the customer's request. The staff could get overloaded especially during peak hours. The proposed system can play a vital role in simplifying and facilitating the ordering and management process, reducing the operating expenses as well as the workload of the staff.

During the development of the proposed application, several software is used. For instance, the Android Studio software is used as the Integrated Development Environment (IDE), Flutter is used as the main framework of the proposed application and the Firebase is used as the cloud-based database of the proposed application and authentication system.

The methodology used for the development of this project is Rapid Application Development (RAD). It enables the project to be completed in a shorter time frame and easier to accommodate changes that occur throughout the development process.

The developed application has gone through an evaluation by five different user types which are the customer, the business owner, the kitchen staff, the waiter, and the cashier. User Acceptance Test (UAT) was also performed to evaluate the effectiveness, usability, and functionality of the system.

5.2 Research Constraint

The constraints during the development of the project are:

i. Limitation of Time

There are some functionalities unable to be implemented to the proposed system due to limited time. For instance, the module for administrator to manage the staff and menus as well as the implementation of payment gateway.

ii. Workforce

As the planning, development, testing and documentation of this project are all completed by the author only, hence, the functionality of the system and the quality of the deliverable are limited.

iii. Operating Environment

Due to the limitation of time and workforce, the developed application only supports Android mobile devices.

5.3 Future Works

Although the developed Order Management System for Restaurant (OMSR) has fulfilled the recorded requirements, there are still a lot of untapped potentials. A few potentials of future works are as listed below:

i. Improve Payment Module

Due to the limitation of time, the current version of the OMSR only allows the customer to pay their bills at the counter. As the world today is moving towards a cashless society, hence, the payment module needs to be integrated with payment gateway to keep up with the trends. Besides, the payment module should also be added with a function for the cashier to select the payment type of the received payment such as Visa/Master, Online Banking and Cash.

ii. Improve Interoperability

The current OMSR version is restricted to only able to be installed on Android mobile devices. This would be a big issue for the user who do not have an Android mobile device.

iii. Implementation of Data Management Module

The current OMSR version does not support the modification of existing staff data and menu data. The implementation of data management module is important for allowing the client to perform task such as add, update or delete a staff data or menu data. Moreover, a new user type, the Administrator, can also be added to the system with the implementation of data management module. Administrator is the person responsible for managing the staff data and menu data.

REFERENCES

Ahn, J., & Back, K.-J. (2017). Influence of brand relationship on customer attitude toward integrated resort brands: a cognitive, affective, and conative perspective. *Journal of Travel & Tourism Marketing*, *35*(4), 449–460. https://doi.org/10.1080/10548408.2017.1358239

Android Studio. (2019). Download Android Studio and SDK tools. Android Developers. https://developer.android.com/studio

Biswas, N. (2018, June 30). *9 Benefits of using a Restaurant POS System*. Focus Softnet India. https://www.focussoftnet.com/blogs/9-benefits-of-a-restaurant-pos-software

Dahake, K., & Bhoi, P. A. D. (2019). ANDROID BASED CANTEEN AUTOMATION USING WIFI. *JournalNX - a Multidisciplinary Peer Reviewed Journal*, *5*(02), 1–6. https://repo.journalnx.com/index.php/nx/article/view/1833

Flutter. (n.d.). *Flutter documentation*. Docs.flutter.dev. https://docs.flutter.dev/

Google. (2019). *Cloud Firestore | Firebase*. Firebase. https://firebase.google.com/docs/firestore

Hospitality Technology. (2011, May 20). *Pizza Hut Goes for Mobile Ordering Trifecta*. Hospitality Technology. https://hospitalitytech.com/pizza-hut-goes-mobile-ordering-trifecta

Lucidchart Content Team. (2018). *4 Phases of Rapid Application Development Methodology | Lucidchart Blog*. Lucidchart.com. https://www.lucidchart.com/blog/rapid-application-development-methodology

Pizza Hut. (n.d.). *Hut Life – Official Pizza Hut Blog*. Hut Life – Pizza Hut Brand Blog. https://blog.pizzahut.com/our-story

Pizza Hut. (2019). *Pizza Hut Malaysia*. Pizzahut.com.my. https://www.pizzahut.com.my/aboutus

Samuely, A. (2017). *McDonald's 7M app downloads highlights effectiveness of relevant incentives | Retail Dive*. Retaildive.com. https://www.retaildive.com/ex/mobilecommercedaily/mcdonalds-7m-app-downloads-highlights-effectiveness-of-welcome-incentives

Shahjee, R. (2016). *THE IMPACT OF ELECTRONIC COMMERCE ON BUSINESS ORGANIZATION* (pp. 3130–3140). Scholarly Research Journal for Interdisciplinary Studies. https://oaji.net/articles/2017/1174-1484826380.pdf

Singh, P., Tembhekar, N., Gurve, K., & Rahate. (2020). *SMART FOOD ORDERING SYSTEM FOR RESTAURANT* (pp. 72–74). International Research Journal of Engineering and Technology (IRJET). https://www.irjet.net/archives/V7/i2/IRJET-V7I215.pdf

Software Testing Help. (2016, July 21). *How to Test Point of Sale (POS) System - Restaurant POS Testing Example*. Software Testing Help. https://www.softwaretestinghelp.com/how-to-test-point-of-sale-pos-system/

Stubbs, A. T., & Conrad, A. (2019, March 26). *What Is a Point of Sale System? A Guide to POS Features*. Software Advice. https://www.softwareadvice.com/resources/what-is-a-point-of-sale-system/

Subramanian, K. (2018). Can Automation Eliminate Human Intervention? *International Journal of Engineering and Management Research*, *8*(3), 100–108. https://www.researchgate.net/publication/327369088_Can_Automation_Eliminate_Human_Intervention

tutorialspoint. (2019). *SDLC - RAD Model - Tutorialspoint*. Tutorialspoint.com. https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm

APPENDIX A
SOFTWARE REQUIREMENT SPECIFICATION (SRS)

For Appendices Heading, use TITLE AT ROMAN PAGES style.

2022

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

[ORDER MANAGEMENT SYSTEM FOR RESTAURANT]

DOCUMENT APPROVAL

	Name	Date
Authenticated by:  _____ Name	Tan Chee Kin	
Approved by: _____ Client		

Software : Microsoft Office

Archiving Place : Google Drive

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

1.1 PROJECT DESCRIPTION

Order Management System for Restaurant is a mobile application that developed to help the restaurant business owner to digitalize the business process, improve customers' experience as well as reduce operating expenses of the business. It facilitates the communication between the customer, waiters, cashiers, and the kitchen staff and ensures a smoother workflow at a lower error rate compared to the traditional method. The proposed system has seven modules which are Place Order, View Order Status, View Order History, Provide Feedback, View Feedback, View Sales Report and Update Order Status.

- Place Order Module is the module that allows the customer to place their order via the system.
- View Order Status Module is the module that allows the customer to keep track with the status of their placed order. The customer can also cancel their placed order here if the order fulfilled the constraints.
- View Order History Module is the module that allows the customer to view their order history and perform a reorder action if they want to.
- Provide Feedback Module is the module that provide the customer with the ability to provide feedback to the restaurant.
- View Feedback Module is the module that provide the business owner with the ability to view the feedback collection from customers. The business owner can also select a range of dates to filter the feedback data.
- View Sales Report Module is the module that allows the business owner to view sales report of the restaurant.

- Update Order Status Module is the module that provide the restaurant staff which are the kitchen staff, the waiter and the cashier to view customers' orders and update the order status after the constraints is fulfilled.

1.2 SYSTEM IDENTIFICATION

The system uses the following convention:

System Identification Number: SRS-OMSR-V01-22

Table 1.1 System Identification Abbreviations

SRS	Software Requirement Specification
OMSR	Order Management System for Restaurant
V01	Version 1
22	Year 2022

1.3 CONTEXT DIAGRAM

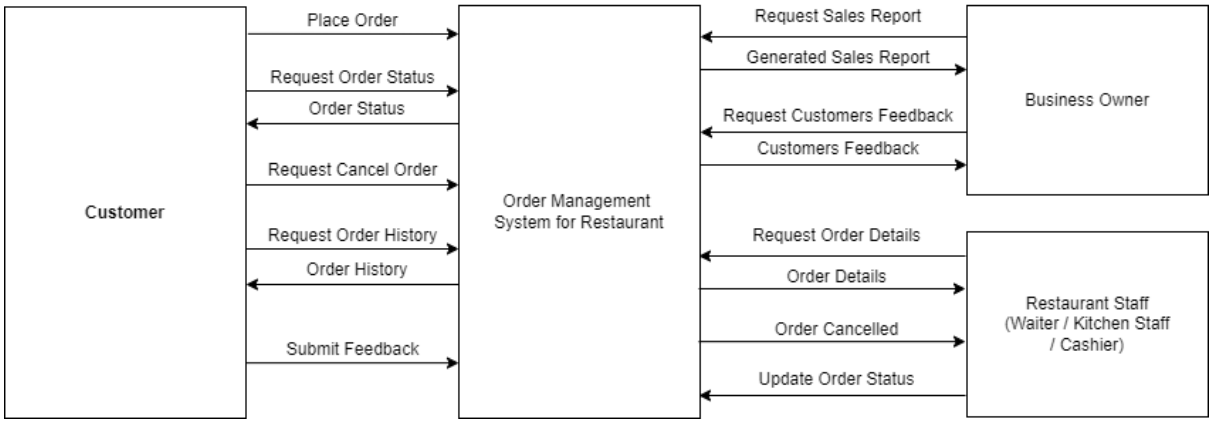

Figure 1.1 Context Diagram of Proposed System

Figure 1.1 shows the context diagram of the order management system for restaurant. There are three entities in the proposed system. The first entity is the customer. Customer can access four main modules of the proposed system which are place order, view order status, view order history and provide feedback. The second entity is business owner who can access two main modules of the proposed system which are view sales reports and view feedback. The last entity is the restaurant staff. They can access only one function of the proposed system which is updating the status.

When the customer places an order, the system will pass the order details to the restaurant staff and the restaurant staff can then update the order status in the update order status module once the constraints have been met. The customer can also request order status to view the order status in the view order status module or request order history in the view order history module to view their order history. Besides, the customer can request cancel the order in the view order status module to cancel their order as long as the order is not being prepared and the system will then inform the restaurant staff. Moreover, the customer can also submit feedback in provide feedback module and the business owner can view the feedback in the view feedback module. Lastly, the business owner can also view the sales report generated by the system by accessing the view sales report module.

CHAPTER 2

2.1 PROJECT DESCRIPTION

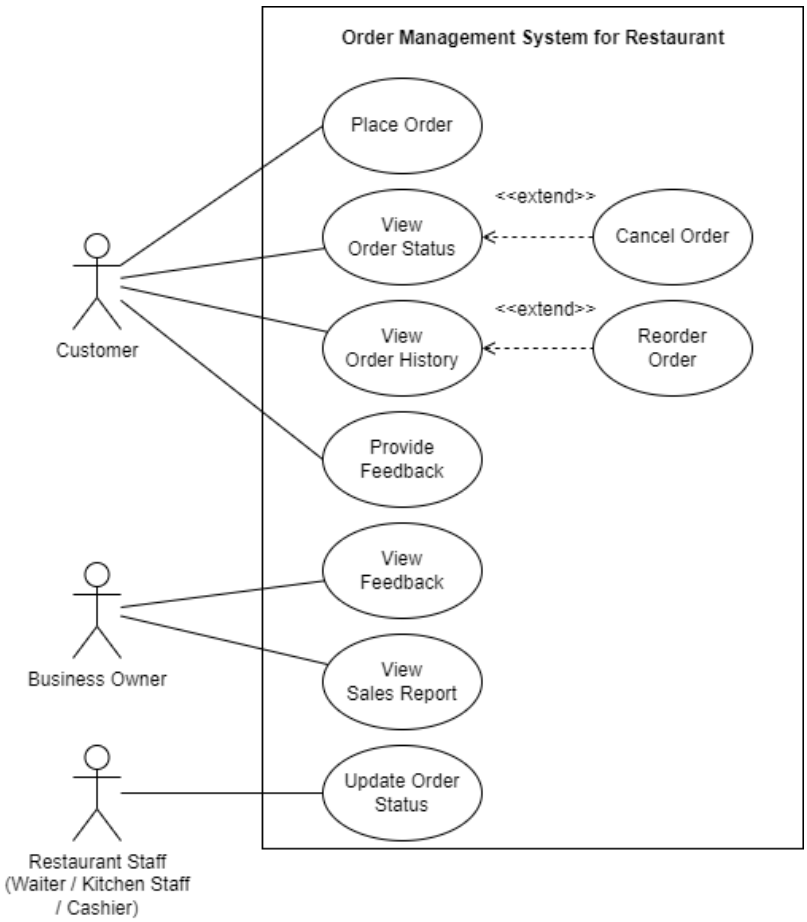

Figure 2.1 Use Case Diagram of Proposed System

Figure 2.1 shows the use case diagram of the order management system for restaurant. According to the figure, there are three types of users using the system. The first user is the customer, who can place an order, view order status, cancel an order under certain conditions, view order history and provide feedback. The next user is the business owner who can view the feedback provided by the customer and view the sales report generated by the system. Lastly, restaurant staff are the third user which consists of waiters, kitchen staff and cashiers. The restaurant staff can view and update the order status of the customer.

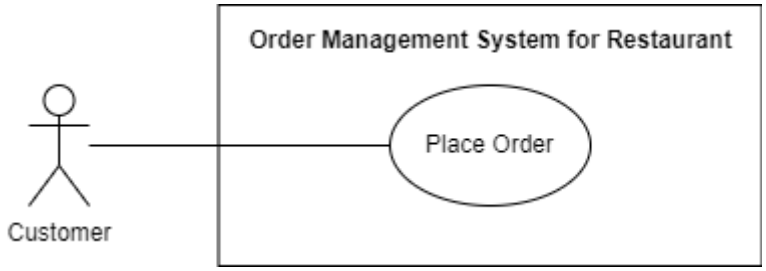

Figure 2.2 Use Case Diagram of Place Order Module

Table 2.1 Use Case Description of Place Order Module

Use Case ID	UC001
Brief Description	This use case describes the process of the customers placing an order.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet.
Basic Flow	**[B1: Place Order]** 1. Use case starts when the customer presses the "Place Order" button. 2. System retrieves menu type data from the type table in database. 3. System displays menu type data in the "Menu Type" page. 4. Customer presses intended menu type. 5. System retrieves menu data of the selected menu type. 6. System displays retrieved menu data in "Menu List" page. 7. Customer presses the "Add to Cart" button to add an intended menu to the order cart. 8. System retrieves data of the intended menu and adds it to the order cart. 9. Customer presses the "Order Cart" button to view their intended menu. 10. System retrieves order cart data from order cart table in database. 11. System displays order cart data in the "Order Cart" page. 12. Customer presses the "Order" button to place their order. 13. System prompts the customer to select an order type. **[A1: Dine In] [A2: Take Away]** 14. System saves data into order database. 15. Use case ends.
Alternative Flow	**[A1: Dine In]**

	1. Use case starts when the customer presses the “Dine In” button. 2. System prompts the customer to input their table number. 3. Customer inputs table number and presses the “Confirm” button. 4. Back to basic flow step 13. **[A2: Take Away]** 1. Use case starts when the customer presses the “Take Away” button. 2. System prompts the customer to select a collection time. 3. Customer selects collection time and presses the “Confirm” button. 4. Back to basic flow step 13.
Exception Flow	NONE
Post-Conditions	Customer will be redirected to the home page.
Rules	**[R1: Table Number]** 1. Customer can only input the table number where they sit on. **[R2: Collection Time]** 1. Customer can only select a time within the range of restaurant business hours.
Constraints	Only the registered customer will be able to place an order.

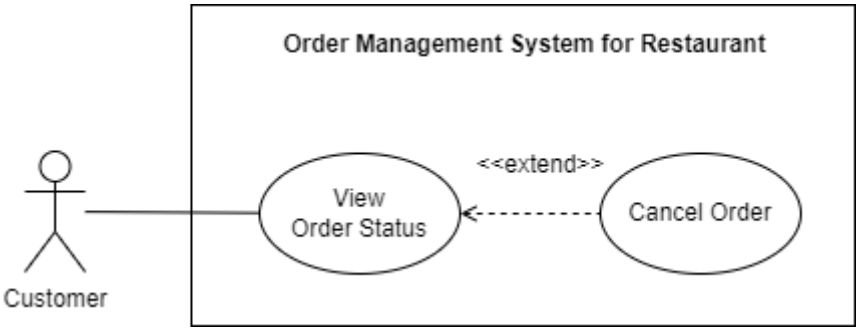

Figure 2.3 Use Case Diagram of View Order Status Module

Table 2.2 Use Case Description of View Order Status Module

Use Case ID	UC002
Brief Description	This use case describes the process of the customers viewing order status and cancelling their order if they want to.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet. 3. Customer had placed an order and the order status is not "Completed" or "Cancelled".
Basic Flow	**[B1: View Order Status]** 1. Use case starts when the customer presses the "Order Status" button. 2. System retrieves customer's order data from the order table in database. **[A1: No Order Data]** 3. System displays order data in the "Order Status" page. 4. Use case ends. **[B2: Cancel Order]** 1. Use case starts when the customer is at the "Order Status" page. 2. Customer presses the "Cancel Order" button to request to cancel an order. 3. System prompts confirmation message. **[A2: Press No]** 4. Customer presses the "Confirm" button to cancel the order. 5. System updates order status to "Cancelled" in order table in database. 6. Use case ends.
Alternative Flow	**[A1: No Order Data]** 1. System retrieves customer's order data from order table in database.

	2. System unable to retrieve data as the customer does not have an order with a status other than "Completed" or "Cancelled". 3. System displays "No New Order" in the "Order Status" page. 4. Use case ends. **[A2: Press No]** 1. Alternative flow continues at Use Case of View Order Status Module, Basic Flow, B1: View Order Status, Step 3. 2. Use case ends
Exception Flow	NONE
Post-Conditions	NONE
Rules	**[R1: Cancel Order]** 1. Order can only be cancelled when its status is "On Queue".
Constraints	Only the customer will be able to perform this use case.

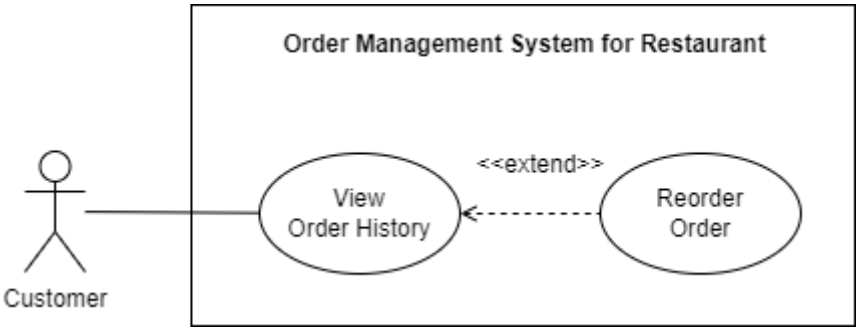

Figure 2.4 Use Case Diagram of View Order History Module

Table 2.3 Use Case Description of View Order History Module

Use Case ID	UC003
Brief Description	This use case describes the process of the customers viewing order history and reordering their previous order.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet. 3. Customer had placed an order.
Basic Flow	**[B1: View Order History]** 1. Use case starts when the customer presses the "Order History" button. 2. System retrieves customer's order data from order table in database. **[A1: No Order History]** 3. System displays order data in the "Order History" page. 4. Customer presses the "View Order" button to view order details. 5. System retrieves order details data from order table in database. 6. System displays order details data in the "Order Details" page. 7. Use case ends. **[B2: Reorder Order]** 1. Use case starts when the customer is at the "Order History" page. 2. Customer presses the "Re-Order" button to reorder an order. 3. System retrieves order data and adds it to the order cart. 4. System redirects the customer to the "Order Cart" page. 5. Basic flow continues at Use Case of Place Order Module, Basic Flow, B1: Place Order, Step 9. 6. Use case ends.

Alternative Flow	**[A1: No Order History]** 1. System unable to retrieve data as the customer does not place any order since they registered. 2. System displays “No Order History” in the “Order History” page. 3. Use case ends.
Exception Flow	4.
Post-Conditions	NONE
Rules	NONE
Constraints	Only the customer will be able to perform this use case.

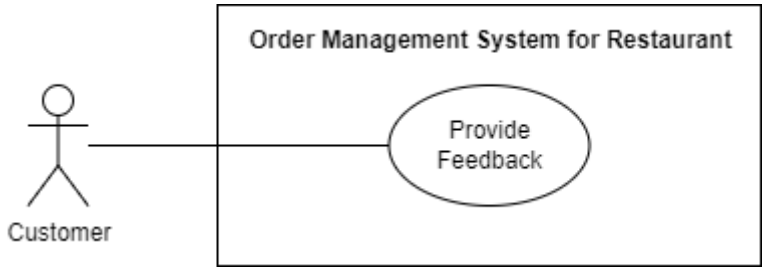

Figure 2.5 Use Case Diagram of Provide Feedback Module

Table 2.4 Use Case Description of Provide Feedback Module

Use Case ID	UC004
Brief Description	This use case describes the process of the customers providing feedback to the restaurant.
Actor	Customer
Pre-Conditions	1. Customer signed in to the system. 2. Customer's device is connected to the internet.
Basic Flow	**[B1: Provide Feedback]** 1. Use case starts when the customer presses the "Feedback" button. 2. System retrieves feedback form. 3. System displays feedback form in the "Feedback" page. 4. Customer selects rating and fills their comment. 5. Customer presses the "Submit" button to submit the feedback. 6. System saves data into feedback table in database. 7. Use case ends.
Alternative Flow	NONE
Exception Flow	NONE
Post-Conditions	Customer will be redirected to the home page.
Rules	NONE
Constraints	Only the customer will be able to provide feedback to the restaurant.

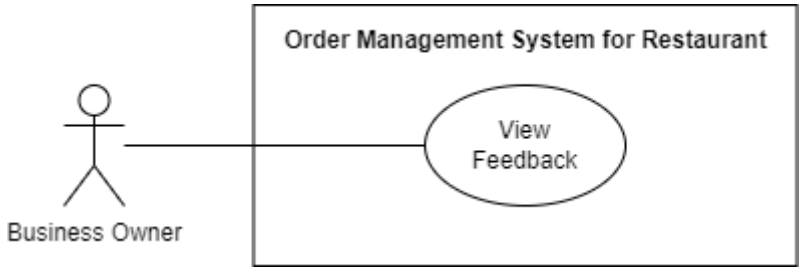

Figure 2.6 Use Case Diagram of View Feedback Module

Table 2.5 Use Case Description of View Feedback Module

Use Case ID	UC005
Brief Description	This use case describes the process of the business owner to view customers' feedback.
Actor	Business Owners
Pre-Conditions	1. Business owners signed in to the system. 2. Business owners' device is connected to the internet.
Basic Flow	**[B1: View Feedback]** 1. Use case starts when the business owners press the "View Feedback" button. 2. System retrieves feedback data from the feedback database. **[A1: No Feedback Data]** 3. System displays feedback data in the "View Feedback" page. **[A2: Filter Feedback Data]** 4. Use case ends.
Alternative Flow	**[A1: No Feedback Data]** 1. System unable to retrieve customers feedback data. 2. System displays "No Feedback Submitted" in "View Feedback" page. 3. Use case ends. **[A2: Filter Feedback Data]** 1. Business owners select date input to filter feedback data. 2. System retrieves and displays filtered feedback data. **[A1: No Feedback Data]** 3. Use case ends
Exception Flow	NONE
Post-Conditions	NONE
Rules	NONE
Constraints	Only the business owners will be able to view customers' feedback.

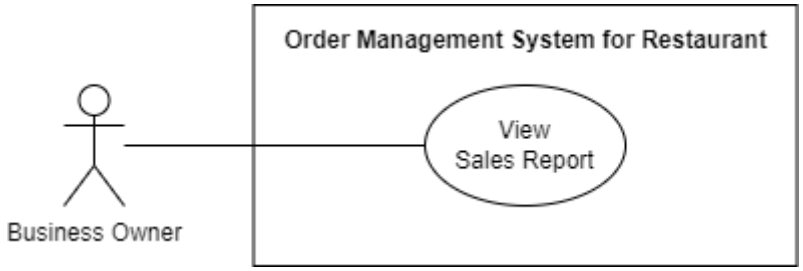

Figure 2.7 Use Case Diagram of View Sales Report Module

Table 2.6 Use Case Description of View Sales Report Module

Use Case ID	UC006
Brief Description	This use case describes the process of the business owner viewing a sales report.
Actor	Business Owners
Pre-Conditions	1. Business owners signed in to the system. 2. Business owners' device is connected to the internet.
Basic Flow	**[B1: View Sales Report]** 1. Use case starts when the business owners press the "Sales Report" button. 2. System retrieves sales report data from the report database. **[A1: No Report Data]** 3. System displays sales report data in the "Sales Report" page. 4. Business owners press the "View Detail" button to view a particular sales report. 5. System retrieves sales report details data from report detail database. 6. System displays sales report details data in the "Sales Report Details" page. 7. Use case ends.
Alternative Flow	**[A1: No Report Data]** 1. System unable to retrieve sales report data. 2. System displays "No Sales Report" in "Sales Report" page. 3. Use case ends.
Exception Flow	NONE
Post-Conditions	NONE
Rules	NONE
Constraints	Only the business owners will be able to view the sales report.

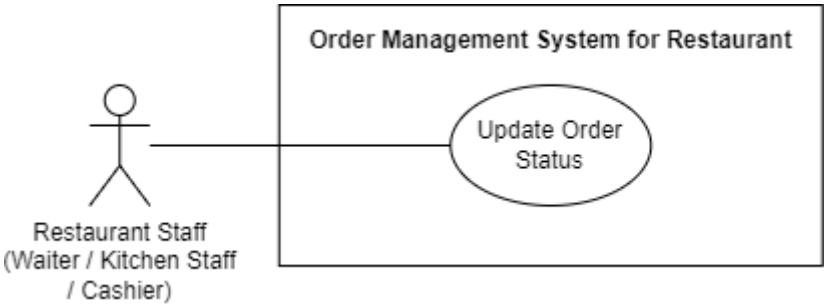

Figure 2.8 Use Case Diagram of Update Order Status Module

Table 2.7 Use Case Description of Update Order Status Module

Use Case ID	UC007
Brief Description	This use case describes the process of the restaurant staff updating order status.
Actor	Restaurant Staff (Waiter, Kitchen Staff, Cashier)
Pre-Conditions	1. Restaurant staff signed in to the system. 2. Restaurant staff's device is connected to the internet.
Basic Flow	**[B1: Update Order Status]** 1. Use case starts when restaurant staff signed in to the system. 2. System retrieves order data from the order database. 3. System displays order data in the "View Order" page. 4. Restaurant staff press the "Update Status" button to update the order status. 5. System prompts confirmation message. **[A1: Press No]** 6. Restaurant staff press the "Confirm" button to update order status. 7. System updates order status in order database. 8. Use case ends.
Alternative Flow	NONE
Exception Flow	**[A1: Press No]** 1. Alternative flow continues at Use Case of Update Order Status Module, Basic Flow, B1: Update Order Status, Step 3. 2. Use case ends.
Post-Conditions	NONE
Rules	**[R1: Order Data Displayed]** 1. Only order data with the status "On Queue" or "Preparing" will be displayed to the kitchen staff. 2. Only order data with the status "To be Serve" will be displayed to the waiter. 3. Only order data with the status "Delivered" will be displayed to the cashier.

	[R2: Order Status Updated] 1. Kitchen staff will update the order with “On Queue” status to “Preparing” status. 2. Kitchen staff will update the order with “Preparing” status to “To be Serve” status. 3. Waiter will update the order with “To be Serve” status to “Delivered” status. 4. Cashier will update the order with “Delivered” status to “Completed” status.
Constraints	• Only the restaurant staff will be able to view orders and update order status. • The kitchen staff will update the order status from “On Queue” to “Preparing” 5 minutes after the order is placed.

2.2 ACTIVITY DIAGRAM

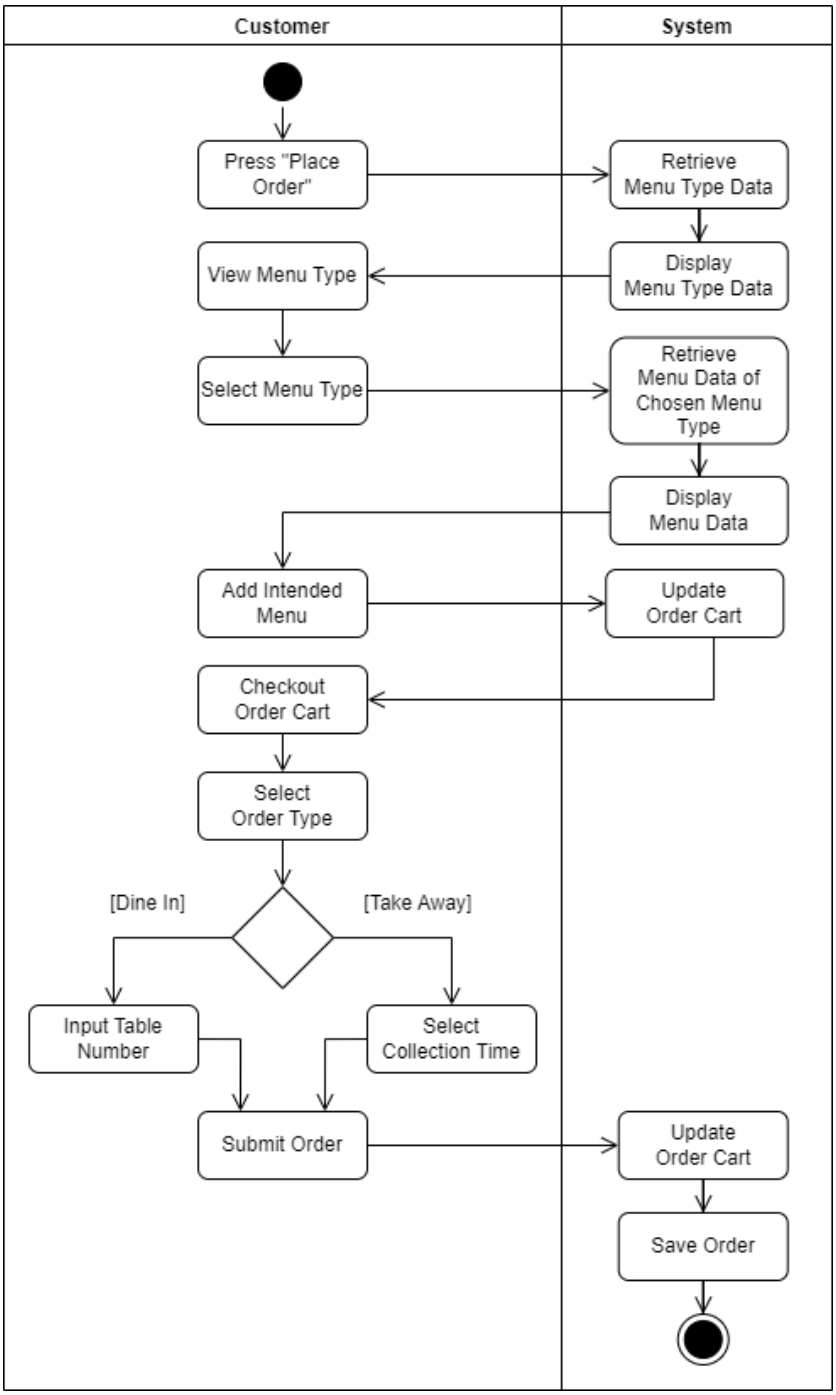

Figure 2.9 Activity Diagram of Place Order Module

Figure 2.9 shows the activity diagram of a customer placing an order. Firstly, the customer will press the “Place Order” button in the homepage to access the “Menu Type” page. After that, the system will retrieve menu type data from the database and display the data in the “Menu Type” page. Next, the customer will press any type of the menu and access to the “Menu List” page of the chosen menu type. After that, system will retrieve and display the menu list of the chosen menu type. Customer can then view the menu displayed and press the “Add to Cart” button of their intended menu. The system will then update the customer’s order cart with the intended menu. Customer will then access their order cart and check out the items in the order cart. After that, the system will prompt the customer to select an order type which can be “Dine In” or “Take Away”. If the customer selects “Dine In”, the system will prompt the customer to input the table number while if the customer selects “Take Away”, the system will prompt the customer to select collection time. Lastly, the customer will press submit button to submit their order to the system and the system will save the order into the database and clear the customer’s order cart.

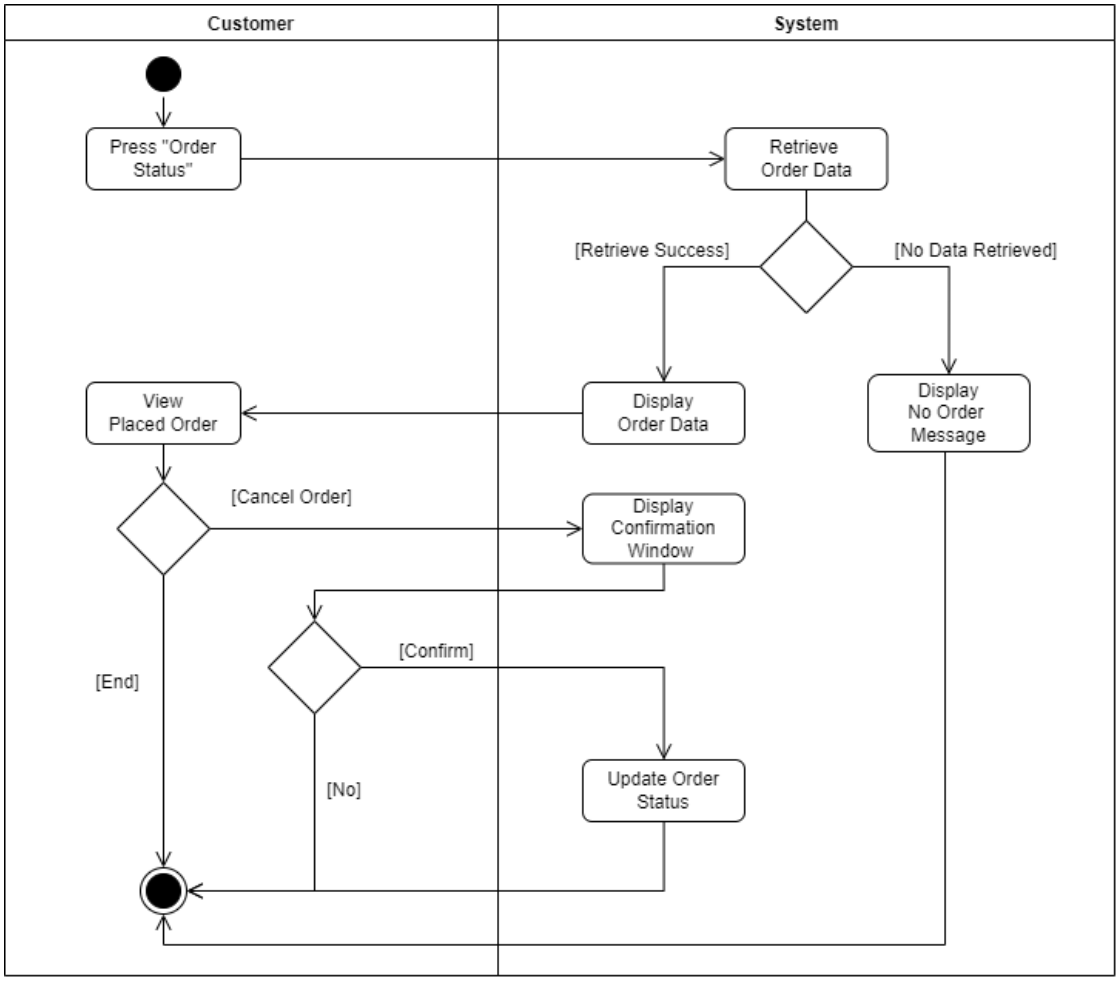

Figure 2.10 Activity Diagram of View Order Status Module

Figure 2.10 shows the activity diagram of a customer viewing their order status and cancelling their order. Firstly, the customer will press the “Order Status” button in the homepage to access the “Order Status” page. Next, the system will retrieve the order data of the customer and display it. Customer will then be able to view their placed order. If there is no order data for the customer, the system will display “No Order” message to the customer. The customer can cancel their order by pressing the cancel button. The system will pop up a confirmation window regarding the order cancellation. If the customer press “Confirm” button, then the order will be cancelled. In addition, the customer only can cancel their order if the order status is “On Queue”.

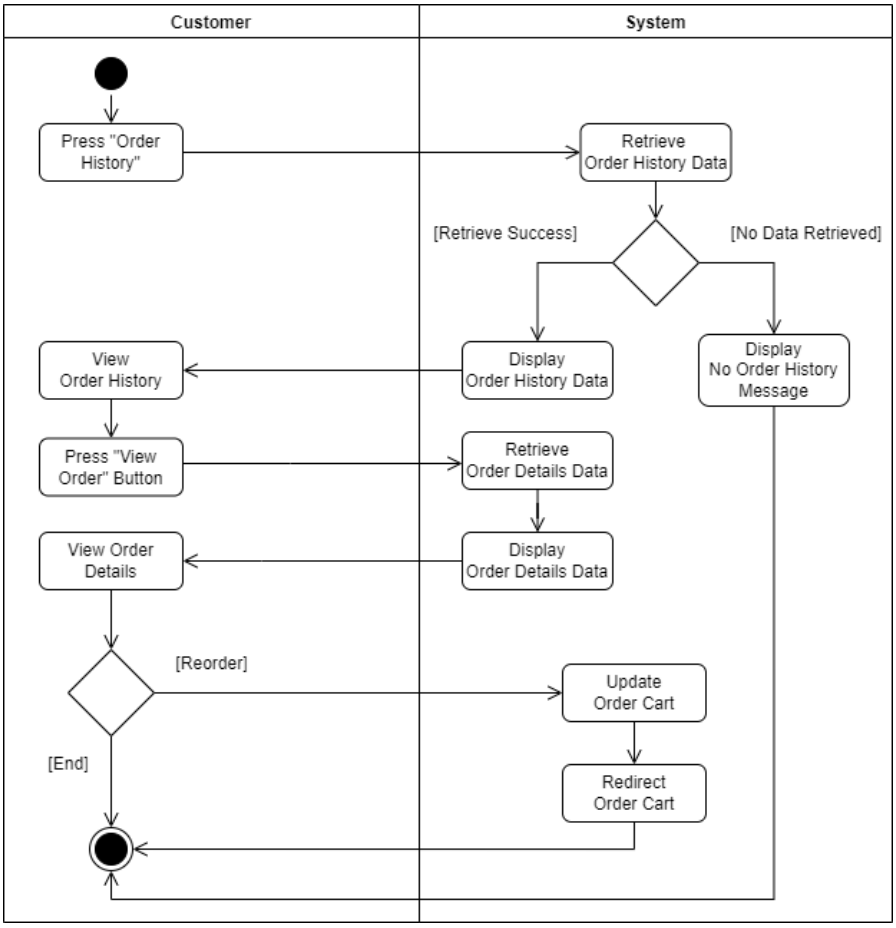

Figure 2.11 Activity Diagram of View Order History Module

Figure 2.11 shows the activity diagram of a customer viewing order history and reordering their previous order. Firstly, the customer will press the “Order History” button in the homepage to access the “Order History” page. Next, the system will retrieve the customer’s order history and display it in the “Order History” page. If there is no order history for the customer, the system will display “No Order History” message to the customer. After that, the customer can view the abstract order details, or they can also press the “View Order” button to view the details of the order. Furthermore, the customer can press the “Re-order” button to order again from the menu of the order view. After they press the button, the system will update the customer’s order cart and redirect them to the order cart page.

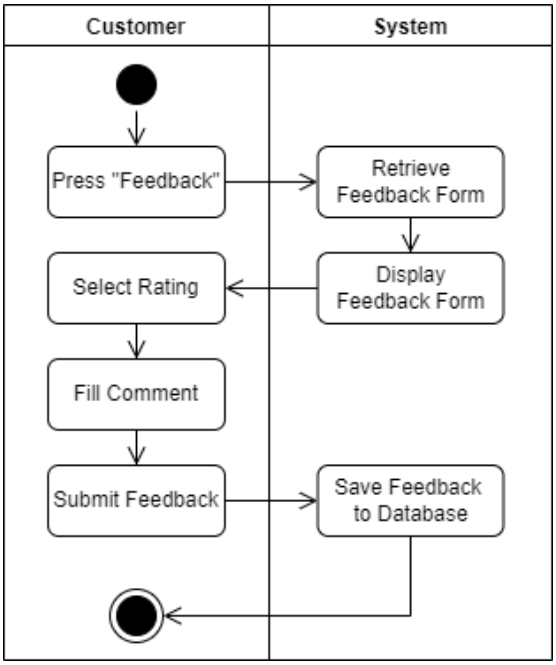

Figure 2.12 Activity Diagram of Provide Feedback Module

Figure 2.12 shows the activity diagram of a customer providing feedback to the restaurant. Firstly, the customer will press the “Feedback” button in the homepage to access the “Feedback Form” page. The system will then retrieve the feedback form and display it to the customer. Customer will be required to select their rating and fill in their comment if any. After that, the customer will press the “Submit” button to submit their feedback and the system will save feedback to the database.

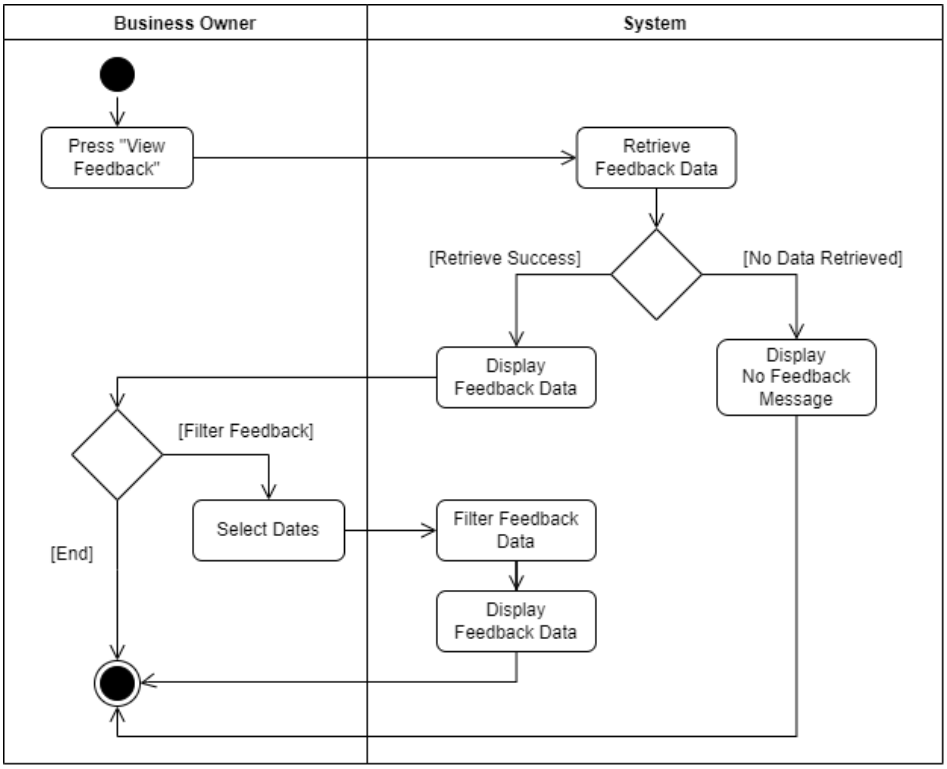

Figure 2.13 Activity Diagram of View Feedback Module

Figure 2.13 shows the activity diagram of the business owner viewing customers' feedback. Firstly, the business owner will press the "View Feedback" button in the homepage to access the "View Feedback" page. The system will then retrieve customers' feedback data from the database and display it to the business owner. If no feedback data retrieved, the system will display "No Feedback" message to the business owner. Besides, the business owner can also filter the feedback by selecting a range of dates of the feedback created. The system will then filter and display only the feedback data created within the range.

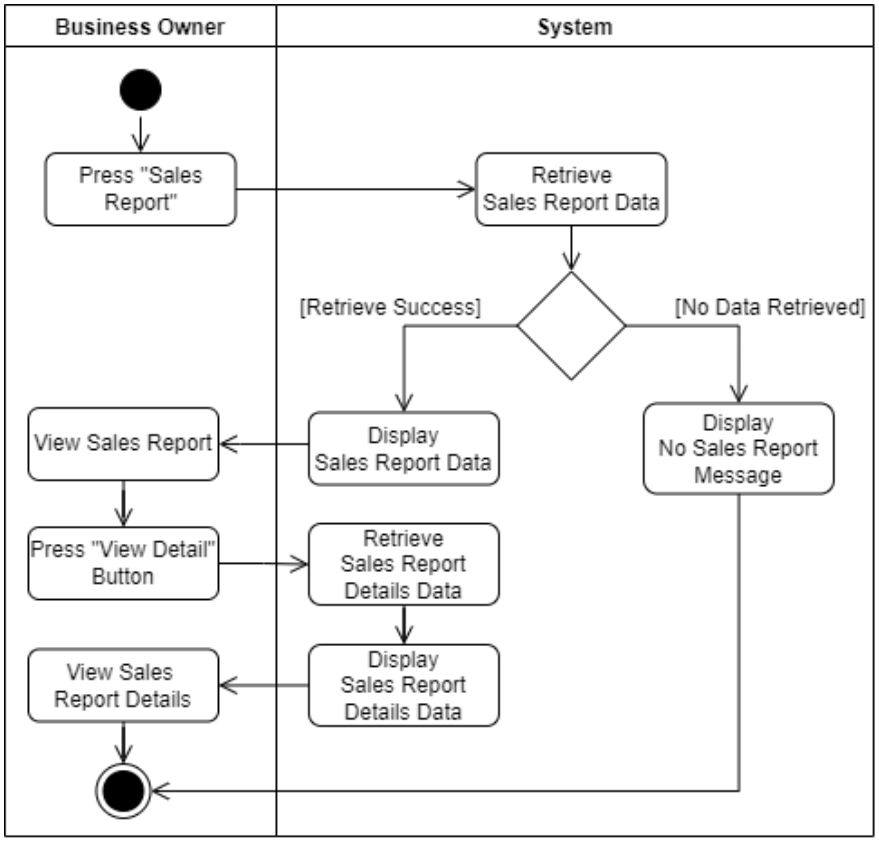

Figure 2.14 Activity Diagram of View Sales Report Module

Figure 2.14 shows the activity diagram of the business owner viewing the sales report. Firstly, the business owner will press the “Sales Report” button in the homepage to access the “Sales Report List” page. The system will retrieve sales report data from the database and display it. The business owner will be able to view the abstract sales report in the “Sales Report List” page. If the system there is no sales report data in database, the system will display “No Sales Report” message to the business owner. The business owner can also press the “View Detail” button to view the full details of the sales report. After the “View Detail” button is pressed, the system will retrieve the sales report details and display them in the “Report Detail” page.

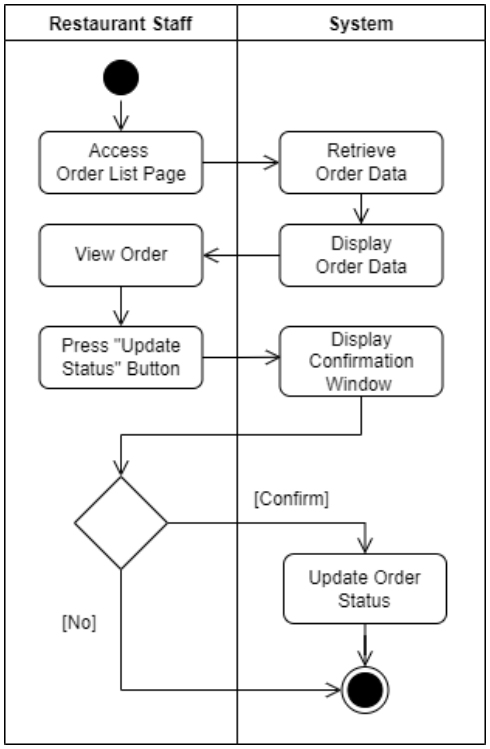

Figure 2.15 Activity Diagram of Update Order Status Module

Figure 2.15 shows the activity diagram of the restaurant staff updating order status. After the restaurant staff sign in to the system, the system will redirect them to the "Order List" page. When they access the "Order List" page, the system will retrieve order data from the database and display it to the restaurant staff. The restaurant staff will then press the "Update Status" button when the constraints are fulfilled. When the "Update Status" button was pressed, the system will pop up the confirmation window. The staff are required to press "Confirm" button to update the order status. For example, the kitchen staff will only update order status from "On Queue" to "Preparing" 5 minutes after the order is placed and they will also update order status from "Preparing" to "To Be Serve" when they have finished preparing the meals. Moreover, the waiter will update the order status from "To Be Serve" to "Delivered" once they have delivered the meal to the customer. Lastly, the cashier will update the order status from "Delivered" to "Completed" once they received payment from the customer. After the restaurant staff press the "Update Status" button, the system will then update the order status.

CHAPTER 3

3.1 INTERFACE DESIGN

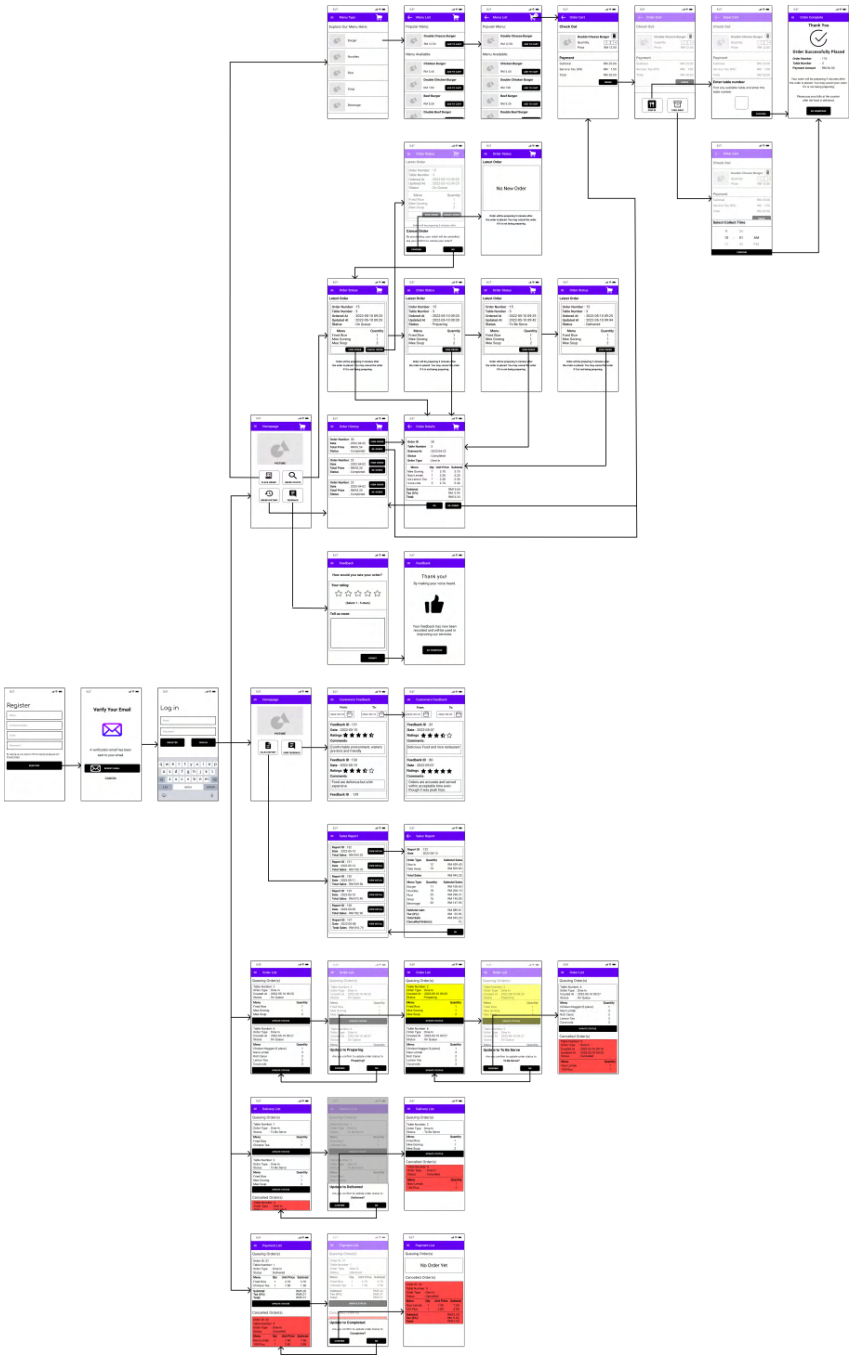

Figure 3.1 Storyboard of Proposed System.

Figure 3.1 shows the storyboard of the proposed system which depicts the flow of interfaces of all modules which are place order module, view order status module, view order history module, provide feedback module, view feedback module, view sales report module and update status module.

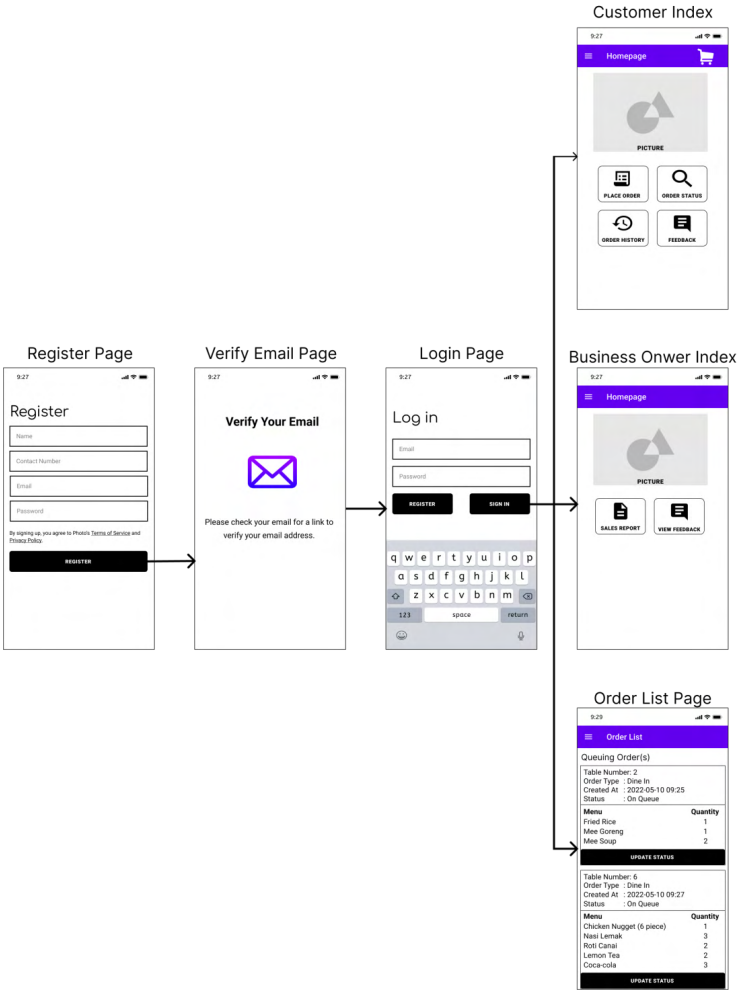

Figure 3.2 Flow of Interfaces for Register and Login

Figure 3.2 shows the flow of interfaces when the user login. If the user does not own an account, they need to go to the “Register” page to register their account. After submitting their information, the system moves to the “Verify Email” page to prompt the user to verify their email address. After the user verified their email address, the system will redirect to the “Verify Success” page to indicate the account has been verified. Next, the user will be redirected to the “Login” page, the user is required to fill in their login credentials. When the login is successful, the user will be redirected to a different homepage according to their user type. For instance, the customer will be redirected to “Customer Homepage”, the business owner will be redirected to “Business Owner Homepage” while the restaurant staff will be redirected to the “Order List” page.

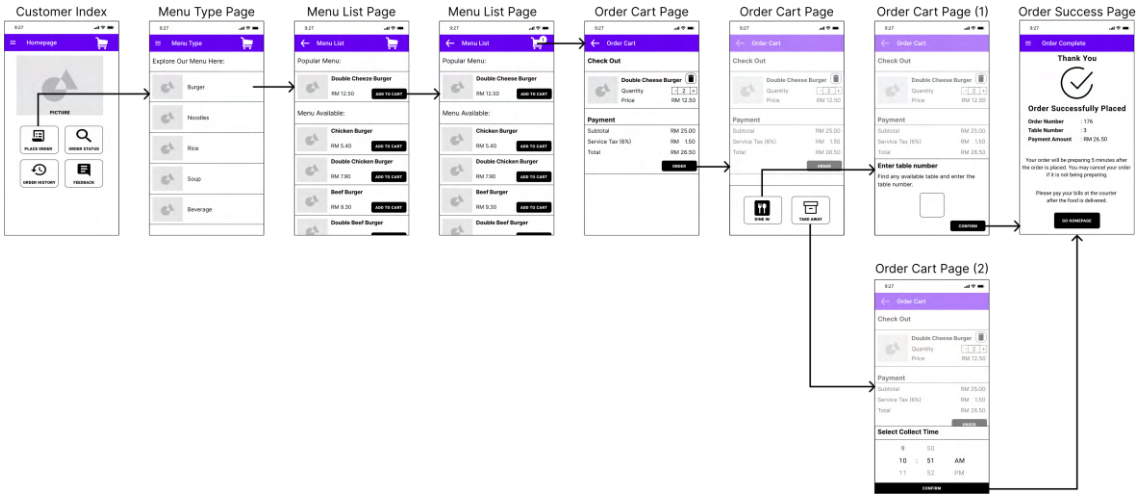

Figure 3.3 Flow of Interfaces for Place Order Module

Figure 3.3 shows the flow of interfaces when a customer places an order. Firstly, the customer will press the “Place Order” button to access the “Menu Type” page. After that, the customer will have to select their intended menu type and they will be redirected to the “Menu List” page to view the menu available under the selected menu type. Next, the customer will press the “Add to Cart” button of their intended menu and the system will update the order cart. As shown in the figure, there is an integer “1” on top of the order cart icon which indicates there is an item in the order cart. The customer will then have to press the order cart to check out their order cart as an order. The system will redirect the customer or order cart page with the details of the menu added. The customer will press the “Order” button to proceed with their order. After pressing the “Order” button, the system will prompt the customer to select their order type. If the customer selects “Dine In”, the system will prompt the customer to enter the table number. On the other hand, if the customer selects “Take Away”, the system will prompt the customer to select a collection time. Lastly, the customer will press the “Confirm” button to place their order. The system saves the order into the database and redirects to the “Order Success” page which shows the information of the order placed.

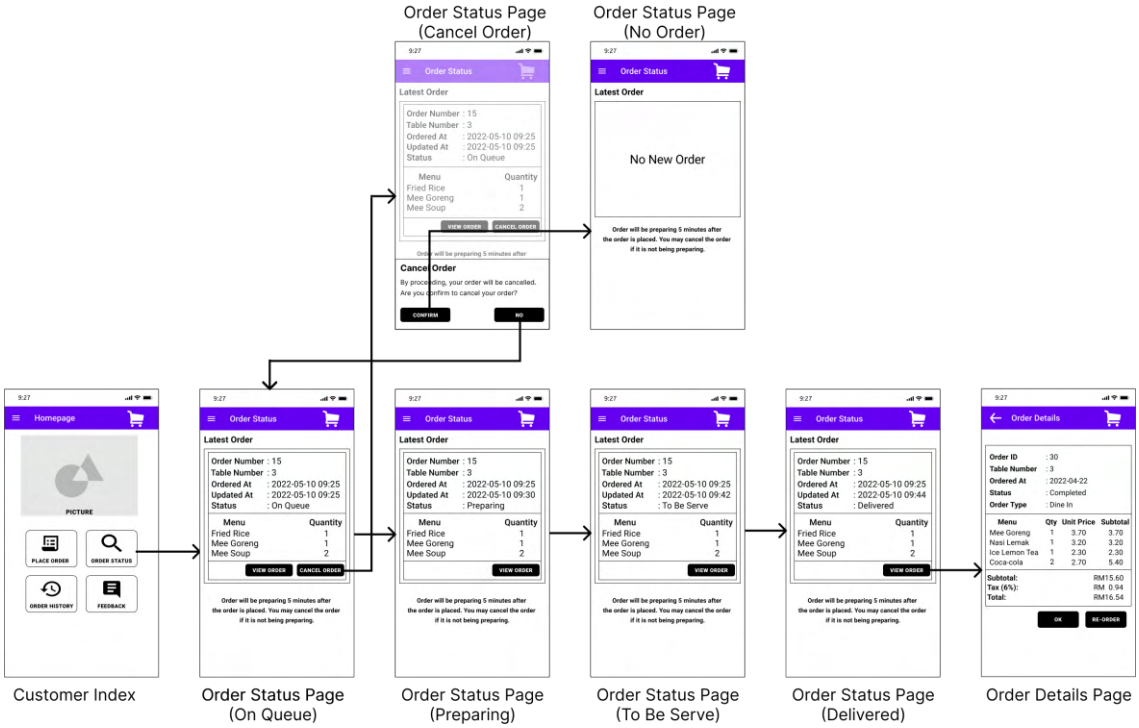

Figure 3.4 Flow of Interfaces for View Order Status Module

Figure 3.4 shows the flow of interfaces when a customer views their order status. Firstly, the customer will press the “Order Status” button to access the “Order Status Page”. After that, if the order placed is with the status “On Queue”, the customer is able to cancel their order. However, if the order status is not “On Queue”, the “Cancel Order” button will not be displayed to the customer. When the customer presses the “Cancel Order” button, the system will prompt a confirmation message to the customer. If the customer presses the “Confirm” button, the system will proceed with the cancel request and be redirected to the “Order Status” page with a “No New Order” message. Furthermore, if the customer presses the “No” button to cancel the action and they will be redirected to the “Order Status” page. Besides, the customer can also press the “View Order” button to view the order details of the order made.

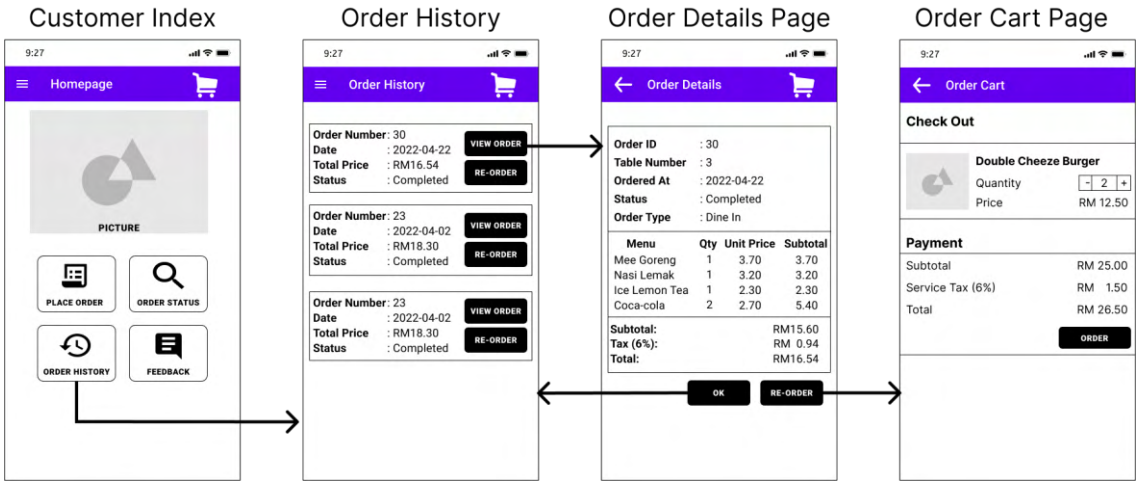

Figure 3.5 Flow of Interfaces for View Order History Module

Figure 3.5 shows the flow of interfaces when a customer viewing order history. The system will redirect the customer to the “Order History” page when they press the “Order History” button in “Customer Homepage”. The customer can check the details of the order by pressing the “View Order” button. The system will redirect the customer to the “Order Details” page. The customer can also perform a reordering action by pressing the “Re-Order” button. The system will add the order details to the order cart and redirect the customer to the “Order Cart” page to perform reordering.

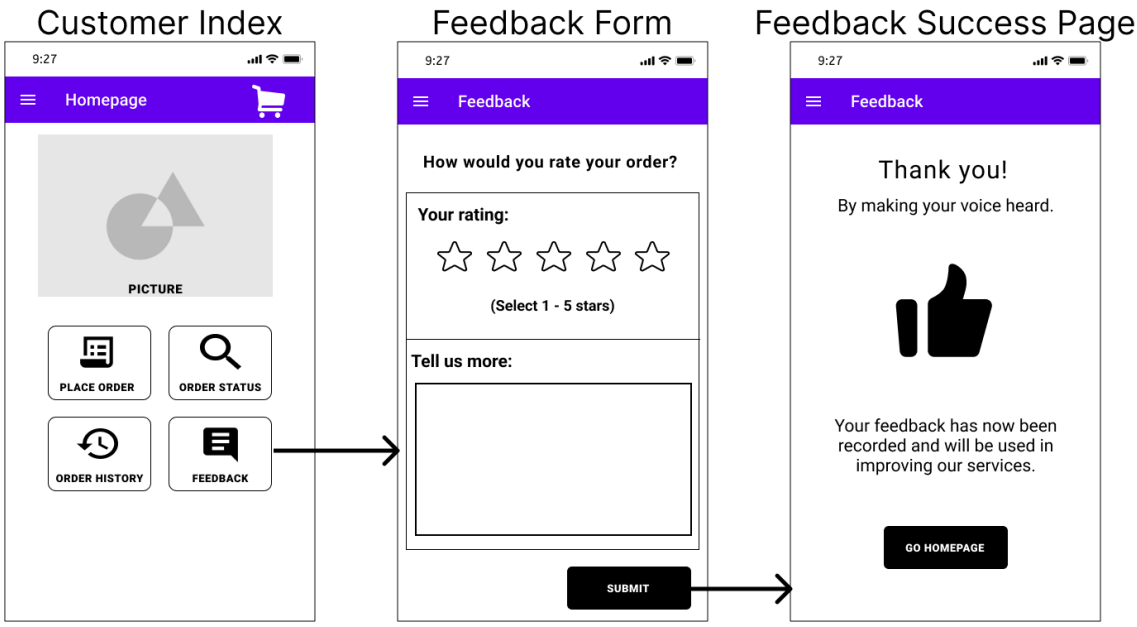

Figure 3.6 Flow of Interfaces for Provide Feedback Module

Figure 3.6 shows the flow of interfaces when a customer provides feedback. Firstly, the customer needs to press the “Feedback” button in “Customer Homepage”. The system will then display the “Feedback Form” page to the customer. The customer is required to select their rating and fill in the comments if any. After the customer press the “Submit” button, the system will save the feedback into the database and redirect them to the “Feedback Success” page.

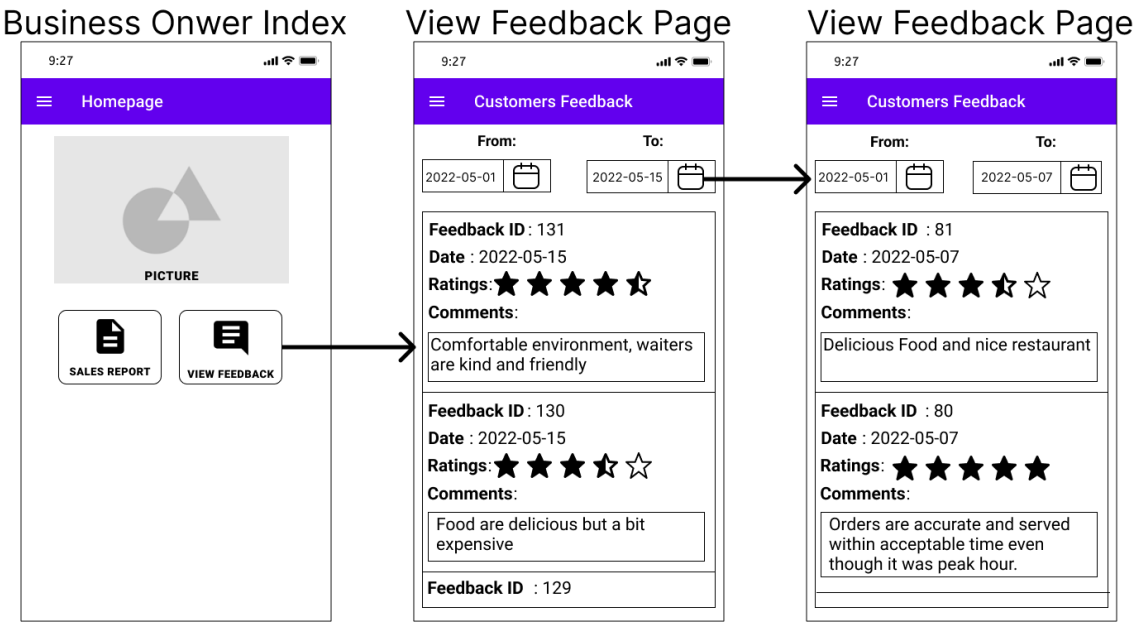

Figure 3.7 Flow of Interfaces for View Feedback Module

Figure 3.7 shows the flow of interfaces for the business owner to view feedback from customers. The business owner needs to press the “View Feedback” button in the “Business Owner Homepage” page to access the “View Feedback” page. After that, the system will retrieve the feedback data from the database and display it in the “View Feedback” page. The business owner can also select a range of dates to filter the feedback data.

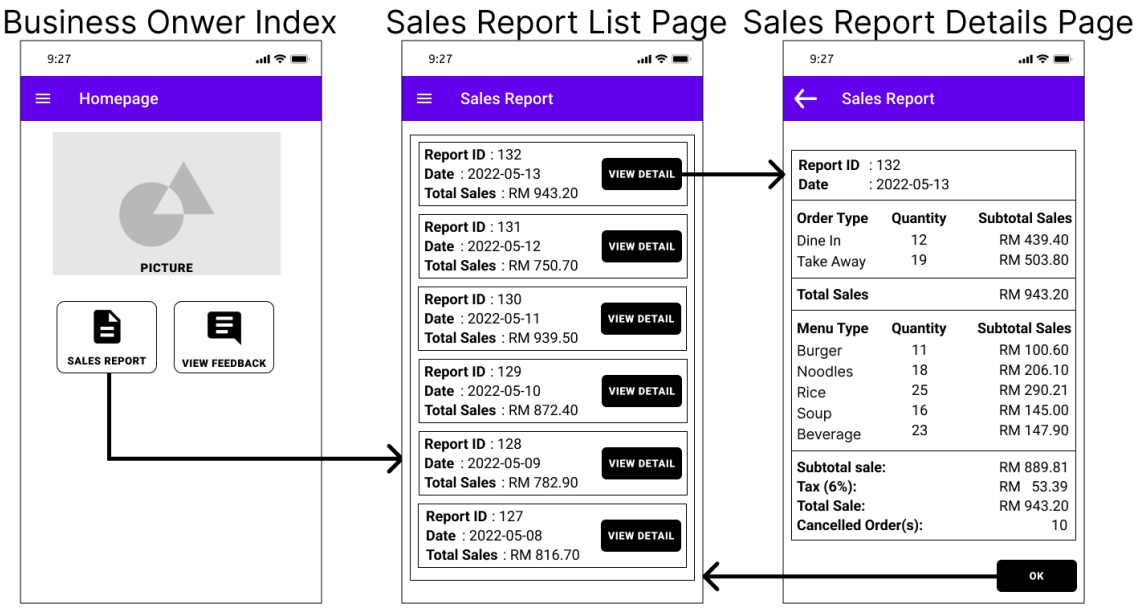

Figure 3.8 Flow of Interfaces for View Sales Report Module

Figure 3.8 shows the flow of interfaces for the business owner to the view sales report. Firstly, the business owner needs to press the “Sales Report” button in the “Business Owner Homepage”. After that, the system will retrieve sales report data from the database, display and redirect the business owner to the “Sales Report List” page. The business owner can press the “View Detail” button to view the details of the sales report.

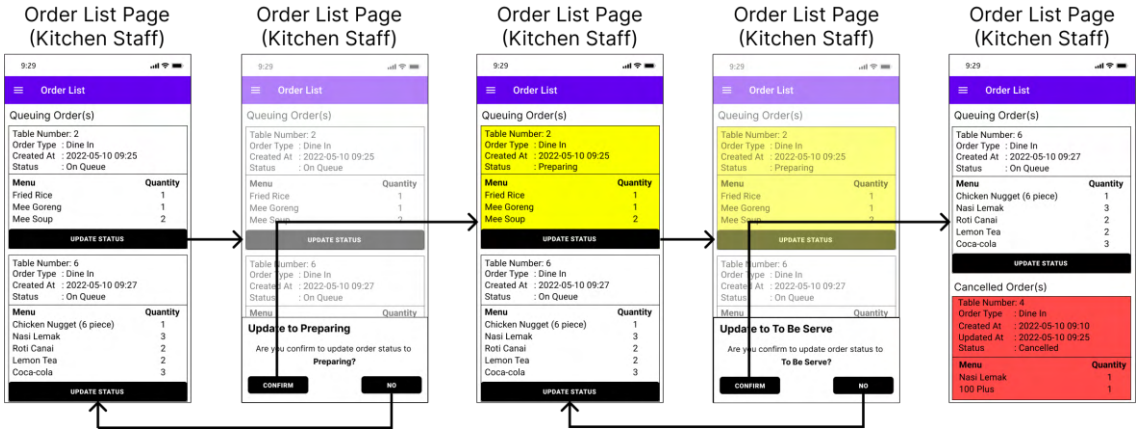

Figure 3.9 Flow of Interfaces for Update Order Status Module (Kitchen Staff)

Figure 3.9 shows the flow of interfaces for kitchen staff to update the order status. After the kitchen staff sign in to the system, they will be redirected to the “Order List” page. The kitchen staff will be able to view the details of customers’ orders. The kitchen staff will press the “Update Status” button to update the order status and they will only update the order with status “On Queue” to “Preparing” 5 minutes after the order is placed. Furthermore, the kitchen staff will also be responsible to update the order status from “Preparing” to “To Be Serve” after they finished preparing the meals of the order. Only the order with status “On Queue” and “Preparing” will be displayed to the kitchen staff at the “Queueing Order(s)” section in the “Order List” page. The kitchen staff will also be able to view today’s cancelled orders at the “Cancelled Order(s)” section.

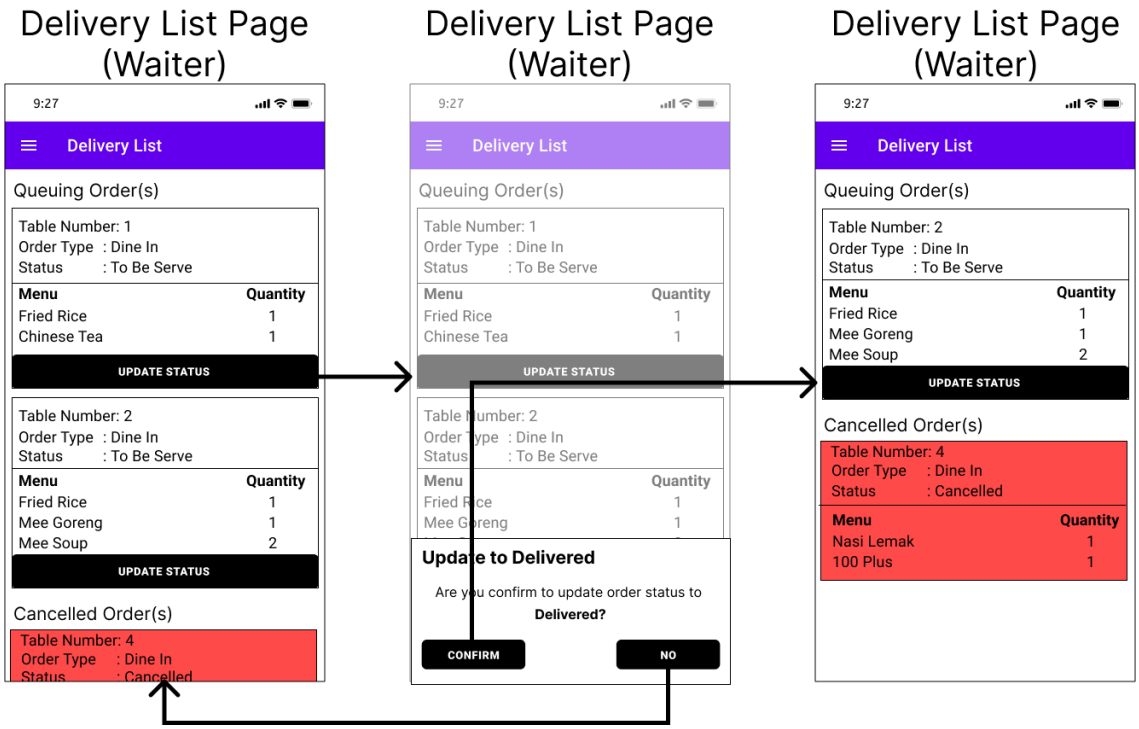

Figure 3.10 Flow of Interfaces for Update Order Status Module (Waiter)

Figure 3.10 shows the flow of interfaces for the waiter to update the order status. The waiter will be redirected to the "Delivery List" page after they sign in. The waiter can view the details of customers' orders and press the "Update Status" button to update the order status. The waiter will update the order with status "To Be Serve" to "Delivered" once they have delivered the meal to the customer. Only the order with status "To Be Serve" will be displayed at the "Queueing Order(s)" section in the "Delivery List" page of the waiter. The waiter will also be able to view today's cancelled orders at the "Cancelled Order(s)" section.

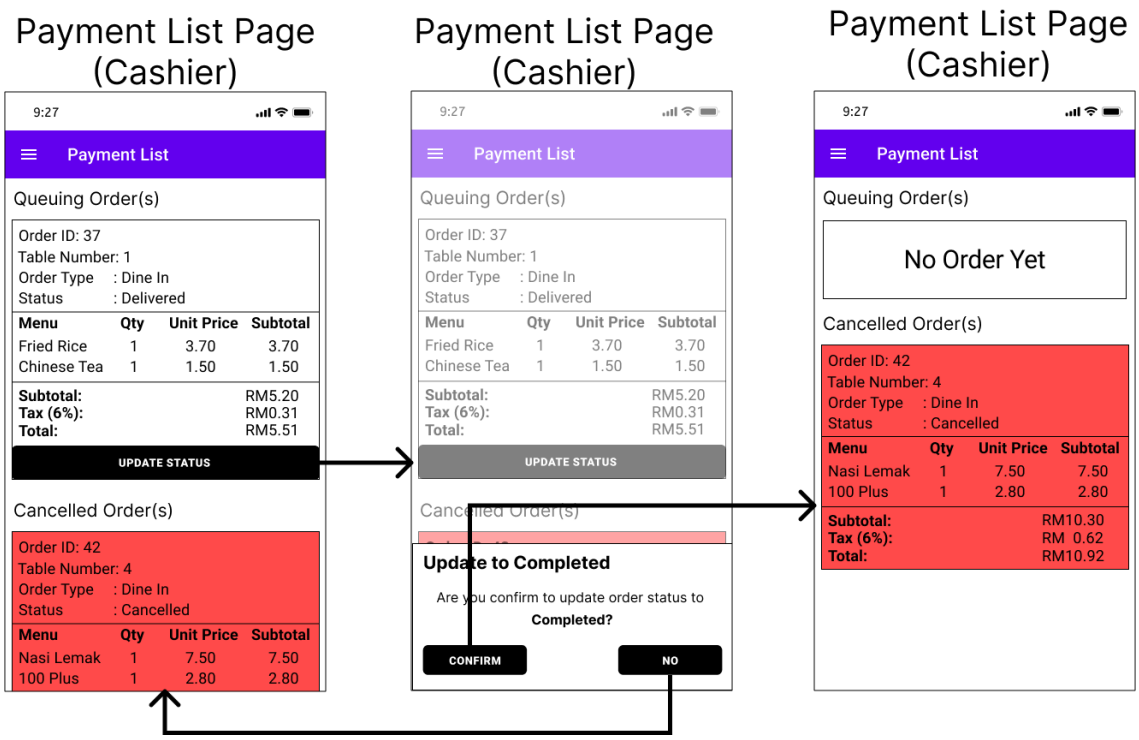

Figure 3.11 Flow of Interfaces for Update Order Status Module (Cashier)

Figure 3.11 shows the flow of interfaces for the cashier to update the order status. The cashier will be able to view the details of customers' orders in the "Payment List" page. They will press the "Update Status" button to update the order status from "Delivered" to "Completed" once they have received payment from the customer. Only orders with the status of "Delivered" will be displayed to the cashier at the "Queueing Order(s)" section in the "Payment List" page. The cashier will also be able to view today's cancelled orders at the "Cancelled Order(s)" section.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

Hardware

Table 3.1 Hardware Requirements

Hardware	Descriptions	Requirements
Laptop / Desktop	To run software for developing the order management system for restaurant.	8GB RAM or above and Core i7 8th Gen or above
Android phone	To test the application run as expected.	Android 7.0 or above and 3GB RAM or above with internet connection

Software

Table 3.2 Software Requirements

Software	Descriptions	Version
Android Studio	To develop the order management system for restaurant.	Bumblebee 2021.1.1 Patch 3
Firebase	The database used by the system.	Not Applicable
Flutter	The framework used to develop the order management system for restaurant.	3.0.0
Operating System	To run software for developing the order management system for restaurant.	Window 10
draw.io	To create the context diagram, use case diagram, activity diagram, and entity relationship diagram.	18.1.2
Figma	To create the prototype (storyboard).	Not Applicable

APPENDIX B
SYSTEM DESIGN DOCUMENT (SDD)

For Appendices Heading, use TITLE AT ROMAN PAGES style.

2022

SOFTWARE DESIGN DESCRIPTION (SDD)

[ORDER MANAGEMENT SYSTEM FOR RESTAURANT]

DOCUMENT APPROVAL

	Name	Date
Authenticated by: _____ Name	Tan Chee Kin	
Approved by: _____ Client		

Software : Microsoft Office

Archiving Place : Google Drive

TABLE OF CONTENT

LIST OF FIGURES

LSIT OF TABLES

LIST OF APPENDICES

CHAPTER 1

1.1 PROJECT DESCRIPTION

Order Management System for Restaurant is a mobile application that developed to help the restaurant business owner to digitalize the business process, improve customers' experience as well as reduce operating expenses of the business. It facilitates the communication between the customer, waiters, cashier, and the kitchen staff and ensures a smoother workflow at a lower error rate compared to the traditional method. The proposed system has seven modules which are Place Order, View Order Status, View Order History, Provide Feedback, View Feedback, View Sales Report and Update Order Status.

- Place Order Module is the module that allows the customer to place their order via the system.
- View Order Status Module is the module that allows the customer to keep track with the status of their placed order. The customer can also cancel their placed order here if the order fulfilled the constraints.
- View Order History Module is the module that allows the customer to view their order history and perform a reorder action if they want to.
- Provide Feedback Module is the module that provide the customer with the ability to provide feedback to the restaurant.
- View Feedback Module is the module that provide the business owner with the ability to view the feedback collection from customers. The business owner can also select a range of dates to filter the feedback data.
- View Sales Report Module is the module that allows the business owner to view sales report of the restaurant.

- Update Order Status Module is the module that provide the restaurant staff which are the kitchen staff, the waiter and the cashier to view customers' orders and update the order status after the constraints is fulfilled.

1.2 SYSTEM IDENTIFICATION

The system uses the following convention:

System Identification Number: SDD-OMSR-V01-22

Table 1.1 System Identification Abbreviations

SDD	Software Design Document
OMSR	Order Management System for Restaurant
V01	Version 1
22	Year 2022

1.3 ARCHITECTURE / BLUE PRINT

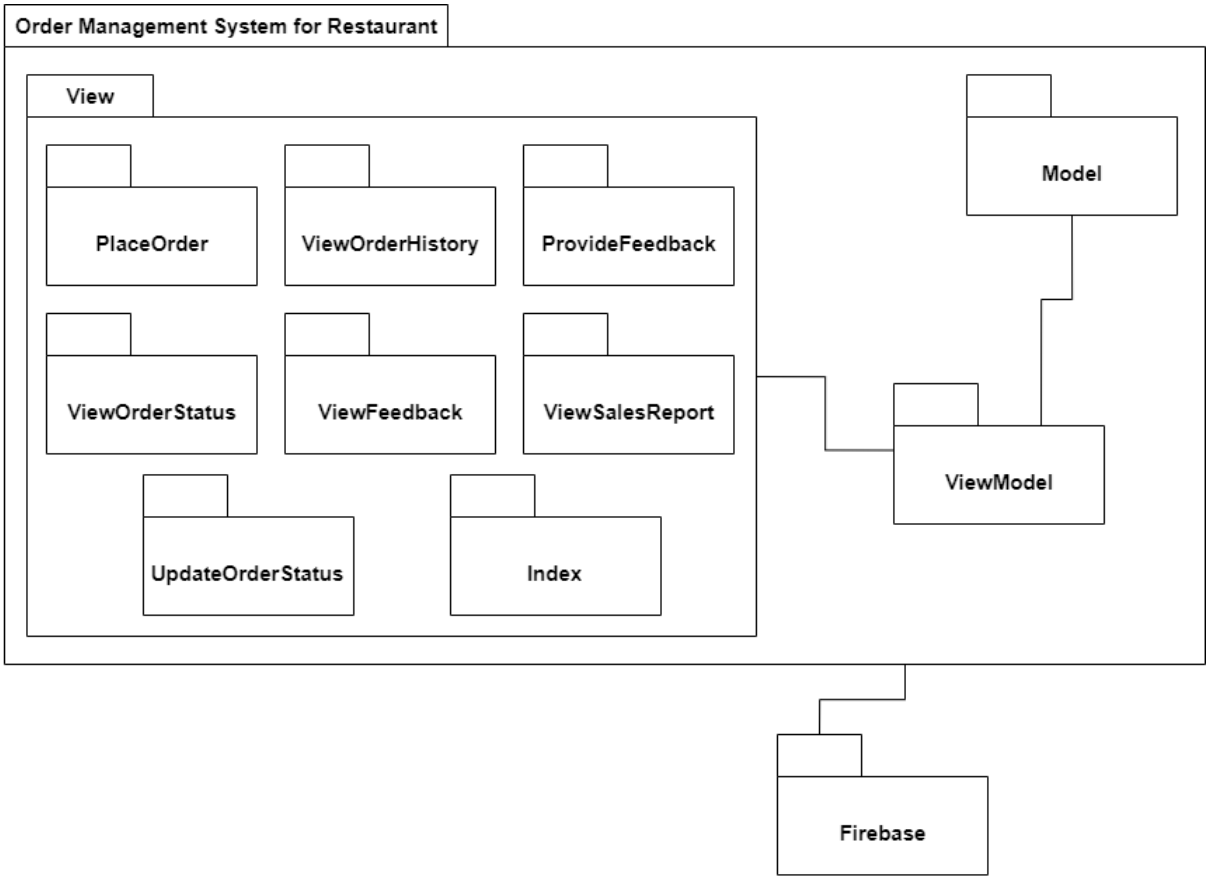

Figure 1.1 General Architecture of Proposed System

Figure 1.1 shows the General Architecture of Proposed System. The architecture used in Order Management System for Restaurant is Model-View-ViewModel (MVVM). This architecture mainly contains three core components which are the model, the view and the view model. The model is the object that hold data and perform data and activity encapsulation of application domain. The view is the interfaces that displayed to the user while the view model is the component that placed between view and model as the mediator. Firebase is the middleware that used as the database.

1.4 ARCHITECTURE / BLUEPRINT DESCRIPTION

1.4.1 Application Layer

1.4.1.1 Place Order

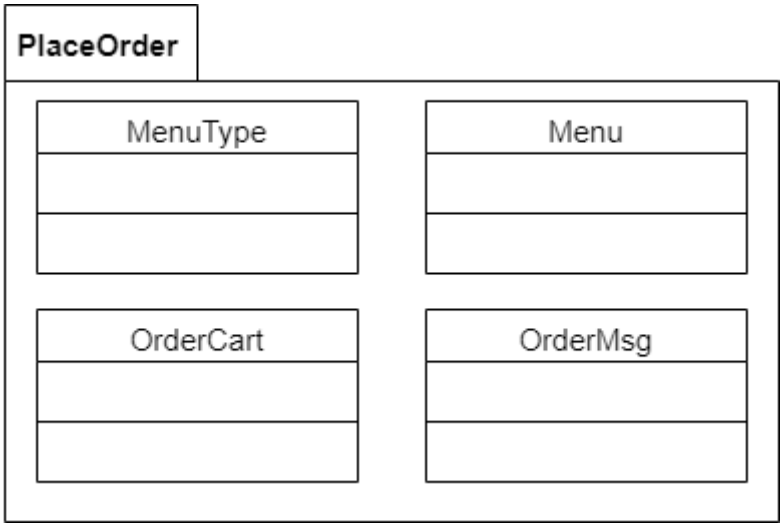

Figure 1.2 Application Layer of Place Order

Table 1.2 Interfaces of Place Order

Class Name	Description
MenuType	This class allows the customer to view and select their intended menu type.
Menu	This class allows the customer to view and select their intended menu under the menu type selected.
OrderCart	This class allows the customer to view list of intended menus added to the order cart and place an order based on the list.
OrderMsg	This class displays the abstract information of the order placed to the customer.

1.4.1.2 View Order Status

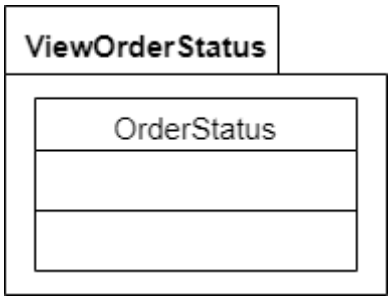

Figure 1.3 Application Layer of View Order Status

Table 1.3 Interfaces of View Order Status

Class Name	Description
OrderStatus	This class allows the customer to view the order status of their order. It also allows the customer to perform cancellation of order.

1.4.1.3 View Order History

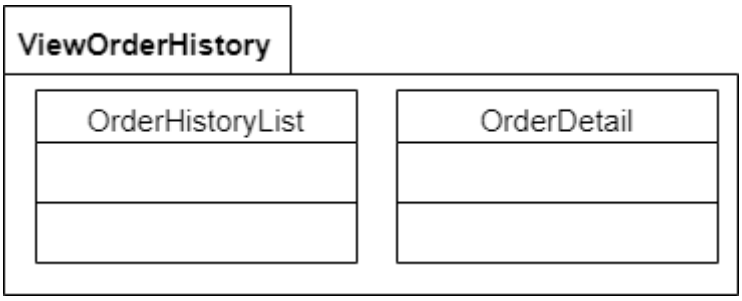

Figure 1.4 Application Layer of View Order History

Table 1.4 Interfaces of View Order History

Class Name	Description
OrderHistoryList	This class allows the customer to view the list of order placed.
OrderDetail	This class allows the customer to view the details of a particular placed order. It also allows the customer to perform reorder action.

1.4.1.4 Provide Feedback

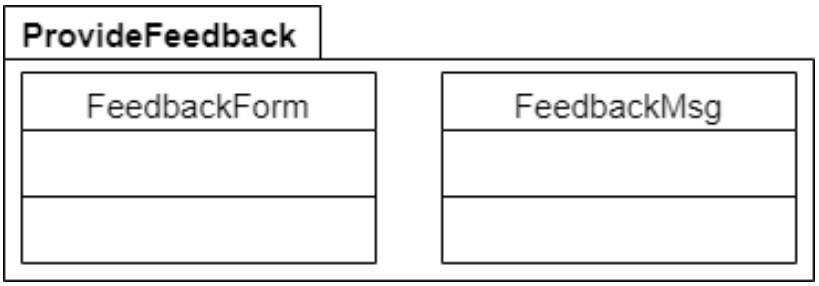

Figure 1.5 Application Layer of Provide Feedback

Table 1.5 Interfaces of Provide Feedback

Class Name	Description
FeedbackForm	This class allows the customer to fill and submit feedback to the restaurant.
FeedbackMsg	This class displays a thank you message to the customer for providing feedback to the restaurant.

1.4.1.5 View Feedback

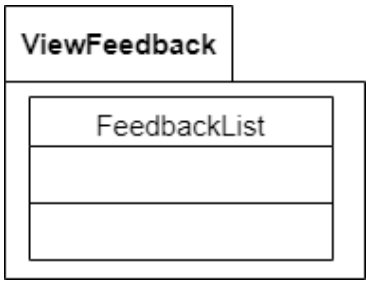

Figure 1.6 Application Layer of View Feedback

Table 1.6 Interfaces of View Feedback

Class Name	Description
FeedbackList	This class allows the business owner to view the feedback gathered from customers. It also allows the business owner to filter the feedback data according to the range of dates selected.

1.4.1.6 View Sales Report

Figure 1.7 Application Layer of View Sales Report

Table 1.7 Interfaces of View Sales Report

Class Name	Description
SalesReportList	This class allows the business owner to view abstract information of a list of sales report. The business owner can also select a particular sales report to view the details.
SalesReportDetail	This class allows the business owner to view the details of a particular sales report.

1.4.1.7 Update Order Status

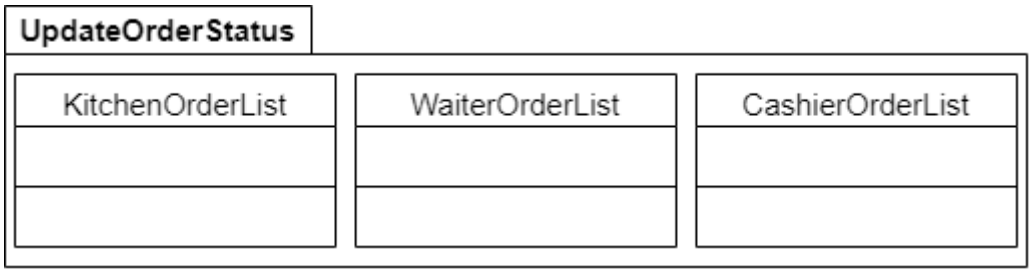

Figure 1.8 Application Layer of Update Order Status

Table 1.8 Interfaces of Update Order Status

Class Name	Description
KitchenOrderList	This class allows the kitchen staff to view the list of order that with a status of “On Queue” and “Preparing”. It also allows the kitchen staff to update the order status from “On Queue” to “Preparing” or from “Preparing” to “To Be Serve” when the constraints if fulfilled.
WaiterOrderList	This class allows the waiter to view the list of order with a status of “To Be Serve”. The waiter can update the order status from “To Be Serve” to “Delivered” after the constraints is fulfilled.

CashierOrderList	This class allows the cashier to view the list of order with a status of "Delivered". The cashier will update the status from "Delivered" to "Completed" after they received payment from the customer.

1.4.1.8 Index

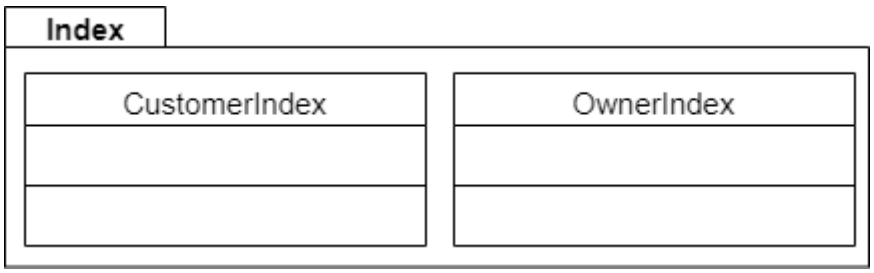

Figure 1.9 Application Layer of Index

Table 1.9 Interfaces of Index

Class Name	Description
CustomerIndex	This class allows the customer to choose to place order, view order status, view order history and provide feedback.
OwnerIndex	This class allows the business owner to choose to view sales report and view feedback gathered from customers.

1.4.2 Business Service Layer

1.4.2.1 ViewModel

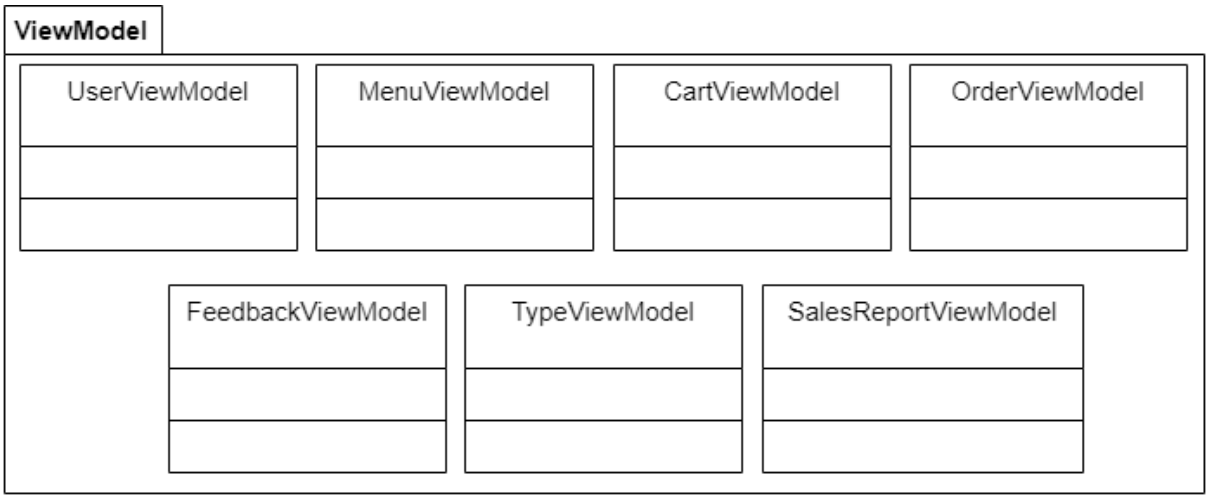

Figure 1.10 Business Service Layer for ViewModel

Table 1.10 Description of Business Service Layer for ViewModel

Class Name	Description
User ViewModel	ViewModel for managing and processing all operations related with UserData Model.
Menu ViewModel	ViewModel for managing and processing all operations related with MenuData Model.
Cart ViewModel	ViewModel for managing and processing all operations related with OrderCartData Model.
Order ViewModel	ViewModel for managing and processing all operations related with OrderData Model.
Feedback ViewModel	ViewModel for managing and processing all operations related with FeedbackData Model.
Type ViewModel	ViewModel for managing and processing all operations related with TypeData Model.
SalesReport ViewModel	ViewModel for managing and processing all operations related with SalesReportData Model and SalesReportDetData Model.

1.4.2.2 Model

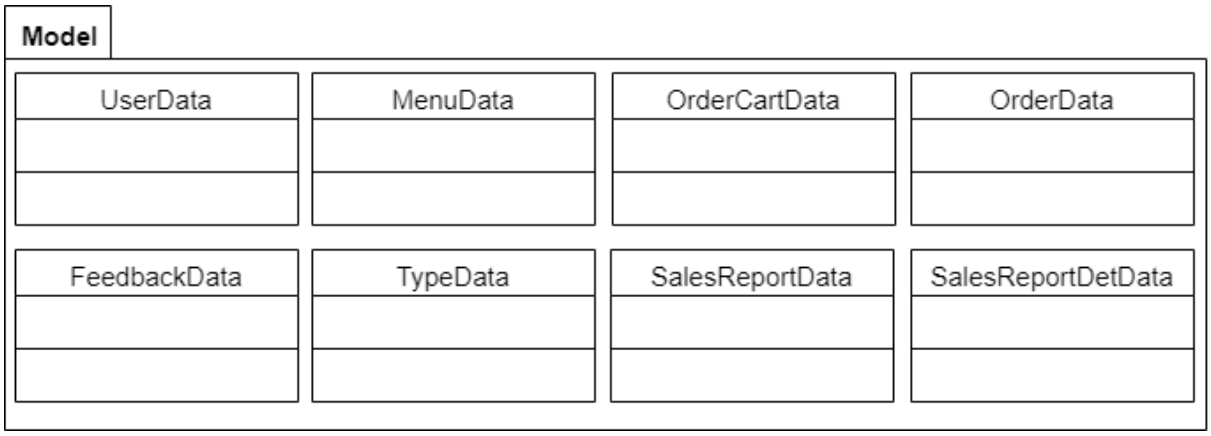

Figure 1.11 Business Service Layer for Model

Table 1.11 Description of Business Service Layer for Model

Class Name	Description
UserData Model	Model that executes and processes Firebase queries related with user data to the database.
MenuData Model	Model that executes and processes Firebase queries related with menu data to the database.
OrderCartData Model	Model that executes and processes Firebase queries related with order cart data to the database.
OrderData Model	Model that executes and processes Firebase queries related with order data to the database.
FeedbackData Model	Model that executes and processes Firebase queries related with feedback data to the database.
TypeData Model	Model that executes and processes Firebase queries related with type data to the database.
SalesReportData Model	Model that executes and processes Firebase queries related with sales report data to the database.
SalesReportDetData Model	Model that executes and processes Firebase queries related with sales report detail data to the database.

1.4.3 Middleware Layer

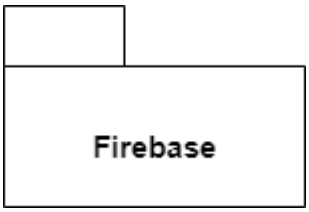

Figure 1.12 Middleware Layer

Table 1.12 Description of Middleware Layer

Class Name	Description
Firebase	A JSON-formatted cloud-hosted NoSQL database.

CHAPTER 2

2.1 Detailed Description

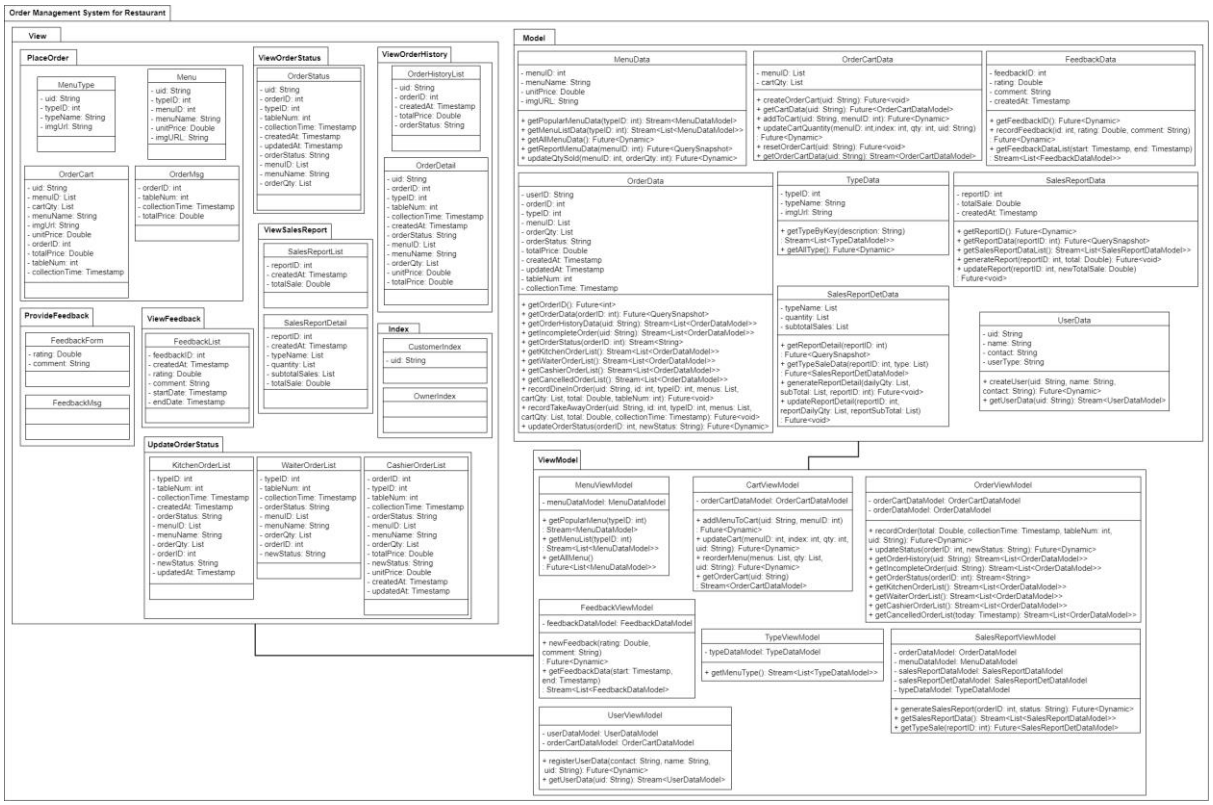

Figure 2.1 Detailed Design of Proposed System

2.1.1 Place Order

2.1.1.1 MenuType View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view and select the intended menu type.	
Attributes	**Attributes Name**	**Attributes Type**
	uid typeID typeName imgUrl	String int String String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.1.2 Menu View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view and select their intended menu under the menu type selected.	
Attributes	**Attributes Name**	**Attributes Type**
	uid typeID menuID menuName unitPrice imgURL	String int int String Double String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.1.3 OrderCart View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view list of intended menus added to the order cart and place an order based on the list.	
Attributes	**Attributes Name**	**Attributes Type**
	uid menuID cartQty menuName imgUrl unitPrice orderID totalPrice tableNum collectionTime	String List List String String Double int Double int Timestamp
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.1.4 OrderMsg View

Class Type	Boundary Class	
Responsibility	This class is to display the abstract information of the order placed to the customer.	
Attributes	**Attributes Name**	**Attributes Type**
	orderID tableNum collectionTime totalPrice	int int Timestamp Double
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable

Algorithm	Not Applicable

2.1.2 View Order Status

2.1.2.1 OrderStatus View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view the order status of their order. It also allows the customer to perform cancellation of order.	
Attributes	**Attributes Name**	**Attributes Type**
	uid orderID typeID tableNum collectionTime createdAt updatedAt orderStatus menuID menuName orderQty	String int int int Timestamp Timestamp Timestamp String List String List
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.3 View Order History

2.1.3.1 OrderHistoryList View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view the list of order placed.	
Attributes	**Attributes Name**	**Attributes Type**
	uid orderID createdAt totalPrice orderStatus	String int Timestamp Double String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.3.2 OrderDetail View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to view the details of a particular placed order. It also allows the customer to perform re-order action.	
Attributes	**Attributes Name**	**Attributes Type**
	uid orderID typeID tableNum collectionTime createdAt orderStatus menuID menuName orderQty unitPrice totalPrice	String int int int Timestamp Timestamp String List String List Double Double
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.4 Provide Feedback

2.1.4.1 FeedbackForm View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to fill and submit feedback to the restaurant.	
Attributes	**Attributes Name**	**Attributes Type**
	rating comment	Double String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.4.2 FeedbackMsg View

Class Type	Boundary Class	
Responsibility	This class displays a thank you message to the customer for providing feedback to the restaurant.	
Attributes	**Attributes Name**	**Attributes Type**
	Not Applicable	Not Applicable
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.5 View Feedback

2.1.5.1 FeedbackList View

Class Type	Boundary Class	
Responsibility	This class is to allow the business owner to view the feedback gathered from customers. It also allows the business owner to filter the feedback data according to the range of dates selected.	
Attributes	**Attributes Name**	**Attributes Type**
	feedbackID createdAt rating comment startDate endDate	int Timestamp Double String Timestamp Timestamp
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.6 View Sales Report

2.1.6.1 SalesReportList View

Class Type	Boundary Class	
Responsibility	This class is to allow the business owner to view abstract information of a list of sales report. The business owner can also select a particular sales report to view the details.	
Attributes	**Attributes Name**	**Attributes Type**
	reportID createdAt totalSale	int Timestamp Double
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.6.2 SalesReportDetail View

Class Type	Boundary Class	
Responsibility	This class is to allow the business owner to view the details of a particular sales report.	
Attributes	**Attributes Name**	**Attributes Type**
	reportID createdAt typeName quantity subtotalSales totalSale	int Timestamp List List List Double
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.7 Update Order Status

2.1.7.1 KitchenOrderList View

Class Type	Boundary Class	
Responsibility	This class is to allow the kitchen staff to view the list of order that with a status of "On Queue" and "Preparing". It also allows the kitchen staff to update the order status from "On Queue" to "Preparing" or from "Preparing" to "To Be Serve" when the constraints if fulfilled.	
Attributes	**Attributes Name**	**Attributes Type**
	typeID tableNum collectionTime createdAt orderStatus menuID menuName orderQty orderID newStatus updatedAt	int int Timestamp Timestamp String List String List int String Timestamp
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.7.2 WaiterOrderList View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to waiter to view the list of order with a status of "To Be Serve". The waiter can update the order status from "To Be Serve" to "Delivered" after the constraints is fulfilled.	
Attributes	**Attributes Name**	**Attributes Type**
	typeID tableNum collectionTime orderStatus menuID menuName orderQty orderID newStatus	int int Timestamp String List String List int String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.7.3 CashierOrderList View

Class Type	Boundary Class	
Responsibility	This class is to allow the cashier to view the list of order with a status of "Delivered". The cashier will update the status from "Delivered" to "Completed" after they received payment from the customer.	
Attributes	**Attributes Name**	**Attributes Type**
	orderID typeID tableNum collectionTime orderStatus menuID menuName orderQty totalPrice newStatus unitPrice createdAt updatedAt	int int int Timestamp String List String List Double String Double Timestamp Timestamp
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.8 Index

2.1.8.1 CustomerIndex View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to choose to place order, view order status, view order history and provide feedback.	
Attributes	**Attributes Name**	**Attributes Type**
	uid	String
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.8.2 OwnerIndex View

Class Type	Boundary Class	
Responsibility	This class is to allow the customer to choose to view sales report and view feedback gathered from customers.	
Attributes	**Attributes Name**	**Attributes Type**
	Not Applicable	Not Applicable
Methods	**Method Name**	**Description**
	Not Applicable	Not Applicable
Algorithm	Not Applicable	

2.1.9 ViewModel

2.1.9.1 Menu ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with MenuData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	menuDataModel	MenuDataModel
Methods	**Method Name**	**Description**
	getPopularMenu(typeID)	To get popular menu data of chosen menu type
	getMenuList(typeID)	To get menu list data of chosen menu type
	getAllMenu()	To get all menu data in a list for displaying menu ordered
Algorithm	**getPopularMenu(typeID)** **START** CALL getPopularMenuData FROM MenuData Model STORE returned result AS popularMenuData RETURN popularMenuData **END** **getMenuList(typeID)** **START** CALL getMenuListData FROM MenuData Model WITH typeID STORE returned result AS menuListData RETURN menuListData **END** **getAllMenu()** **START** CALL getAllMenuData FROM MenuData Model STORE returned result AS menuDatas DECLARE menuList WITH VARIABLE TYPE MenuDataModel ADD ALL menuDatas TO menuList WITH MenuDataModel RETURN menuList **END**	

2.1.9.2 Cart ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with OrderCartData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	orderCartDataModel	OrderCartDataModel
Methods	**Method Name**	**Description**
	addMenuToCart(uid, menuID)	To add menu data to user order cart
	updateCart(menuID, index, qty, uid)	To update the quantity of menu in order cart
	reorderMenu(menus, qty, uid)	To perform re-order function
	getOrderCart(uid)	To get data of user order cart
Algorithm	**addMenuToCart(uid, menuID)** **START** CALL addToCart FROM OrderCartData Model WITH uid AND menuID STORE returned result AS result RETURN result **END** **updateCart(menuID, index, qty, uid)** **START** CALL updateCartQuantity FROM OrderCartData Model WITH menuID, index, qty AND uid STORE returned result AS result RETURN result **END** **reorderMenu(menus, qty, uid)** **START** CALL getCartData FROM OrderCartData Model WITH uid STORE returned result AS orderCartData FOREACH menus AS menu CALL addToCart FROM OrderCartData Model WITH uid AND menu ENDFOREACH **END** **getOrderCart(uid)**	

	START CALL getOrderCartData FROM OrderCartData Model WITH uid STORE returned result AS orderCartData RETURN orderCartData **END**

2.1.9.3 Order ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with OrderData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	orderCartDataModel orderDataModel	OrderCartDataModel OrderDataModel
Methods	**Method Name**	**Description**
	recordOrder(total, collectionTime, tableNum, uid)	To save order cart information as a new order in database.
	updateStatus(orderID, newStatus)	To update the status of order based on the user type of the user.
	getOrderHistory(uid)	To get order history of a customer.
	getIncompleteOrder(uid)	To get customer's incomplete order
	getOrderStatus(orderID)	To get the order status of an order
	getKitchenOrderList()	To get order data for kitchen staff
	getWaiterOrderList()	To get order data for waiter
	getCashierOrderList()	To get order data for cashier
	getCancelledOrderList(today)	To get cancelled order data
Algorithm	**recordOrder(total, collectionTime, tableNum, uid)** **START** CALL getCartData FROM OrderCartData Model WITH uid STORE returned result AS orderCartData	

```
    CALL getOrderID FROM OrderData Model
    STORE returned result AS id
    SET typeID TO 1
    IF tableNum IS 0
        SET typeID TO 2
        CALL recordTakeAwayOrder FROM OrderData Model WITH uid, id,
                typeID, total, collectionTime AND menuID, cartQty IN
                orderCartData
    ELSE
        CALL recordDineInOrder FROM OrderData Model WITH uid, id,
                typeID, total, tableNum AND menuID, cartQty IN orderCartData
    ENDIF
    CALL resetOrderCart FROM OrderCartData Model WITH uid
    RETURN id+1;
END

updateStatus(orderID, newStatus)
START
    CALL updateOrderStatus FROM OrderData Model WITH orderID AND
            newStatus
    STORE returned result AS result
    RETURN result
END

getOrderHistory(uid)
START
    CALL getOrderHistoryData FROM OrderData Model WITH uid
    STORE returned result AS orderHistoryData
    RETURN orderHistoryData
END

getIncompleteOrder(uid)
START
    CALL getIncompleteOrder FROM OrderData Model WITH uid
    STORE returned result AS incompleteOrderData
    RETURN incompleteOrderData
END

getOrderStatus(orderID)
START
    CALL getOrderStatus FROM OrderData Model WITH orderID
    STORE returned result AS orderStatusData
    RETURN orderStatusData
END
```

	getKitchenOrderList() **START** CALL getKitchenOrderList FROM OrderData Model STORE returned result AS kitchenOrderListData RETURN kitchenOrderListData **END** **getWaiterOrderList()** **START** CALL getWaiterKitchenOrderList FROM OrderData Model STORE returned result AS waiterOrderListData RETURN waiterOrderListData **END** **getCashierOrderList()** **START** CALL getCashierOrderList FROM OrderData Model STORE returned result AS cashierOrderListData RETURN cashierOrderListData **END** **getCancelledOrderList(today)** **START** CALL getCancelledOrderList FROM OrderData Model STORE returned result AS cancelledOrderData RETURN cancelledOrderData **END**

2.1.9.4 Feedback ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with FeedbackData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	feedbackDataModel	FeedbackDataModel
Methods	**Method Name**	**Description**
	newFeedback(rating, comment)	To save new feedback in database

	getFeedbackData(start, end)	To get feedback data
Algorithm	**newFeedback(rating, comment)** **START** CALL getFeedbackID FROM FeedbackData Model STORE returned result AS id CALL recordFeedback FROM FeedbackData Model WITH id, rating AND comment STORE returned result AS result RETURN result **END** **getFeedbackData(start, end)** **START** CALL getFeedbackDataList FROM FeedbackData Model WITH start AND end STORE returned result AS feedbackData RETURN feedbackData **END**	

2.1.9.5 Type ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with TypeData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	typeDataModel	TypeDataModel
Methods	**Method Name**	**Description**
	getMenuType()	To get type of menu.
Algorithm	**getMenuType()** **START** CALL getTypeByKey FROM TypeDataModel WITH ‘menu’ STORE returned result AS menuTypeData RETURN menuTypeData **END**	

2.1.9.6 SalesReport ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with SalesReportData Model and SalesReportDetData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	orderDataModel menuDataModel salesReportDataModel salesReportDetDataModel typeDataModel	OrderDataModel Number MenuDataModel SalesReportDataModel SalesReportDetDataModel TypeDataModel
Methods	**Method Name**	**Description**
	generateSalesReport(orderID, status)	To generate new sales report or update sales report if existed
	getSalesReportData()	To get sales report data
	getTypeSale(reportID)	To get type sale data
Algorithm	**generateSalesReport(orderID, status)** **START** CALL getOrderData FROM OrderData Model WITH orderID STORE returned result AS orderData CALL getReportMenuData FROM MenuData Model WITH menuID IN orderData STORE returned result AS reportMenuData CALL updateQtySold FROM MenuDataModel WITH menuID IN reportMenuData AND orderQty IN orderData CALL getReportID FROM SalesReportData Model STORE returned result AS reportID CALL getReportData FROM SalesReportData Model WITH reportID STORE returned result AS lastReportData CALC subTotal, dailyQty BY typeID IF createdAt IN lastReportData EQUAL TO TODAY CALL getReportDetail FROM SalesReportDetData Model WITH reportID STORE returned result AS reportDetData CALC newReportSubtotal, newReportDailyQty, newTotalSale BY typeID CALL updateReport FROM SalesReportData Model WITH reportID AND newReportTotalSale CALL updateReportDetail FROM SalesReportDetDataModel WITH reportID, newReportDailyQty AND newReportTotalSale	

```
    ELSE
        CALL generateReport FROM SalesReportData Model WITH reportID AND
                total IN orderData
        CALL generateReportDetail FROM SalesReportDetData Model WITH
                dailyQty, subTotal AND reportID
    ENDIF
END

getSalesReportData()
START
    CALL getSalesReportDataList FROM SalesReportData Model WITH reportID
    STORE returned result AS salesReportDataList
    RETURN salesReportDataList
END

getTypeSale(reportID)
START
    CALL getReportDetail FROM SalesReportDetData Model WITH reportID
    STORE returned result AS reportDetQs
    STORE FIRST id OF reportDetQs AS docID
    DECLARE typeList WITH VARIABLE TYPE List
    CALL getAllType FROM TypeData Model
    STORE returned result AS typeDatas
    ADD ALL typeDatas TO typeList
    CALL getTypeSaleData FROM SalesReportDetData Model WITH reportID AND
            typeList
    STORE returned result AS typeSaleData
    RETURN typeSaleData
END
```

2.1.9.7 User ViewModel

Class Type	ViewModel Class	
Responsibility	This class is responsible for managing and processing all operations related with UserData Model.	
Attributes	**Attributes Name**	**Attributes Type**
	userDataModel orderCartDataModel	UserDataModel OrderCartDataModel
Methods	**Method Name**	**Description**
	registerUserData(contact, name, uid)	To register a new user
	getUserData(uid)	To get user data
Algorithm	**registerUserData(contact, name, uid)** **START** CALL createOrderCart FROM OrderCartData Model WITH uid CALL createUser FROM UserData Model WITH uid, name AND contact STORE returned result AS result RETURN result **END** **getUserData(uid)** **START** CALL getUserData FROM UserData Model WITH uid STORE returned result AS userData RETURN userData **END**	

2.1.10 Model

2.1.10.1 MenuData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with menu data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	menuID menuName unitPrice imgURL	int String Double String
Methods	**Method Name**	**Description**
	getPopularMenuData(typeID)	To get popular menu data of chosen menu type
	getMenuListData(typeID)	To get menu list data of chosen menu type
	getAllMenuData()	To get all menu data in a list for displaying menu ordered
	getReportMenuData(menuID)	To get menu data for sales report generation
	updateQtySold(menuID, orderQty)	To update value of total quantity sold of a menu
Algorithm	**getPopularMenuData(typeID)** **START** QUERY menu data FROM Menu Collection IN Firebase WITH typeID ORDER BY qtySold STORE query result AS popularMenuData RETURN menuTypeData and typeNameData **END** **getMenuListData(typeID)** **START** QUERY menu data FROM Menu Collection IN Firebase WITH typeID ORDER BY menuID STORE query result AS menuListData RETURN menuListData **END** **getAllMenuData()**	

	START QUERY menu data FROM Menu Collection IN Firebase ORDER BY menuID STORE query result AS allMenuData RETURN allMenuData **END** **getReportMenuData(menuID)** **START** QUERY menu data FROM Menu Collection IN Firebase WITH menuID STORE query result AS menuData RETURN menuData **END** **updateQtySold(menuID, orderQty)** **START** QUERY qtySold FROM Menu Collection IN Firebase WITH menuID STORE query result AS currentQtySold SUM currentQtySold AND orderQty AS newQty UPDATE qtySold FROM Menu Collection IN Firebase WITH menuID TO newQty **END**

2.1.10.2 OrderCartData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with order cart data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	menuID cartQty	List List
Methods	**Method Name**	**Description**
	getCartData(uid)	To get order cart data for re-ordering.
	getOrderCartData(uid)	To get order cart data for display.
	createOrderCart(uid)	To create order cart for user.
	addToCart(uid, menuID)	To add menu into order cart.

	updateCartQuantity(menuID, index, qty, uid)	To update the quantity of menu added in order cart.
	resetOrderCart(uid)	To reset the order cart to empty.

Algorithm	**getCartData(uid)**
	START
	QUERY order cart data FROM OrderCart Collection IN Firebase WITH uid
	STORE query result AS cartData
	RETURN cartData
	END
	
	getOrderCartData(uid)
	START
	QUERY order cart data FROM OrderCart Collection IN Firebase WITH uid
	STORE query result AS cartData
	RETURN cartData
	END
	
	createOrderCart(uid)
	START
	CREATE order cart data IN OrderCart Collection IN Firebase WITH uid AS document ID
	END
	
	addToCart(uid, menuID)
	START
	UPDATE menuID FROM OrderCart Colelction IN Firebase WITH uid TO menuID
	UPDATE cartQty FROM OrderCart Colelction IN Firebase WITH uid BY APPEND 1
	END
	
	updateCartQuantity(menuID,index, qty, uid)
	START
	IF qty IS NOT 0
	UPDATE cartQty FROM OrderCart Collection IN Firebase WITH uid TO qty AT index
	ELSE
	DELETE menuID FROM OrderCart Collection IN Firebase WITH uid AND menuID AT index
	DELETE cartQty FROM OrderCart Collection IN Firebase WITH uid AND menuID AT index
	ENDIF
	END	

	resetOrderCart(uid) **START** UPDATE menuID AND cartQty FROM OrderCart Collection IN Firebase WITH uid TO NULL **END**

2.1.10.3 OrderData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with order data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	userID orderID typeID menuID orderQty orderStatus totalPrice createdAt updatedAt tableNum collectionTime	String int int List List String Double Timestamp Timestamp int Timestamp
Methods	**Method Name**	**Description**
	getOrderID()	To get orderID for new order.
	getOrderData(orderID)	To get order data of an order.
	getOrderHistoryData(uid)	To get order history data of a particular customer.
	getIncompleteOrder(uid)	To get incomplete order data of a particular customer.
	getOrderStatus(orderID)	To get order status of an order.
	getKitchenOrderList()	To get order waiting to be cook.
	getWaiterOrderList()	To get order waiting to be deliver.

	getCashierOrderList()	To get order to be pay.
	getCancelledOrderList()	To get today's cancelled order data
	recordDineInOrder(uid, id, typeID, menus, cartQty, total, tableNum)	To save order as Dine In order into database.
	recordTakeAwayOrder(uid, id, typeID, menus, cartQty, total, collectionTime)	To save order as Take Away order into database.
	updateOrderStatus(orderID, newStatus)	To update the status of customer order.

Algorithm

```
getOrderID()
START
    COUNT order data FROM Order Collection IN Firebase
    STORE query result AS orderID
    RETURN orderID
END

getOrderData(orderID)
START
    QUERY order data IN Order Colelction IN Firebase WITH orderID
    STORE query result AS orderData
    RETURN orderData
END

getOrderHistoryData(uid)
START
    QUERY order data FROM Order Collection IN Firebase WITH uid ORDER
            BY createdAt DESCENDING
    STORE query result AS orderHistoryData
    RETURN orderHistoryData
END

getIncompleteOrder(uid)
START
    QUERY order data FROM Order Collection IN Firebase WITH uid AND
            orderStatus NOT "Completed" OR "Cancelled" ORDER BY
            createdAt DESCENDING
    STORE query result AS incompleteOrderData
    RETURN incompleteOrderData
END

getOrderStatus(orderID)
START
```

```
    QUERY order data FROM Order Collection IN Firebase WITH orderID
    STORE query result AS orderStatusData
    RETURN orderStatusData
END

getKitchenOrderList()
START
    QUERY order data FROM Order Collection IN Firebase WITH orderStatus
            IS "On Queue" OR "Preparing"
    STORE query result AS orderListData
    RETURN orderListData
END

getWaiterOrderList()
START
    QUERY order data FROM Order Collection IN Firebase WITH orderStatus
            IS "To Be Serve"
    STORE query result AS orderListData
    RETURN orderListData
END

getCashierOrderList()
START
    QUERY order data FROM Order Collection IN Firebase WITH orderStatus
            IS "Delivered"
    STORE query result AS orderListData
    RETURN orderListData
END

getCancelledOrderList()
START
    QUERY order data FROM Order Collection IN Firebase WITH orderStatus
            IS "Cancelled"
    STORE query result AS cancelledOrderListData
    RETURN cancelledOrderListData
END

recordDineInOrder(uid, id, typeID, menus, cartQty, total, tableNum)
START
    INSERT uid, id, typeID, menus, cartQty, total, tableNum INTO Order
            Collection IN Firebase WITH orderStatus IS'On Queue',
            createdAt AND updatedAt EQUAL TO NOW
END
```

	recordTakeAwayOrder(uid, id, typeID, menus, cartQty, total, collectionTime) **START** INSERT uid, id, typeID, menus, cartQty, total, collectionTime INTO Order Collection IN Firebase WITH orderStatus IS'On Queue', createdAt AND updatedAt EQUAL TO NOW **END** **updateOrderStatus(orderID, newStatus)** **START** UPDATE orderStatus AND updatedAt FROM Order Collection IN Firebase WITH orderID TO newStatus AND NOW END

2.1.10.4 FeedbackModel

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with feedback data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	feedbackID rating comment createdAt	int Double String Timestamp
Methods	**Method Name**	**Description**
	getFeedbackID()	To get feedback ID for new feedback
	recordFeedback(id, rating, comment)	To record feedback into database.
	getFeedbackDataList(start, end)	To get feedback data.
Algorithm	**getFeedbackID()** **START** COUNT feedback data FROM Feedback Collection IN Firebase STORE query result AS feedbackID RETURN feedbackID **END**	

	recordFeedback(id, rating, comment) **START** INSERT id+1, rating, comment INTO Feedback Collection WITH createdAt EQUAL TO NOW **END** **getFeedbackDataList(start, end)** **START** SET end EQUAL TO end+1 QUERY feedback data FROM Feedback Collection WITH createdAt BETWEEN start AND end ORDER BY createdAt DESCENDING STORE query result AS feedbackDataList RETURN feedbackDataList **END**

2.1.10.5 TypeData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with type data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	typeID typeName imgUrl	int String String
Methods	**Method Name**	**Description**
	getTypeByKey(description)	To get type data using keyword
	getAllType()	To get the data of the type.
Algorithm	**getTypeByKey(description)** **START** QUERY type data FROM Type Collection IN Firebase WITH description ORDER BY typeID STORE query result AS typeData RETURN typeData **END**	

	getAllType() **START** QUERY type data FROM Type Collection IN Firebase ORDER BY typeID STORE query result AS typeData RETURN typeData **END**

2.1.10.6 SalesReportData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with sales report data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	reportID totalSale createdAt	int Double Timestamp
Methods	**Method Name**	**Description**
	getReportID()	To get new reportID
	getReportData(reportID)	To get data of a report.
	getSalesReportDataList()	To get sales report list.
	generateReport(reportID, total)	To generate sales report.
	updateReport(reportID, newTotalSale)	To update existing sales report.
Algorithm	**getReportID()** **START** COUNT report data FROM SalesReport Collection IN Firebase STORE query result AS reportID RETURN reportID **END** **getReportData(reportID)** **START** QUERY report data FROM SalesReport Collection IN Firebase WITH reportID STORE query result AS salesReportData RETURN salesReportData	

	END **getSalesReportDataList()** **START** QUERY report data FROM SalesReport Collection IN Firebase ORDER BY reportID STORE query result AS salesReportData WITH SalesReportDataModel RETURN salesReportData **END** **generateReport(reportID, total)** **START** INSERT reportID+1, total INTO SalesReport Collection IN Firebase WITH createdAt EQUAL TO NOW **END** **updateReport(reportID, newTotalSale)** **START** UPDATE totalSale FROM SalesReport Collection IN Firebase WITH reportID TO newTotalSale **END**

2.1.10.7 SalesReportDetData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with sales report detail data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	typeName quantity subTotal	List List List
Methods	**Method Name**	**Description**
	getReportDetail(reportID)	To get sales report detail data of a report.
	getTypeSaleData(reportID, type)	To get type sale data of a report.
	generateReportDetail(dailyQty, subTotal, reportID)	To generate sales report detail into database.

	updateReportDetail(reportID, reportDailyQty, reportSubTotal)	To update sales report detail of a report.
Algorithm	**getReportDetail(reportID)** **START** QUERY sales report detail data FROM SalesReportDetail Collection IN Firebase WITH reportID STORE query result AS reportDetail RETURN reportDetail **END** **getTypeSaleData(reportID, type)** **START** QUERY sales report detail data FROM SalesReportDetail Collection IN Firebase WITH reportID STORE query result AS reportDetail EXTRACT dailyQuantity AND subTotal IN reportDetail INTO SalesReportDetDataModel WITH type STORE result AS typeSaleData RETURN typeSaleData **END** **generateReportDetail(dailyQty, subTotal, reportID)** **START** INSERT dailyQty, subTotal, reportID+1 INTO SalesReportDetail Collection **END** **updateReportDetail(reportID, reportDailyQty, reportSubTotal)** **START** UPDATE dailyQuantity, subTotal FROM SalesReportDetail Collection WITH reportID TO reportDailyQty AND reportSubTotal **END**	

2.1.10.8 UserData Model

Class Type	Model Class	
Responsibility	This class executes and processes Firebase queries related with user data to the database.	
Attributes	**Attributes Name**	**Attributes Type**
	userID name contactuserType	String String String
Methods	**Method Name**	**Description**
	createUser(uid, name, contact)	To create new user.
	getUserData(uid)	To get user data.
Algorithm	**createUser(uid, name, contact)** **START** INSERT uid, name, contact INTO User Collection IN Firebase WITH userType EQUAL TO “Customer” **END** **getUserData(uid)** **START** QUERY user data FROM User Collection IN Firebase WITH uid STORE query result AS userData WITH UserDataModel RETURN userData **END**	

2.2 DATA DICTIONARY

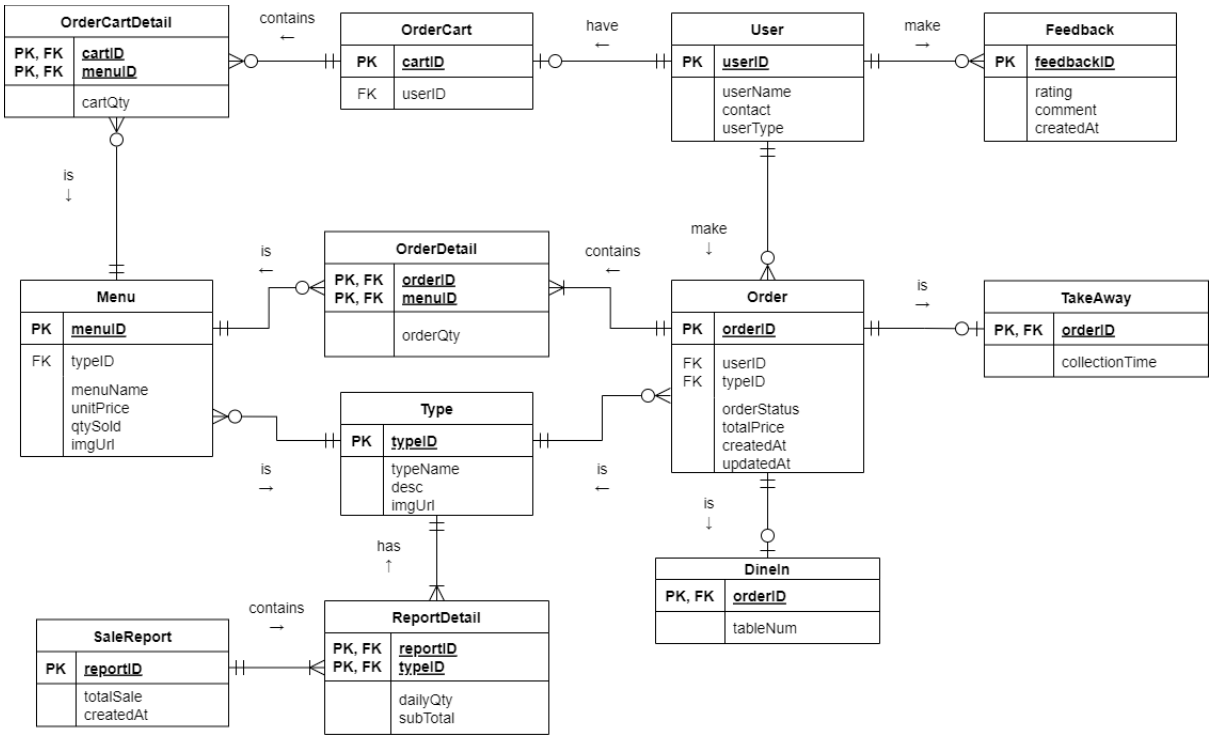

Figure 2.2 Entity Relationship Diagram

Figure 2.2 shows the attributes of the entities in MySQL database. However, the database used for the proposed system is Firebase database which is a NoSQL database. There might be some entities combined in Firebase database, but the attributes used are still the same. Figures below shows how the data is stored in the collection in Firebase database and the table depicts the data dictionary of the entities in Figure 2.2 that involved in the collection in Firebase database.

```
[
    {
        "__id__": "UCtabgOJv0P8yQ6IMha2IxADkNX2",
        "userName": "Chong Kai Jie",
        "contact": "0167789136",
        "userType": "Owner"
    },
    {
        "__id__": "YawfIM1trmcyPIXDrTiZmvG7xcS2",
        "userName": "Wong See Hua",
        "contact": "0175439665",
        "userType": "Kitchen Staff"
    },
    {
        "__id__": "hJaJCd8x0zPXvSjNqmxhoy2qrHf1",
        "userName": "Chan Kin Keong",
        "contact": "0163289462",
        "userType": "Waiter"
    },
    {
        "__id__": "eMxDha1An3SxUVxSjQcQaG0nvRs3",
        "userName": "Cheong Kai Wei",
        "contact": "0124834452",
        "userType": "Cashier"
    },
    {
        "__id__": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "userName": "Lee Yong Jian",
        "contact": "0112368846",
        "userType": "Customer"
    }
]
```


Figure 2.3 User Collection in JSON

Table 2.1 Data Dictionary of User Entity

Attribute Name	Attribute Type	Description	Constraints
userID	String	ID of user.	PK
userName	String	Name of user.	
contact	String	User contact number.	
userType	String	Type of user	

```
[
    {
        "__id__": "YC1MWy7qFPeehB0aNlry",
        "typeID": 1,
        "typeName": "Dine In",
        "desc": "order"
    },
    {
        "__id__": "hP1Kw6OvN8kaslJ5LDQp",
        "typeID": 2,
        "typeName": "Take Away",
        "desc": "order"
    },
    {
        "__id__": "BBPYiHP73PMYZ1hqIehp",
        "typeID": 3,
        "typeName": "Burger",
        "desc": "menu",
        "imgUrl": "burger/00.jpg"
    },
    {
        "__id__": "VGKAyYfNJaFmGYzHTlem",
        "typeID": 4,
        "typeName": "Noodle",
        "desc": "menu",
        "imgUrl": "noodle/00.jpg"
    },
    {
        "__id__": "dDpABAuBkvncJKiUDBJf",
        "typeID": 5,
        "typeName": "Rice",
        "desc": "menu",
        "imgUrl": "rice/00.jpg"
    },
    {
        "__id__": "9TiK79hACTTl7crFuTIv",
        "typeID": 6,
        "typeName": "Soup",
        "desc": "menu",
        "imgUrl": "soup/00.jpg"
    },
    {
        "__id__": "RAvQbmzQX7pnSETvTbdn",
        "typeID": 7,
        "typeName": "Beverage",
        "desc": "menu",
        "imgUrl": "beverage/00.jpg"
    }
]
```


Figure 2.4 Type Collection in JSON

Table 2.2 Data Dictionary of Type Entity

Attribute Name	Attribute Type	Description	Constraints
typeID	Number	ID of type.	PK
typeName	String	Name of the type.	
desc	String	Description of the type.	
imgUrl	String	Path of image of the type located.	


```
[
    {
        "__id__": "ItIOhavD0TCArLz5wNc5",
        "feedbackID": 1,
        "rating": 3.5,
        "comment": "Food are delicious but waiting time are too long.",
        "createdAt": "__Timestamp__2022-03-13T07:07:47.429Z"
    },
    {
        "__id__": "XNr9ITaxRTa1eglq6aUv",
        "feedbackID": 1,
        "rating": 3.5,
        "comment": "Staff are kind and friendly. Worth of money.",
        "createdAt": "__Timestamp__2022-03-13T07:07:47.429Z"
    }
]
```


Figure 2.5 Feedback Collection in JSON

Table 2.3 Data Dictionary of Feedback Entity

Attribute Name	Attribute Type	Description	Constraints
feedbackID	Number	ID of user feedback.	PK
rating	Number	User rating on their ordering experience.	
comment	String	User comment on their ordering experience.	
createdAt	Timestamp	Date and time of feedback created.	

```
[
    {
        "__id__": "iHpbFHmKHCTPIIB5ZHRf",
        "menuID": 1,
        "typeID": 3,
        "menuName": "Cheese Burger",
        "unitPrice": 12.5,
        "qtySold": 22,
        "imgUrl": "burger/01.jpg"
    },
    {
        "__id__": "DXD63djMlQjrHPqMzuJE",
        "menuID": 2,
        "typeID": 3,
        "menuName": "Beef Burger",
        "unitPrice": 9.3,
        "qtySold": 7,
        "imgUrl": "burger/02.jpg"
    },
    {
        "__id__": "yvhhjyZp28rEqtVFNflM",
        "menuID": 3,
        "typeID": 4,
        "menuName": "Mee Goreng",
        "unitPrice": 5.6,
        "qtySold": 16,
        "imgUrl": "noodle/01.jpg"
    }
]
```


Figure 2.6 Menu Collection in JSON

Table 2.4 Data Dictionary of Menu Entity

Attribute Name	Attribute Type	Description	Constraints
menuID	Number	ID of menu.	PK
typeID	Number	ID of type of the menu.	FK
menuName	String	Name of menu.	
unitPrice	Number	Unit price of menu.	
qtySold	Number	Quantity of menu sold.	
imgURL	String	Path of menu image located.	

```
[
    {
        "__id__": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "menuID": [
            2,
            4
        ],
        "cartQty": [
            1,
            3
        ]
    }
]
```

Figure 2.7 OrderCart Collection in JSON

Table 2.5 Data Dictionary of OrderCart Entity

Attribute Name	Attribute Type	Description	Constraints
cartID	String	ID of user order cart.	PK
userID	String	ID of user.	FK

Table 2.6 Data Dictionary of OrderCartDetail Entity

Attribute Name	Attribute Type	Description	Constraints
cartID	String	ID of user order cart.	PK, FK
menuID	Array (Number)	ID of menu.	PK, FK
cartQty	Array (Number)	Quantity of menu added.	

```
[
    {
        "__id__": "MqHdIKZc9uN89SYqbOI7",
        "orderID": 2,
        "userID": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "typeID": 1,
        "orderStatus": "On Queue",
        "totalPrice": 36.36,
        "createdAt": "__Timestamp__2022-03-13T07:13:59.133Z",
        "updatedAt": "__Timestamp__2022-03-13T07:22:38.909Z",
        "menuID": [
            3,
            7,
            10,
            1
        ],
        "orderQty": [
            1,
            1,
            1,
            1
        ],
        "tableNum": 5
    },
    {
        "__id__": "ibBOaoZWuHLxNSfmz9rb",
        "orderID": 1,
        "userID": "wjZrQn6yC8GDdxPiLpgFvR9kdMx0",
        "typeID": 2,
        "orderStatus": "Cancelled",
        "totalPrice": 49.61,
        "createdAt": "__Timestamp__2022-03-13T07:12:51.714Z",
        "updatedAt": "__Timestamp__2022-03-13T07:13:40.412Z",
        "menuID": [
            3,
            7,
            10,
            1
        ],

        "orderQty": [
            1,
            1,
            1,
            2
        ],
        "collectionTime": "__Timestamp__2022-03-13T07:25:48.000Z"
    }
]
```


Figure 2.8 Order Collection in JSON

Table 2.7 Data Dictionary of Order Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK
userID	String	ID of user.	FK
typeID	Number	ID of type of the order.	FK
orderStatus	String	Status of user order.	
totalPrice	Number	Total price of user order.	
createdAt	Timestamp	Date and time of order created.	
updatedAt	Timestamp	Date and time of order updated.	

Table 2.8 Data Dictionary of TakeAway Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
collectionTime	Timestamp	Collection time of an order.	

Table 2.9 Data Dictionary of DineIn Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
tableID	Number	ID of the table for an order.	

Table 2.10 Data Dictionary of OrderDetail Entity

Attribute Name	Attribute Type	Description	Constraints
orderID	Number	ID of user order.	PK, FK
menuID	Array (Number)	ID of menu ordered.	PK, FK
orderQty	Array (Number)	Quantity of menu ordered.	

```
[
    {
        "__id__": "1M4S5xRb9FqHSflBZNei",
        "reportID": 1,
        "totalSale": 72.72,
        "createdAt": "__Timestamp__2022-03-13T07:21:13.344Z"
    }
]
```

Figure 2.09 SaleReport Collection in JSON

Table 2.11 Data Dictionary of SaleReport Entity

Attribute Name	Attribute Type	Description	Constraints
reportID	Number	ID of sale report.	PK
totalSale	Number	Total sale of the report.	
createdAt	Timestamp	Date and time of report created.	

```
[
    {
        "__id__": "EKKBFnHCJo8BA1XMPra9",
        "reportID": 1,
        "dailyQuantity": [
            2,
            0,
            2,
            2,
            0,
            2,
            2
        ],
        "subTotal": [
            72.72,
            0,
            25,
            11.2,
            0,
            25.2,
            7.2
        ]
    }
]
```


Figure 2.10 ReportDetail Collection in JSON

Table 2.12 Data Dictionary of ReportDetail Entity

Attribute Name	Attribute Type	Description	Constraints
reportID	Number	ID of report.	PK, FK
typeID	Number	ID of type.	PK, FK
dailyQuantity	Array (Number)	Quantity sold for each type.	
subTotal	Array (Number)	Subtotal of profit for each type.	

APPENDIX C
USER MANUAL OF ORDER MANAGEMENT SYSTEM FOR RESTAURANT (OMSR)

For Appendices Heading, use TITLE AT ROMAN PAGES style.

TABLE OF CONTENT

1.0 General Information

1.1 System Overview

Order Management System for Restaurant (OMSR) is an ordering application that facilitates and simplifies the ordering and management process. There are five types of user involved which is customer, business owner, kitchen staff, waiter and cashier. Each user will have different accessibility to the system module. The system consists of seven modules which is place order module, view order status module, view order history module, provide feedback module, view sales report module, view feedback module and update order status module.

2.0 System Summary

2.1 System Configuration

Order Management System for Restaurant (OMSR) operates on mobile devices with Android operating system. The mobile device used to run the application must connected with internet, have Android version 7.0 or above and at least 3GB RAM.

3.0 Getting Started

3.1 System Interfaces

3.1.1 Login Page

Figure 3.1 shows the login page that will be displayed to the user when they are not logged in to the application. User must fill in the input field before they can login. Table 3.1 shows the description of each element in Figure 3.1.

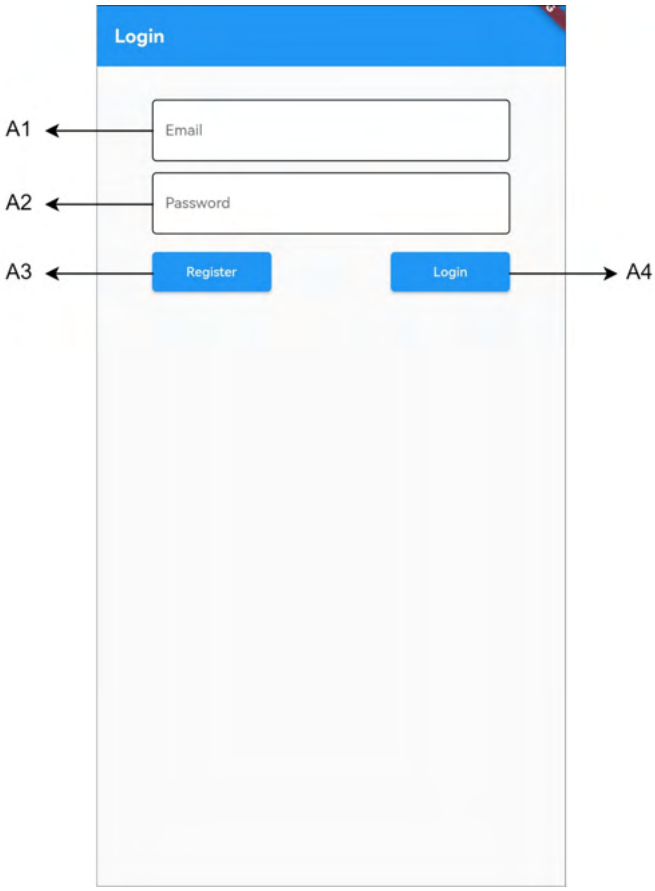

Figure 3.1 Login Page

Table 3.1 Description of Elements in Login Page

Label	Description
A1	Input Field for Email Address of Registered Account
A2	Input Field for Password of Registered Account
A3	Redirects Registration Page
A4	Login User with Values in A1 and A2

3.1.2 Registration Page

Figure 3.2 depicts the registration page of Order Management System for Restaurant. Users must fill in the provided input field and press the "Register" button to register their account. Table 3.2 shows the description of each element in Figure 3.2.

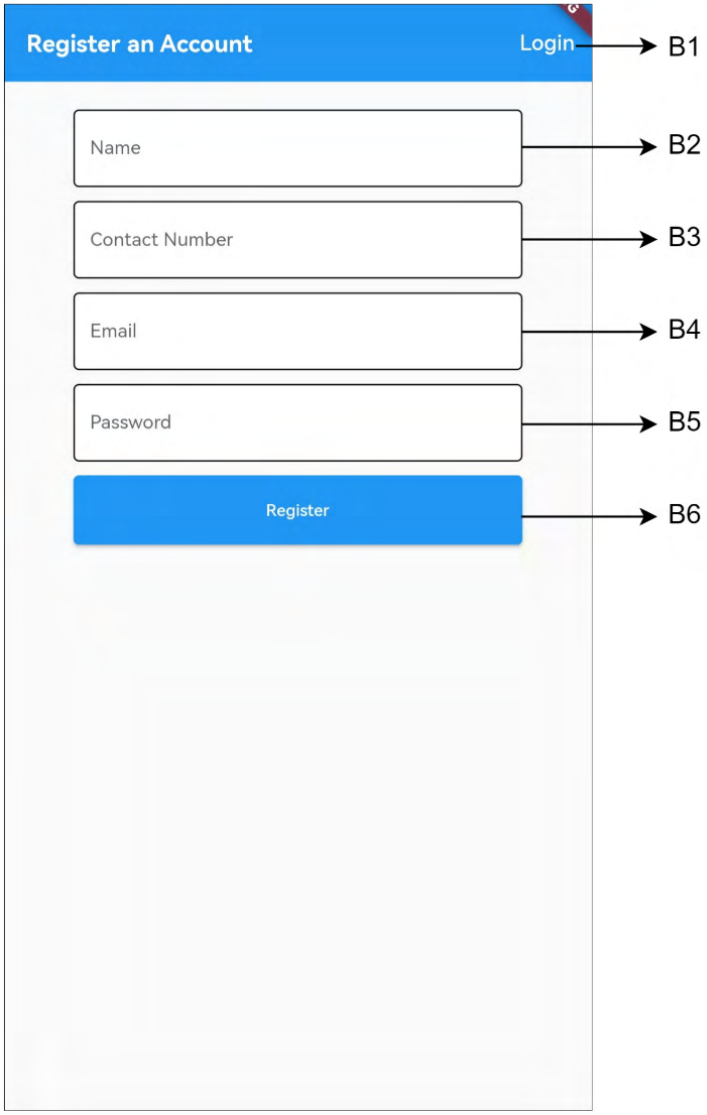

Figure 3.2 Registration Page

Table 3.2 Description of Elements in Registration Page

Label	Description
B1	Redirect User to Login Page
B2	Input Field for Name of User
B3	Input Field for User Contact Number
B4	Input Field for User Email Address
B5	Input Field for Password
B6	Register User Account with Values in B2, B3, B4 and B5.

3.1.3 Verify Email Page

Figure 3.3 illustrates the verify email page. All registered users must verify their email before they can access the system modules. Table 3.3 shows the description of each element in Figure 3.3.

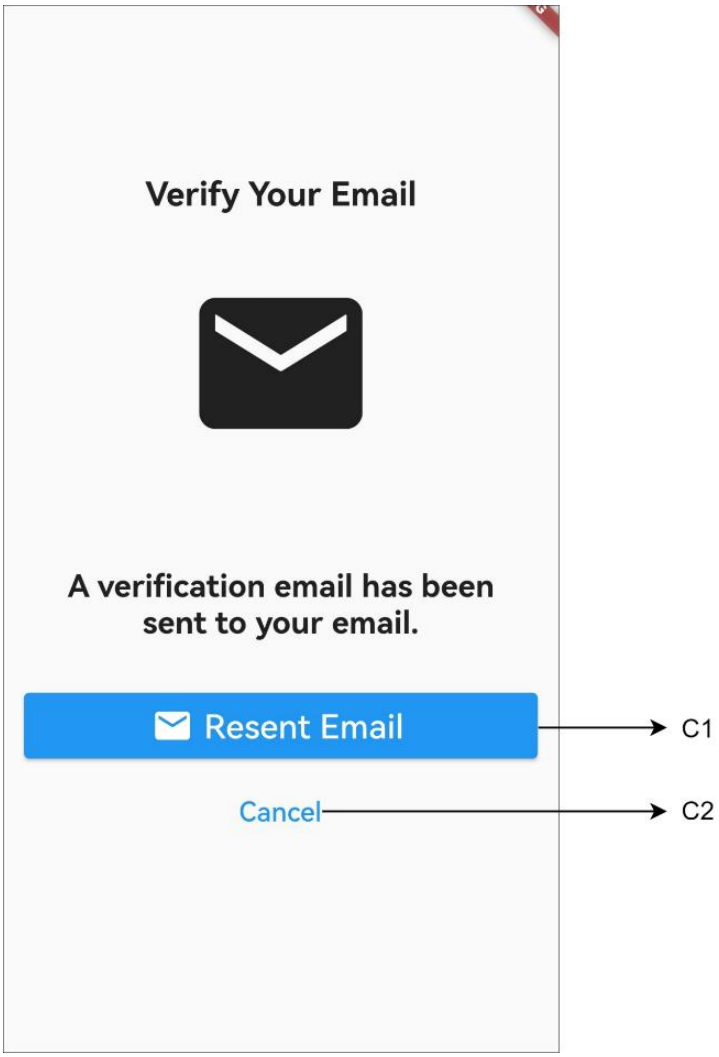

Figure 3.3 Verify Email Page

Table 3.3 Description of Elements in Verify Email Page

Label	Description
C1	Resent Verification Email to Registered Email Address
C2	Terminate Email Verification Process, Logout and Redirect to Login Page

3.2 Customer Interfaces

3.2.1 Customer Homepage

Figure 3.4 shows the homepage of customer. This page will be accessed when the logged in as customer. Customer can access to different module by pressing the related button in this page. Table 3.4 describes the function of each element in Figure 3.4.

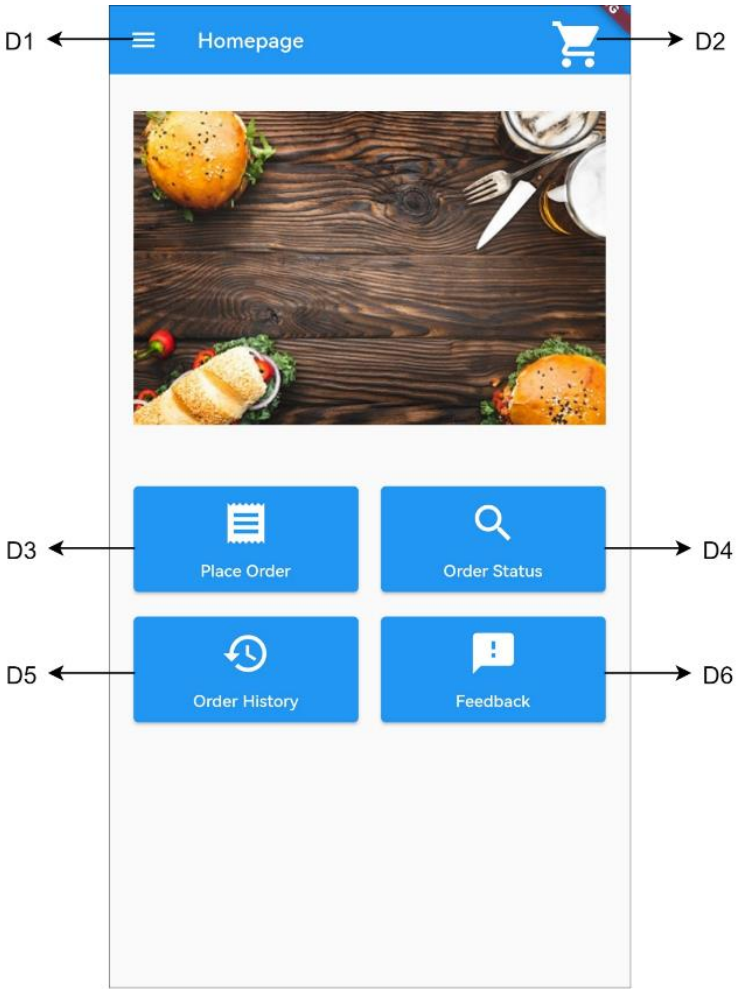

Figure 3.4 Customer Homepage

Table 3.4 Description of Elements in Customer Homepage

Label	Description
D1	Access Customer Navigation Menu
D2	Redirect to Order Cart Page of Place Order Module
D3	Redirect to Menu Type Page of Place Order Module
D4	Redirect to Order Status Page of View Order Status Module
D5	Redirect to Order History Page of View Order History Module
D6	Redirect to Feedback Form Page of Provide Feedback Module

3.2.2 Customer Navigation Menu

Figure 3.5 illustrates the navigation menu of the customer. Other than pressing the button on homepage to access module, they can also access the module using this navigation menu. Table 3.5 shows the description of each element in Figure 3.5.

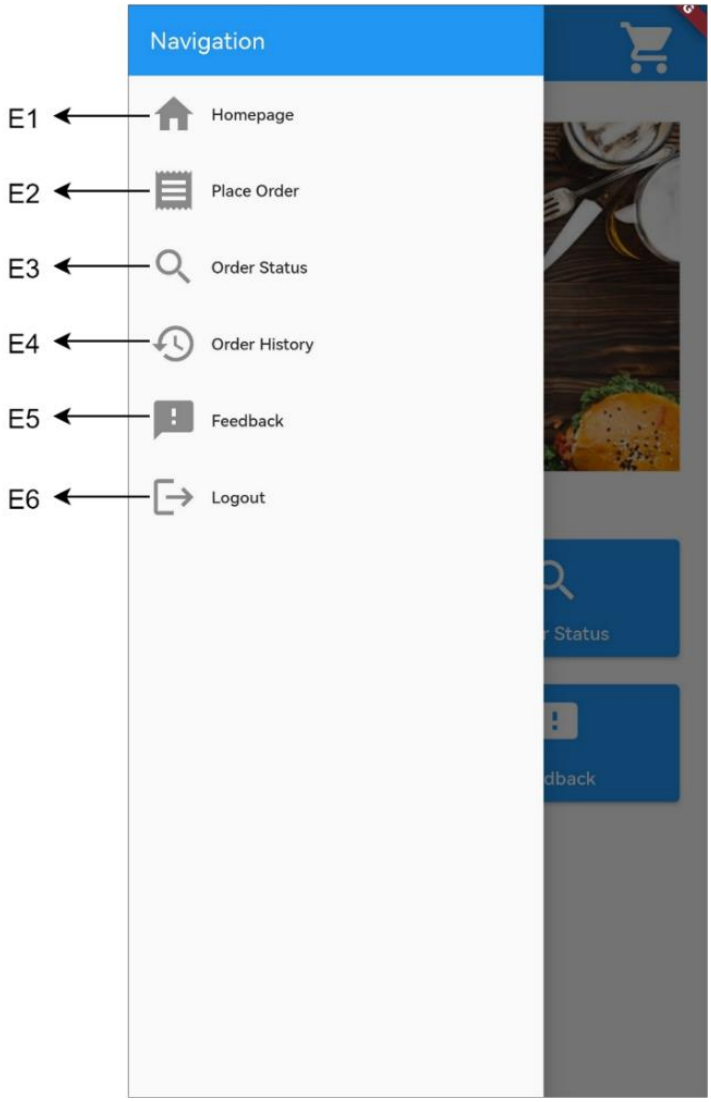

Figure 3.5 Customer Navigation Menu

Table 3.5 Description of Elements in Customer Navigation Menu

Label	Description
E1	Redirect to Customer Homepage
E2	Redirect to Menu Type Page of Place Order Module
E3	Redirect to Order Status Page of View Order Status Module
E4	Redirect to Order History Page of View Order History Module
E5	Redirect to Feedback Form Page of Provide Feedback Module
E6	Logout Customer and Redirect to Login Page

3.2.3 Menu Type Page

Figure 3.6 shows the menu type page when customer access the place order module. Customer will need to choose their desired menu type to view the menu available for ordering. Table 3.6 describes the function of each element in Figure 3.5.

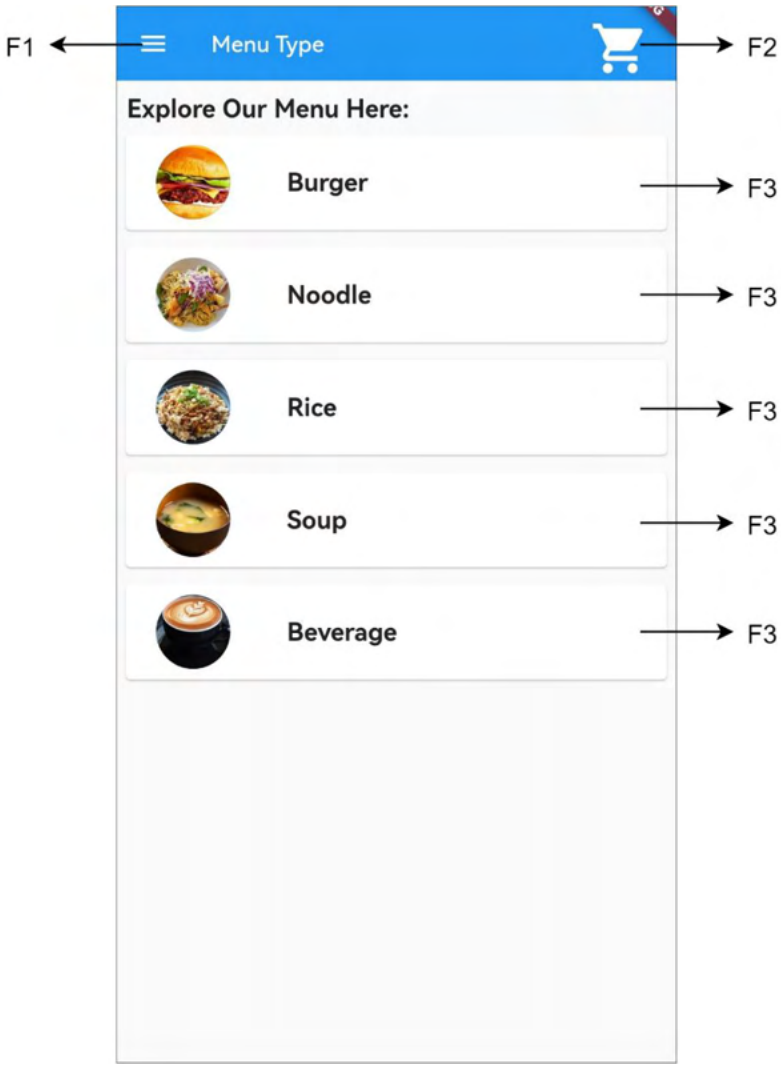

Figure 3.6 Menu Type Page

Table 3.6 Description of Elements in Menu Type Page

Label	Description
F1	Access Customer Navigation Menu
F2	Redirect to Order Cart Page of Place Order Module
F3	Redirect to Menu List Page of Selected Menu Type

3.2.4 Menu List Page

Figure 3.7 shows the menu list page of menu type “Burger”. This page is displayed when the customer pressed the “Burger” option in Figure 3.6. If the customer presses a different menu type, a different list of menus will be displayed. The customer can add their desired menu to order cart by pressing the button labelled with G3. Table 3.7 shows the description of each element in Figure 3.7.

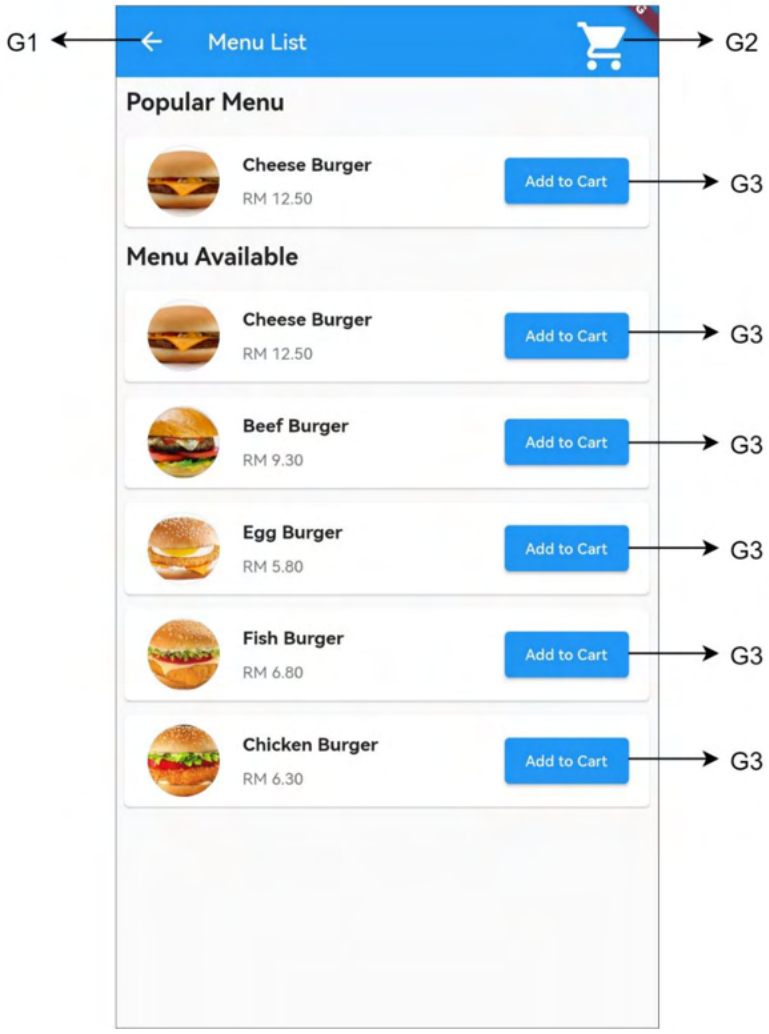

Figure 3.7 Menu List Page

Table 3.7 Description of Elements in Menu List Page

Label	Description
G1	Access Customer Navigation Menu
G2	Redirect to Order Cart Page of Place Order Module
G3	Add Selected Menu to Order Cart

3.2.5 Order Cart Page

Figure 3.8 shows the order cart page of the customer. The customer can manage their desired menu before they place it as an order. Table 3.8 describes the function of each element in Figure 3.8.

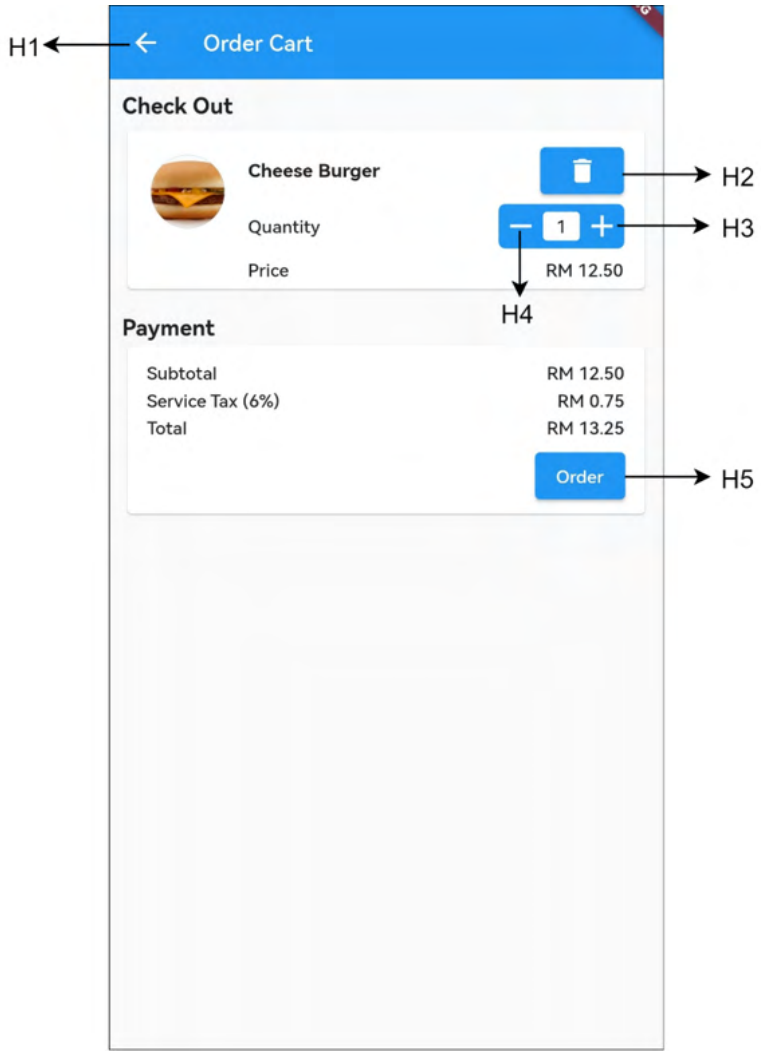

Figure 3.8 Order Cart Page

Table 3.8 Description of Elements in Order Cart Page

Label	Description
H1	Return to Previous Page
H2	Delete Selected Menu from Order Cart Page
H3	Increase Order Quantity of Selected Menu by 1
H4	Decrease Order Quantity of Selected Menu by 1
H5	Proceed the Items in Order Cart as Order and Display Select Order Type Panel

3.2.6 Select Order Type Panel

Figure 3.9 depicts the select order type panel when the customer wants to place their order. Customer needs to choose either they want to “Dine In” or “Take Away” the order. Table 3.9 describes the function of each element in Figure 3.9.

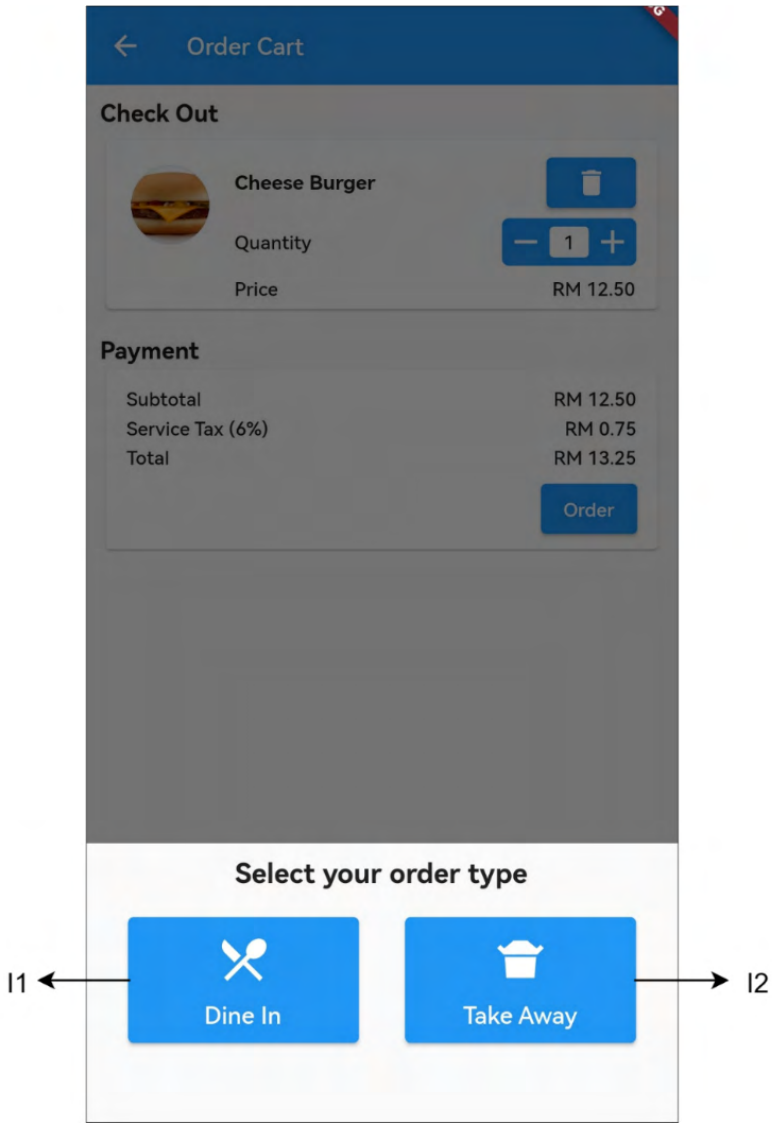

Figure 3.9 Select Order Type Panel

Table 3.9 Description of Elements in Select Order Type Panel

Label	Description
I1	Proceed the Order with Type “Dine In” and Display Dine In Panel
I2	Proceed the Order with Type “Take Away” and Display Take Away Panel

3.2.7 Dine In Panel

Figure 3.10 shows the panel displayed when the customer decided to dine in their order. Customer is required to fill in their table number before they can proceed their order. Table 3.10 describes the function of each element in Figure 3.10.

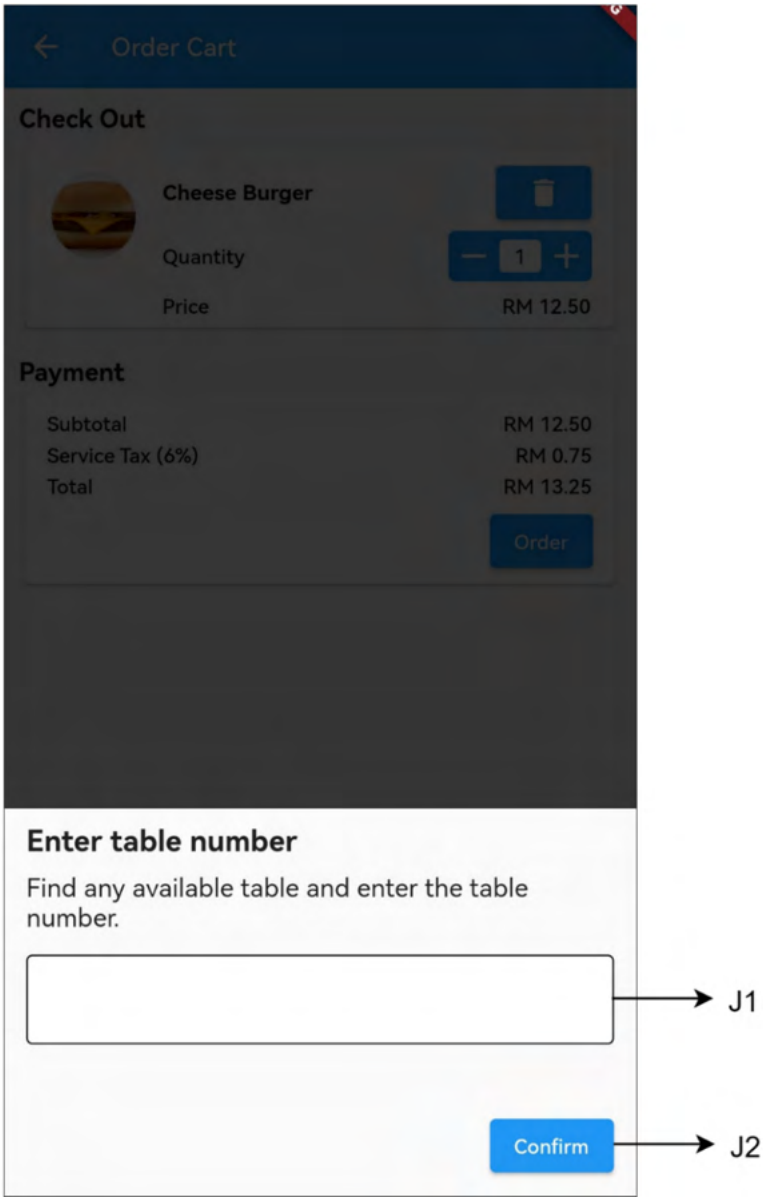

Figure 3.10 Dine In Panel

Table 3.10 Description of Elements in Dine In Panel

Label	Description
J1	Input Field for Table Number of the Customer
J2	Save the Dine In Order with Value in J1 and Redirect to Order Message Page

3.2.8 Take Away Panel

Figure 3.11 shows the panel displayed when the customer decided to take away their order. The customer needs to select the time they want to pick up the order before the order is recorded into the system. Table 3.11 shows the description of each element in Figure 3.11.

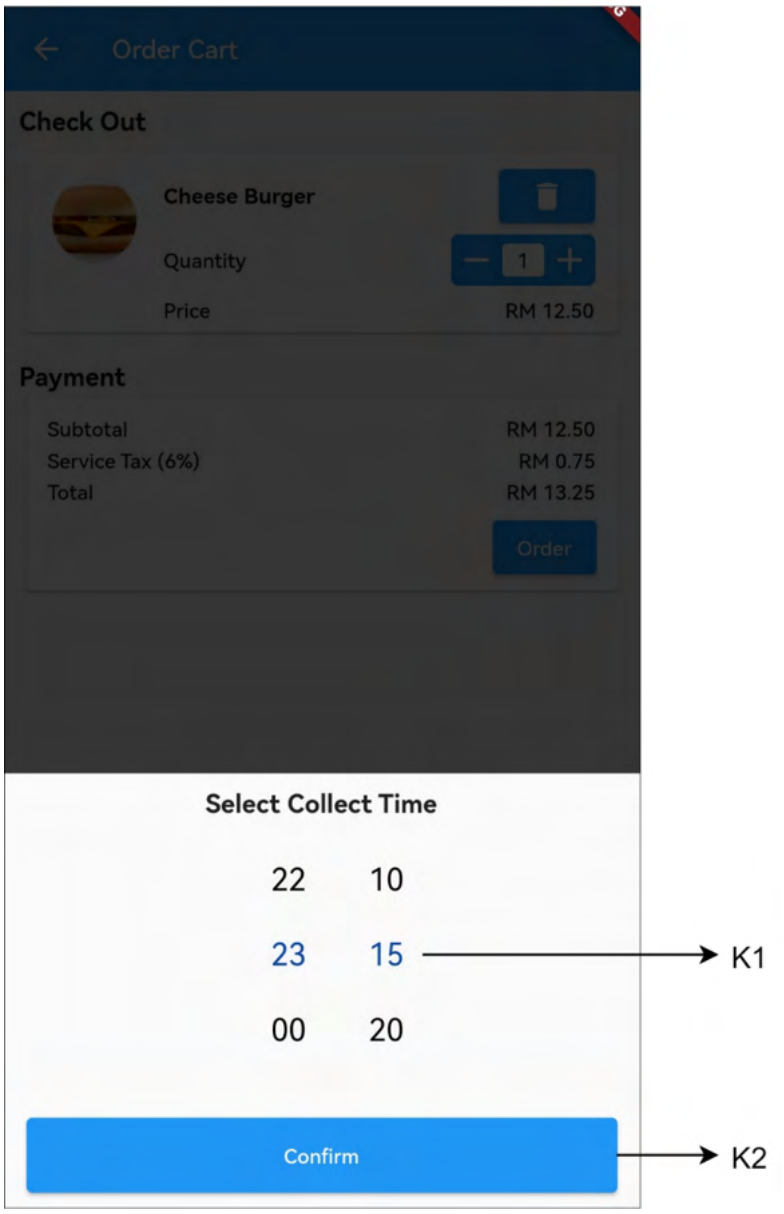

Figure 3.11 Take Away Panel

Table 3.11 Description of Elements in Take Away Panel

Label	Description
K1	Time Picker for Customer Select Collect Time of Take Away Order
K2	Save the Take Away Order with Value in K1 and Redirect to Order Message Page

3.2.9 Order Message Page

Figure 3.12 illustrates the order message page displayed to the customer after the order is successfully placed. The summary of the order placed will also be displayed to the customer. If the order type is “Dine In”, the table number will be displayed in the order summary. If the order type is “Take Away”, the collect time will be displayed in the order summary. Table 3.12 describe the function of each element In Figure 3.12.

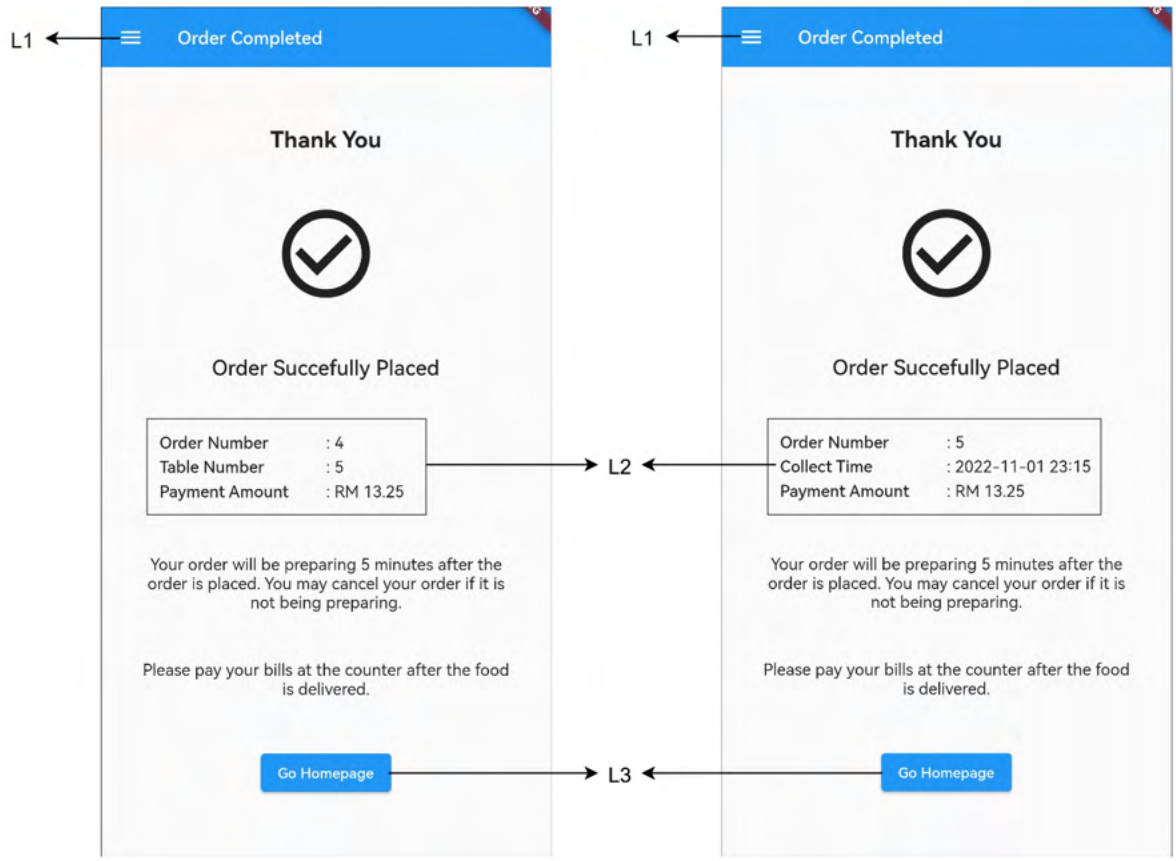

Figure 3.12 Order Message Page

Table 3.12 Description of Elements in Order Message Page

Label	Description
L1	Access Customer Navigation Menu
L2	Summary of the Order Placed
L3	Redirect to Customer Homepage

3.2.10 Order Status Page

Figure 3.13 illustrates the order status page when the customer accesses the view order status module. Customer can choose to view the details of the order and also cancel the order. The “Cancel Order” button labelled with “M4” will only be displayed if the current order status is “On Queue”. Table 3.13 explains the function of each element in Figure 3.13.

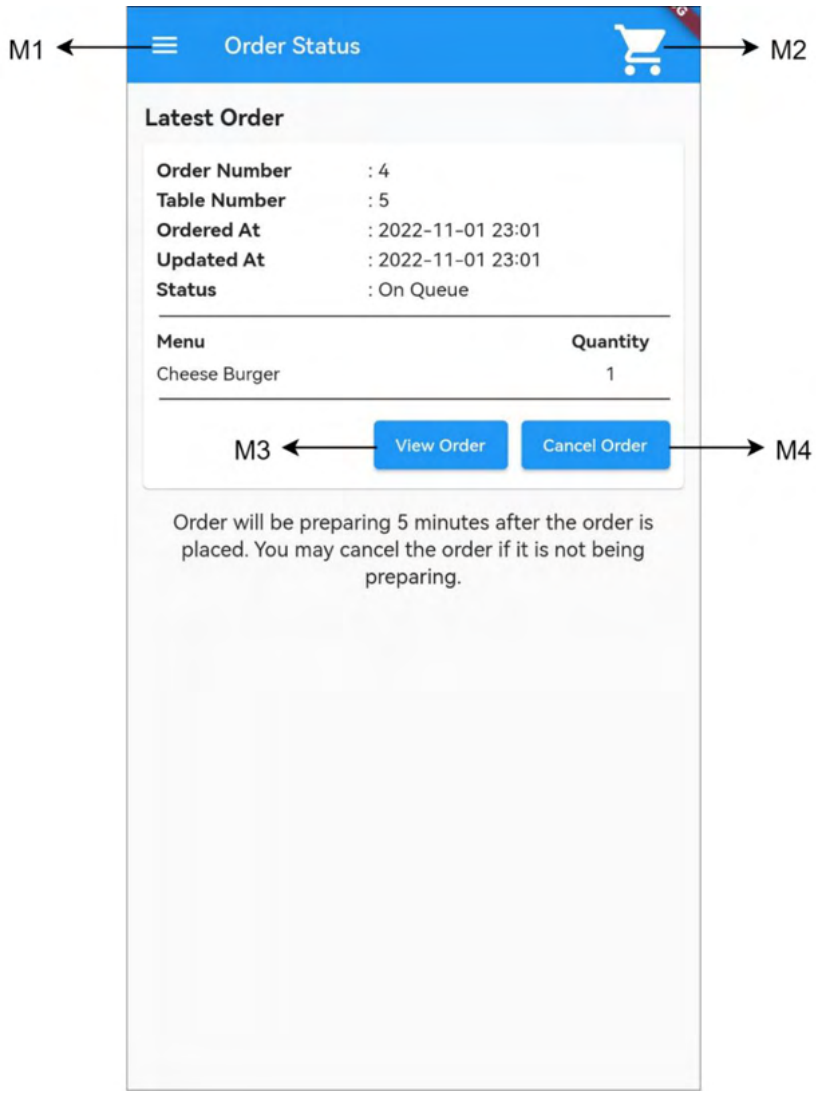

Figure 3.13 Order Status Page

Table 3.13 Description of Elements in Order Status Page

Label	Description
M1	Access Customer Navigation Menu
M2	Redirect to Order Cart Page of Place Order Module
M3	Redirect to Order Details Page of the Order
M4	Display Cancel Order Panel

3.2.11 Cancel Order Panel

Figure 3.14 depicts the cancel order panel when the customer wants to cancel their order. Table 3.14 describes the function of each element in Figure 3.14.

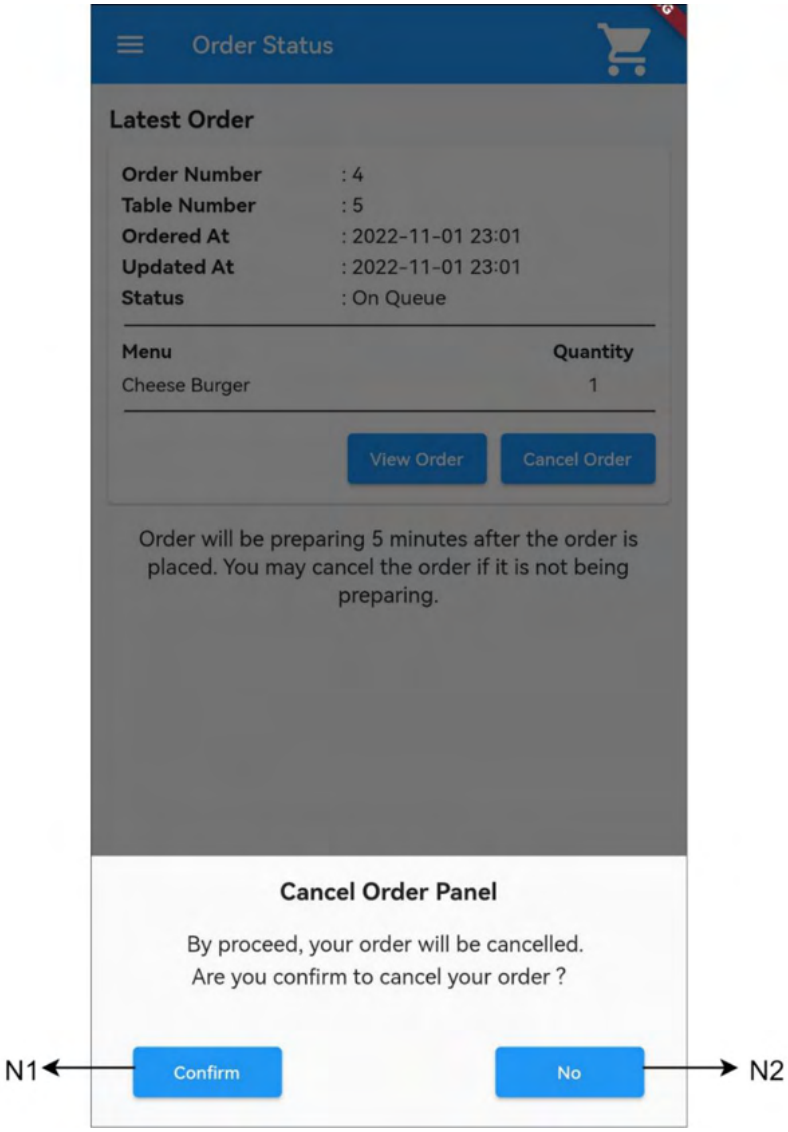

Figure 3.14 Cancel Order Panel

Table 3.14 Description of Elements in Cancel Order Panel

Label	Description
N1	Proceed to Cancel the Order and Update Order Status to “Cancelled”
N2	Abort Cancel Order Action

3.2.12 Order History Page

Figure 3.15 illustrates the order history page of view order history module. Customer can view all their order history in this page as well as performing the reorder action on their completed order. The “Re-Order” button labelled with Q4 will only appear if the order status is “Completed” or “Cancelled”. Table 3.15 shows the description of the function of each element in Figure 3.15.

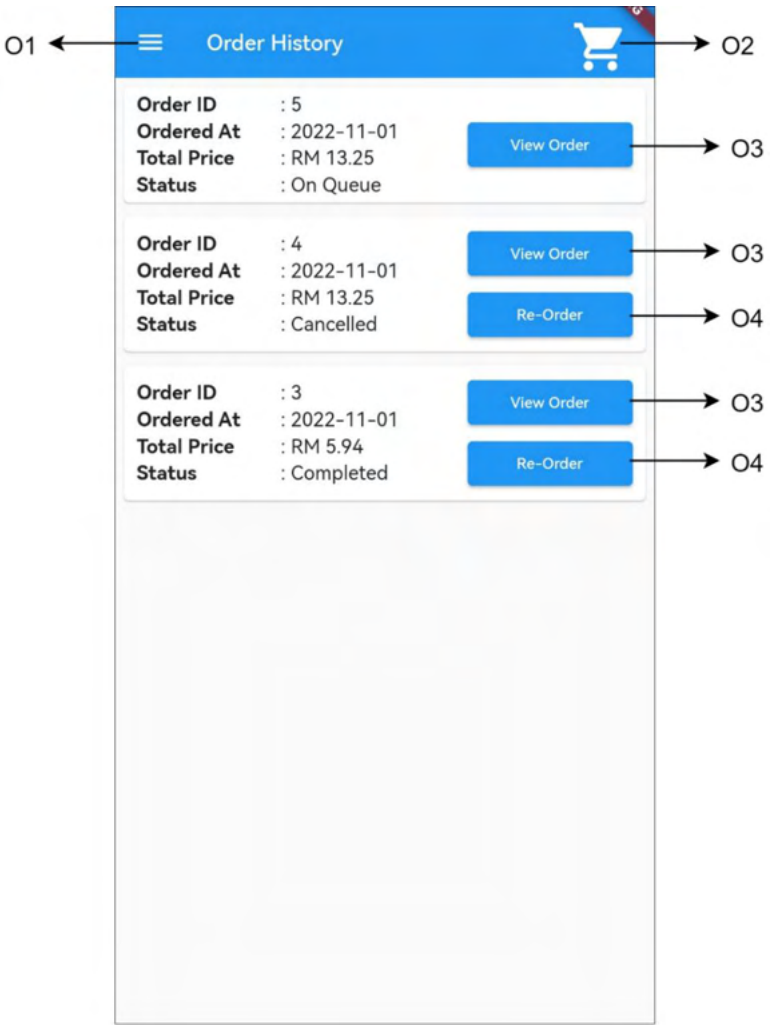

Figure 3.15 Order History Page

Table 3.15 Description of Elements in Order History Page

Label	Description
O1	Access Customer Navigation Menu
O2	Redirect to Order Cart Page of Place Order Module
O3	Redirect to Order Details Page of the Order
O4	Add the Menu of the Selected Past Order to Order Cart and Redirect to Order Cart Page

3.2.13 Order Details Page

Figure 3.16 shows the order details page of an order. This page will be displayed when the customer wants to view the details of an order. Same as Figure 3.15, the “Re-Order” button labelled with P4 will only appear if the order status is “Completed” or “Cancelled”. Table 3.16 describes the function of each element in Figure 3.16.

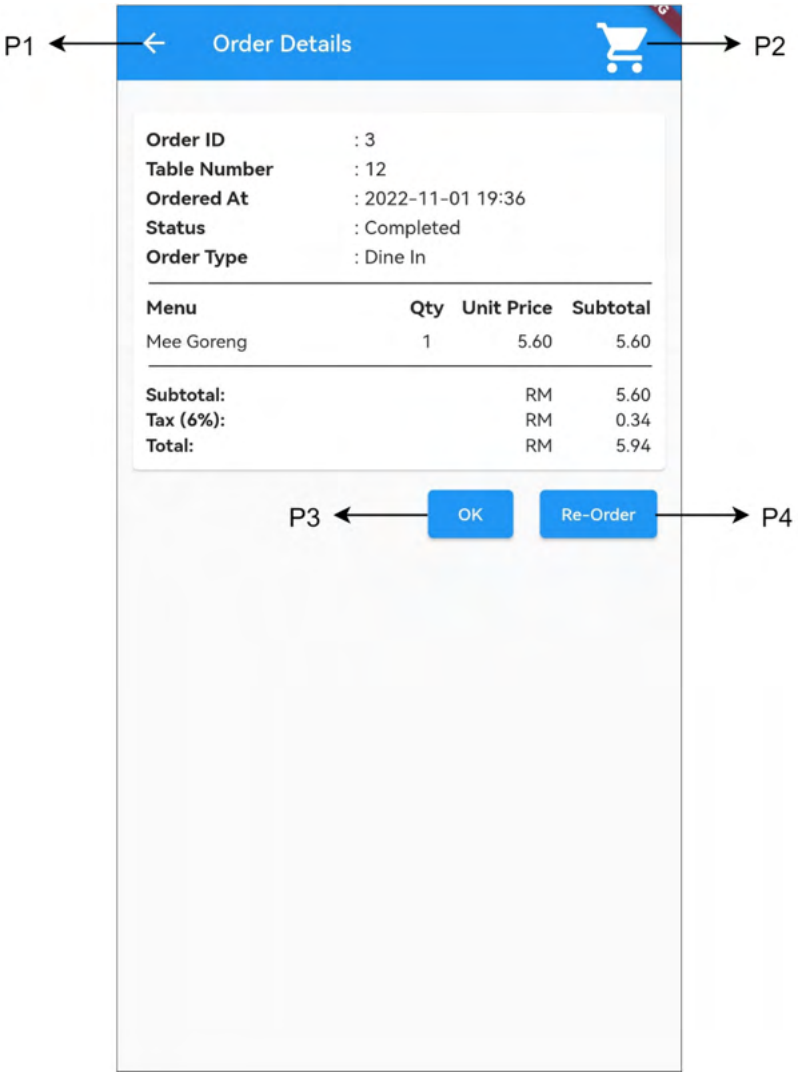

Figure 3.16 Order Details Page

Table 3.16 Description of Elements in Order Details Page

Label	Description
P1	Return to Previous Page
P2	Redirect to Order Cart Page of Place Order Module
P3	Return to Previous Page
P4	Add the Menu of the Selected Past Order to Order Cart and Redirect to Order Cart Page

3.2.14 Feedback Form Page

Figure 3.17 depicts the feedback form page of provide feedback module. Customer is required to select their rating and fill their comments before submitting the feedback. Table 3.17 shows the description of function of each element in Figure 3.17.

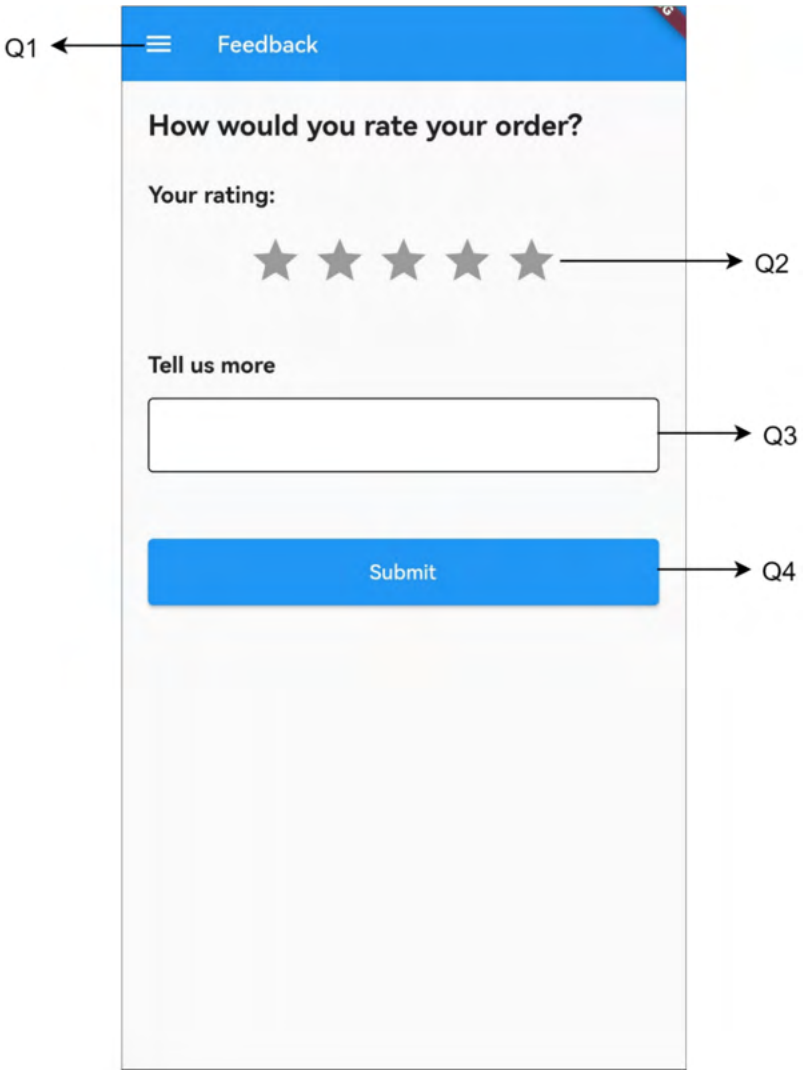

Figure 3.17 Feedback Form Page

Table 3.17 Description of Elements in Feedback Form Page

Label	Description
Q1	Access Customer Navigation Menu
Q2	Input Field for Customer Select Rating
Q3	Input Field for Customer Comments
Q4	Save Values of Q2 and Q3 as Customers' Feedback and Redirect to Thank You Message Page

3.2.15 Thank You Message Page

Figure 3.18 illustrates the thank you message page which will be displayed after the customer submitted their feedback. Table 3.18 explain the function of each element in Figure 3.18.

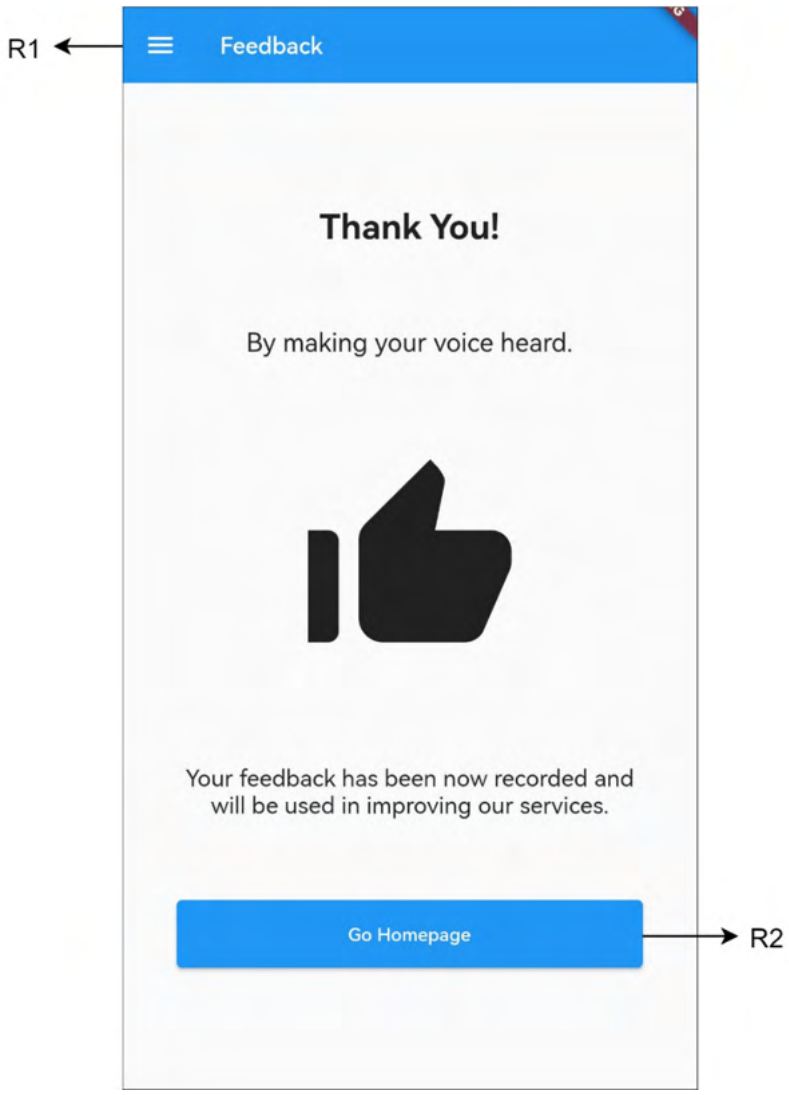

Figure 3.18 Thank You Message Page

Table 3.18 Description of Elements in Thank You Message Page

Label	Description
R1	Access Customer Navigation Menu
R2	Redirect to Customer Homepage

3.3 Business Owner Interfaces

3.3.1 Business Owner Homepage

Figure 3.19 shows the home page when the business owner logged in to the system. Business owner can access the view sales report module and view feedback module by pressing the button displayed on this page. Table 3.19 shows the description of the function of each element in Figure 3.19.

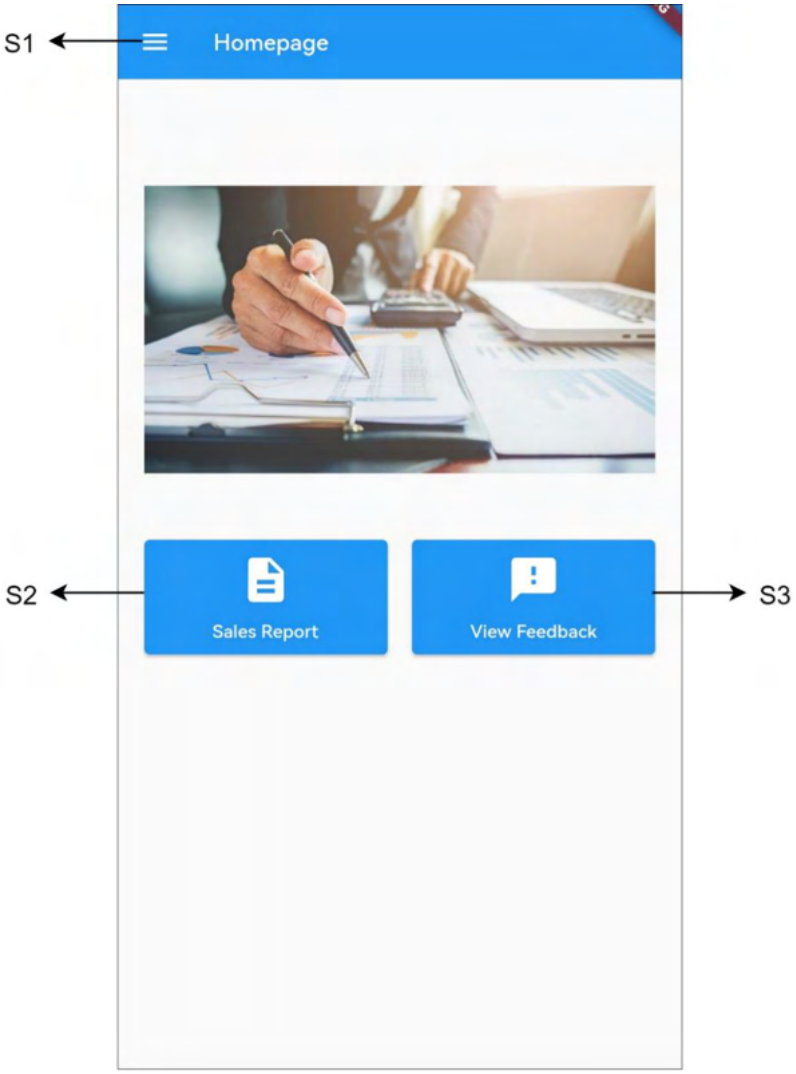

Figure 3.19 Business Owner Homepage

Table 3.19 Description of Elements in Business Owner Homepage

Label	Description
S1	Access Business Owner Navigation Menu
S2	Redirect to Sales Report List Page of View Sales Report Module
S3	Redirect to Feedback List Page of View Feedback Module

3.3.2 Business Owner Navigation Menu

Figure 3.20 illustrates the navigation menu of business owner. Business owner can also use this navigation menu to access the homepage, view sales report module, view feedback module and logout the system. Table 3.20 explains the function of each element in Figure 3.20.

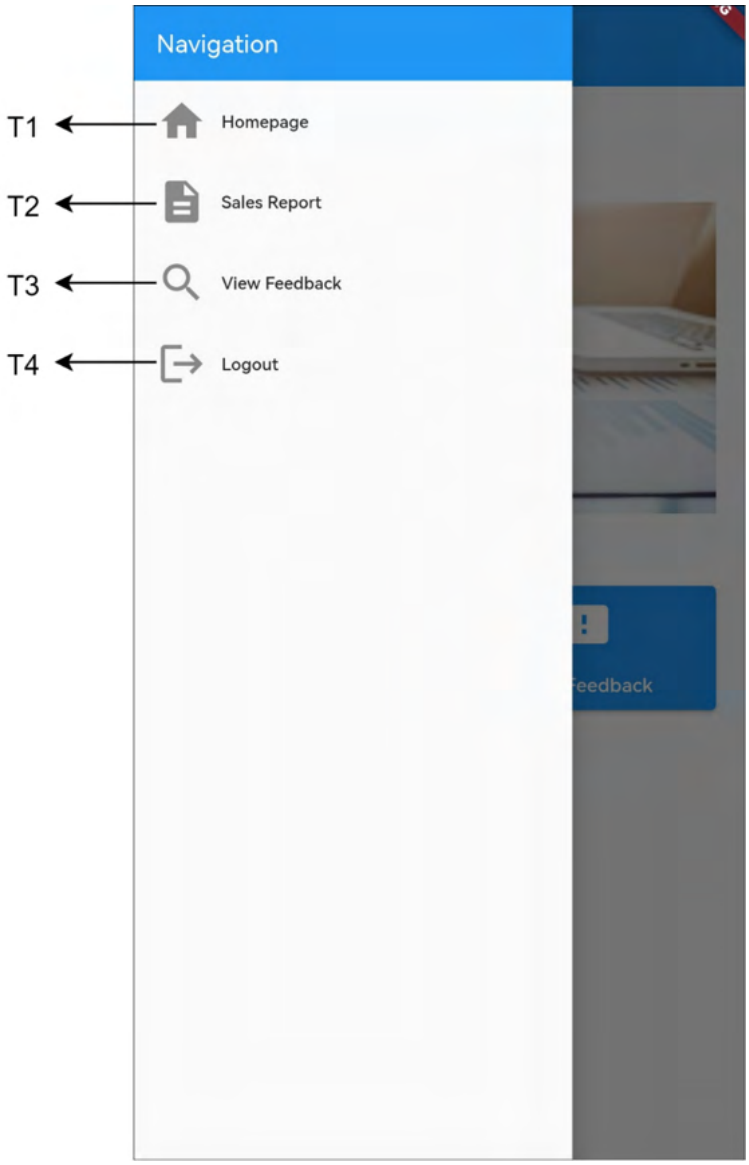

Figure 3.20 Business Owner Navigation Menu

Table 3.20 Description of Elements in Business Owner Navigation Menu

Label	Description
T1	Redirect to Business Owner Homepage
T2	Redirect to Sales Report List Page of View Sales Report Module
T3	Redirect to Feedback List Page of View Feedback Module
T4	Logout Business Owner and Redirect to Login Page

3.3.3 Sales Report List Page

Figure 3.21 shows the sales report list page of view sales report module. The business owner will be able to see a list of sales report in this page and they can press the "View Detail" button that labelled with U2 to view the details of the report. Table 3.21 shows the description of the function of each element in Figure 3.21.

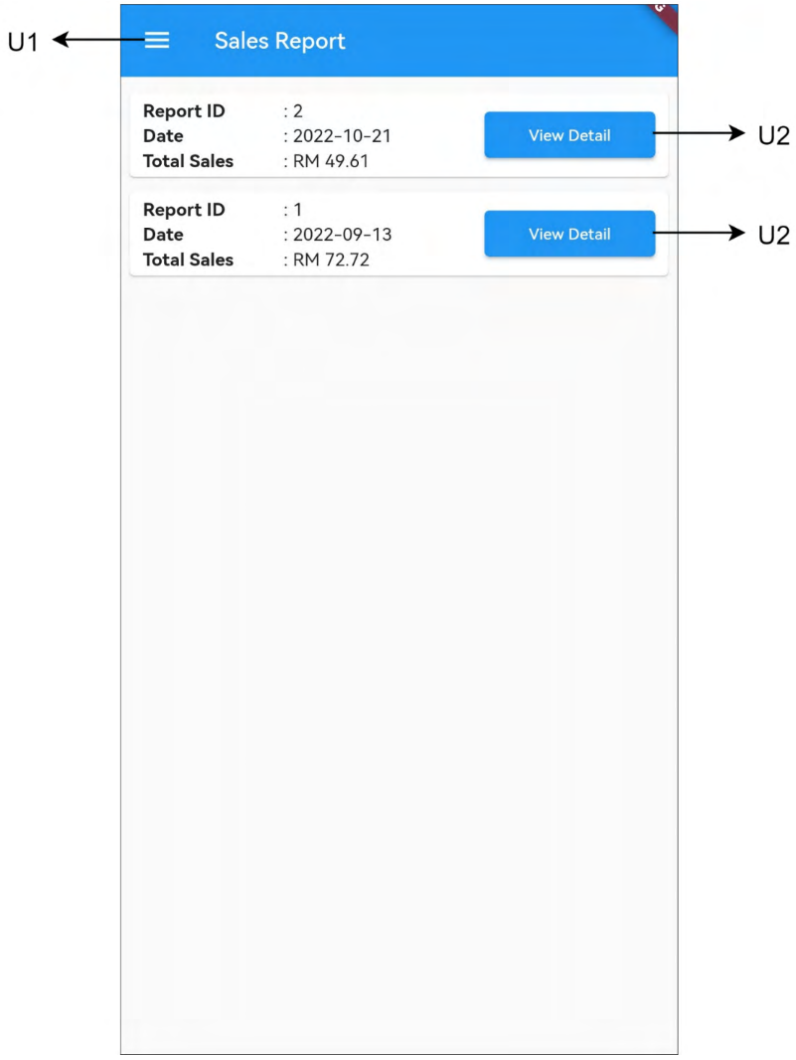

Figure 3.21 Sales Report List Page

Table 3.21 Description of Elements in Sales Report List Page

Label	Description
U1	Access Business Owner Navigation Menu
U2	Redirect to Sales Report Details Page of View Sales Report Module

3.3.4 Sales Report Details Page

Figure 3.22 shows the sales report details page of a report. This page is displayed after the business owner selected a report to be viewed. They will be able to view and evaluate the daily performance of the restaurant. Table 3.22 describes the function of each element in Figure 3.22.

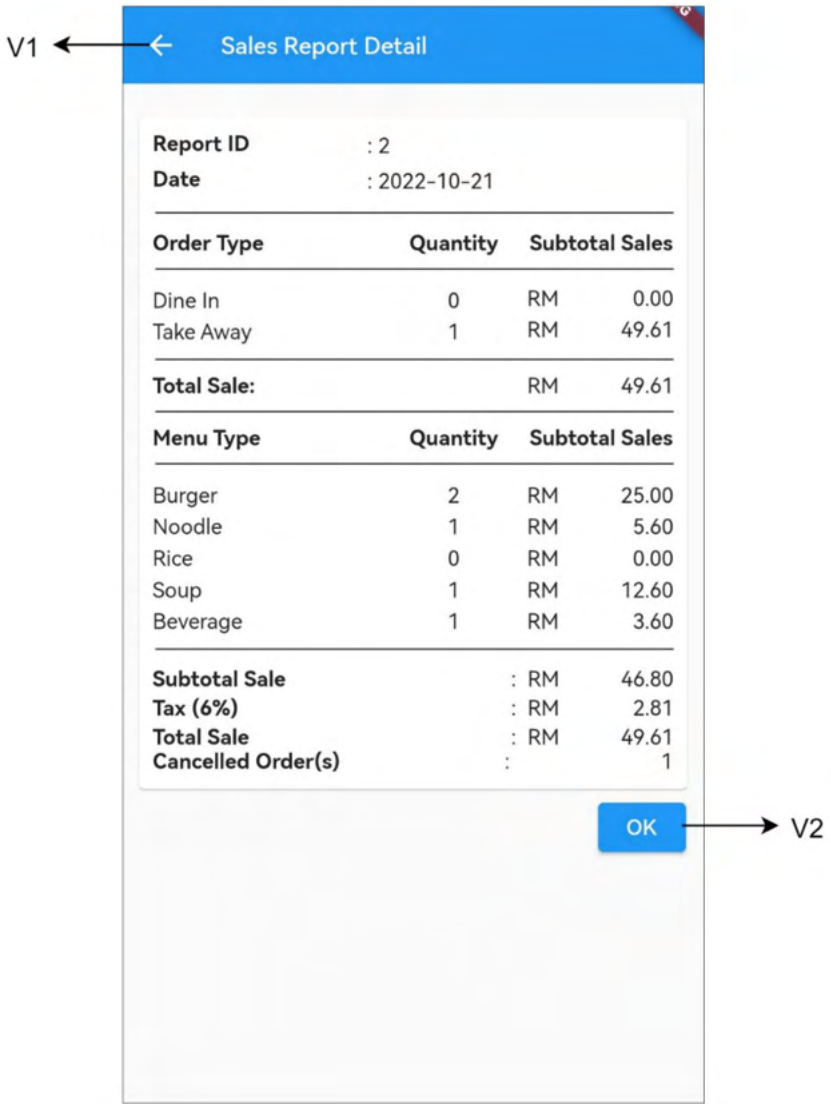

Figure 3.22 Sales Report Details Page

Table 3.22 Description of Elements in Sales Report Details Page

Label	Description
V1	Return to Previous Page
V2	Return to Previous Page

3.3.5 Feedback List Page

Figure 3.23 shows the feedback list page of view feedback module. The feedback will be displayed in a list with descending order. The business owner can also filter the customers' feedback by the selecting a range of dates. Table 3.23 shows the description of the function of each element in Figure 3.23.

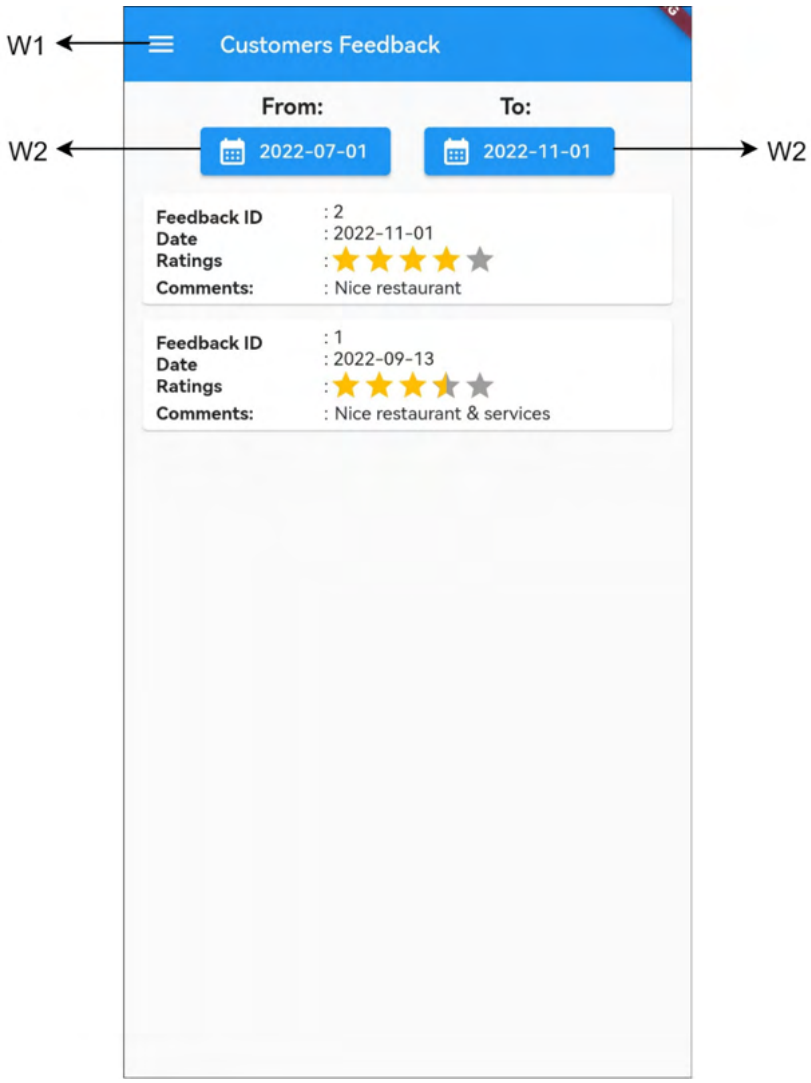

Figure 3.23 Feedback List Page

Table 3.23 Description of Elements in Feedback List Page

Label	Description
W1	Access Business Owner Navigation Menu
W2	Redirect to Select Date Range Page and Select Date Range

3.3.6 Select Date Range Page

Figure 3.24 illustrates the select date range page displayed when the business owner wants to filter the feedback. The business owner needs to select a start date and end date to filter the customers' feedback. Table 3.24 explains the function of each element in Figure 3.24.

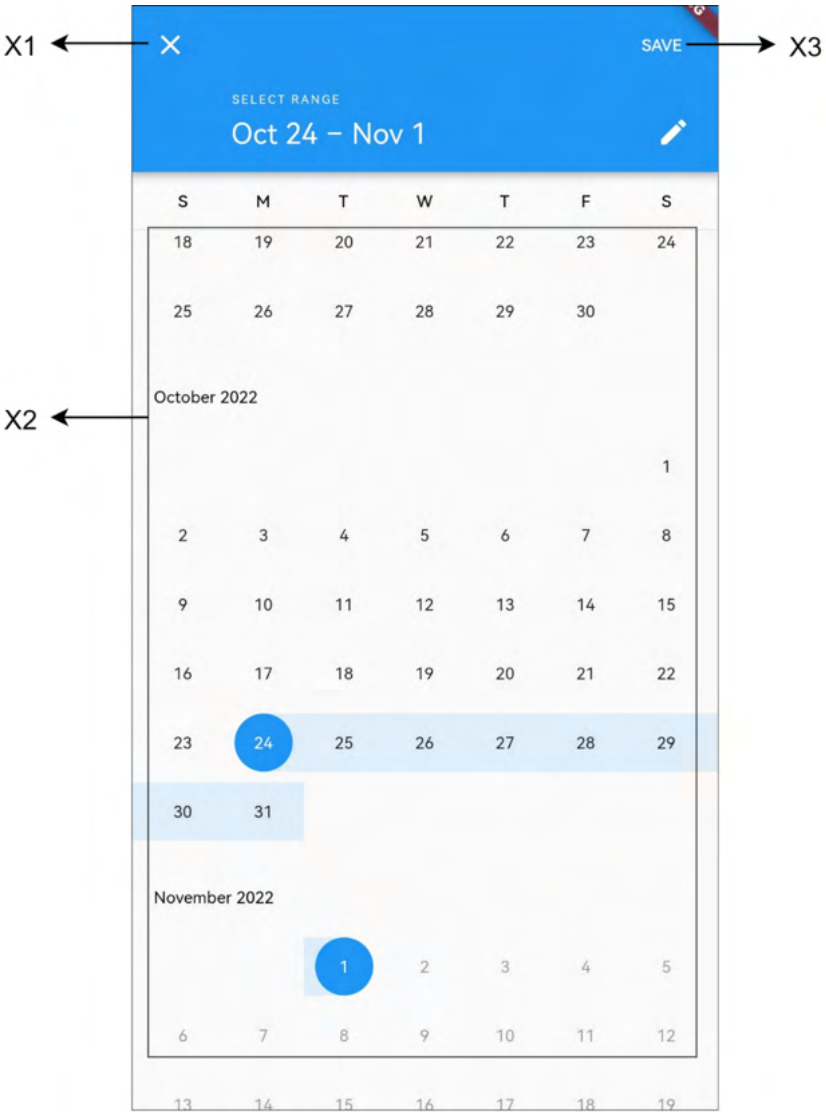

Figure 3.24 Select Date Range Page

Table 3.24 Description of Elements in Select Date Range Page

Label	Description
X1	Close Select Date Range Page
X2	Date Inputs to be Selected by Business Owner
X3	Return Selected Date Range and Display Filtered Feedback

3.4 Kitchen Staff Interfaces

3.4.1 Kitchen Staff Order List Page

Figure 3.25 shows the order list page for kitchen staff. This page will be displayed as the homepage of the kitchen staff. The kitchen staff will be able to view the queuing order and today's cancelled order. They need to update the order status by pressing the "Update Status" button if the prerequisite is satisfied. Only the order with status "On Queue" and "Preparing" will be displayed in this page. If the current order status is "Preparing", the order will have an orange colour background. Table 3.25 shows the explanation of each element in Figure 3.25.

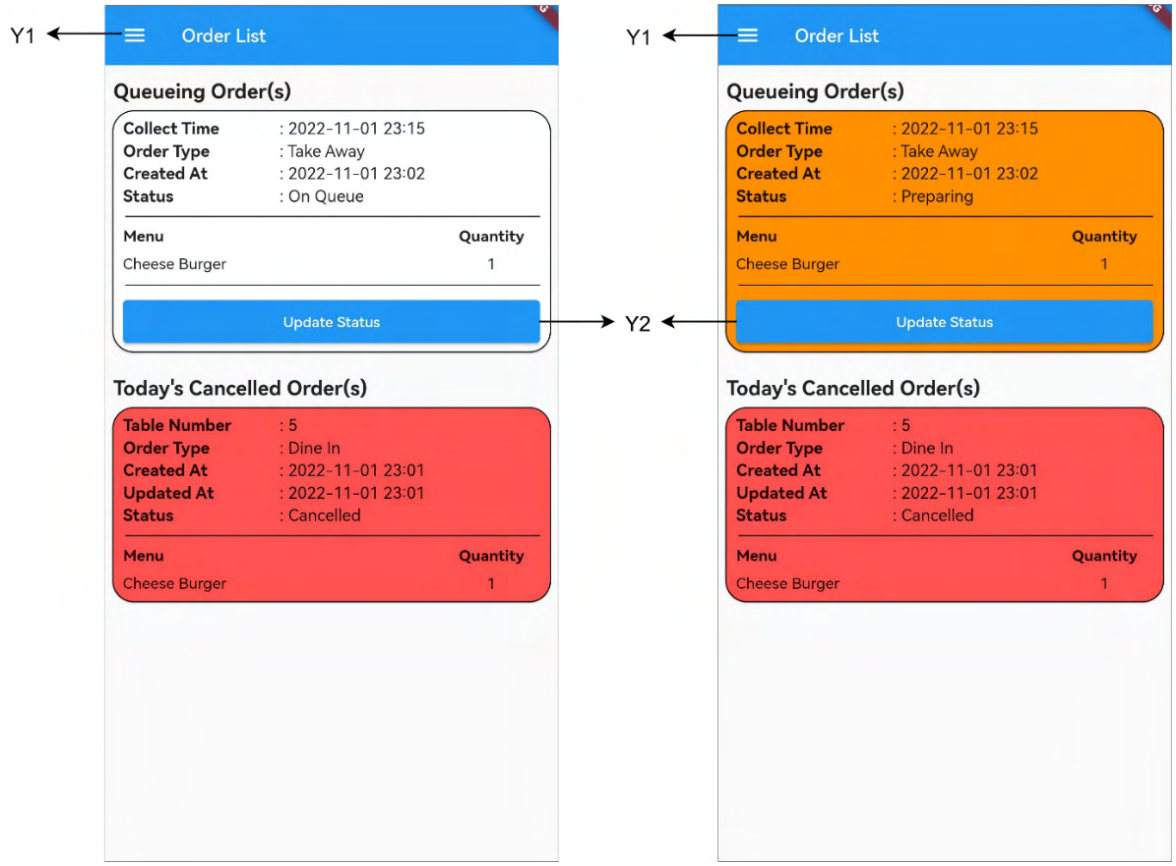

Figure 3.25 Kitchen Staff Order List Page

Table 3.25 Description of Elements in Kitchen Staff Order List Page

Label	Description
Y1	Access Kitchen Staff Navigation Menu
Y2	Display Kitchen Staff Update Order Status Panel According to the Current Status

3.4.2 Kitchen Staff Navigation Menu

Figure 3.26 illustrates the navigation menu of the kitchen staff. The kitchen staff can access the order list by pressing the button and logout the system. Table 3.26 describes the function of each element in Figure 3.26.

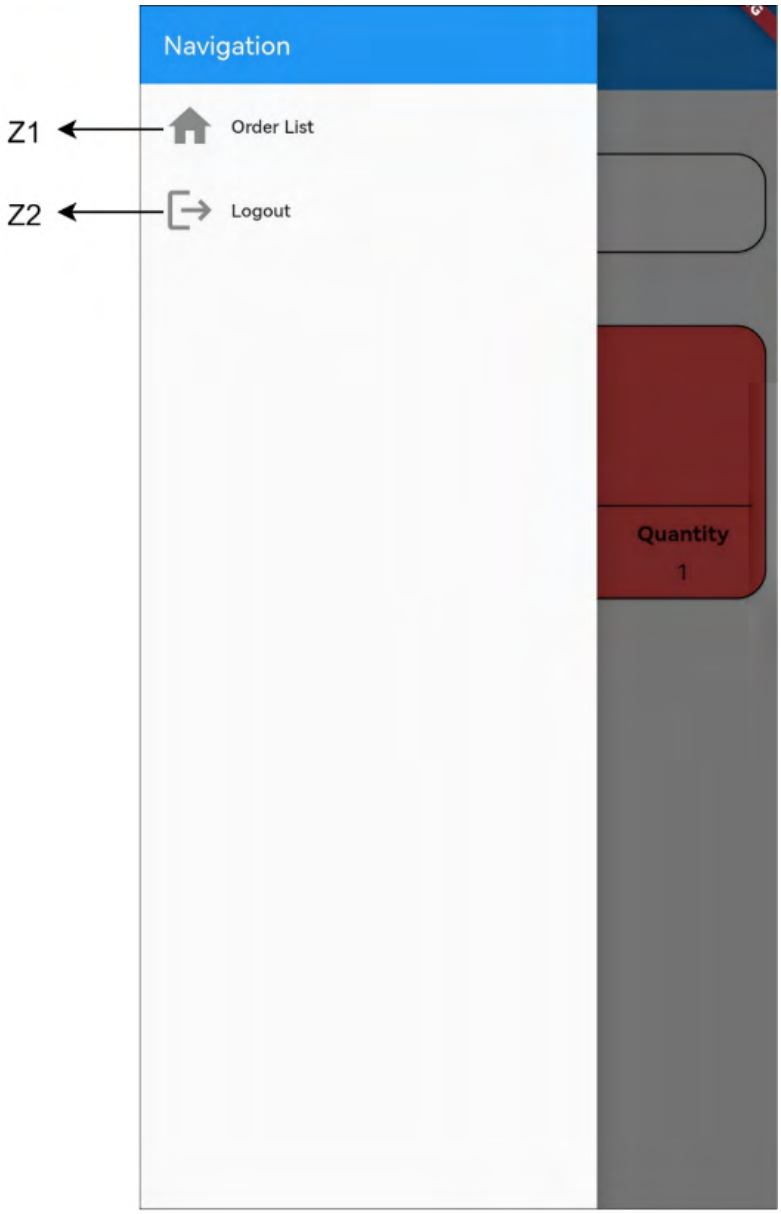

Figure 3.26 Kitchen Staff Navigation Menu

Table 3.26 Description of Elements in Kitchen Staff Navigation Menu

Label	Description
Z1	Redirect to Kitchen Staff Order List Page
Z2	Logout Kitchen Staff and Redirect to Login Page

3.4.3 Kitchen Staff Update Order Status Panel

Figure 3.27 depicts the update order status panel of kitchen staff. This panel will be displayed to the kitchen staff when they pressed the “Update Status button” in Figure 3.25. There are two types of update order status panel which is the panel that update order status from “On Queue” to “Preparing” and the panel that update order status from “Preparing” to “To Be Serve”. Table 3.27 explains the function of each element in Figure 3.27.

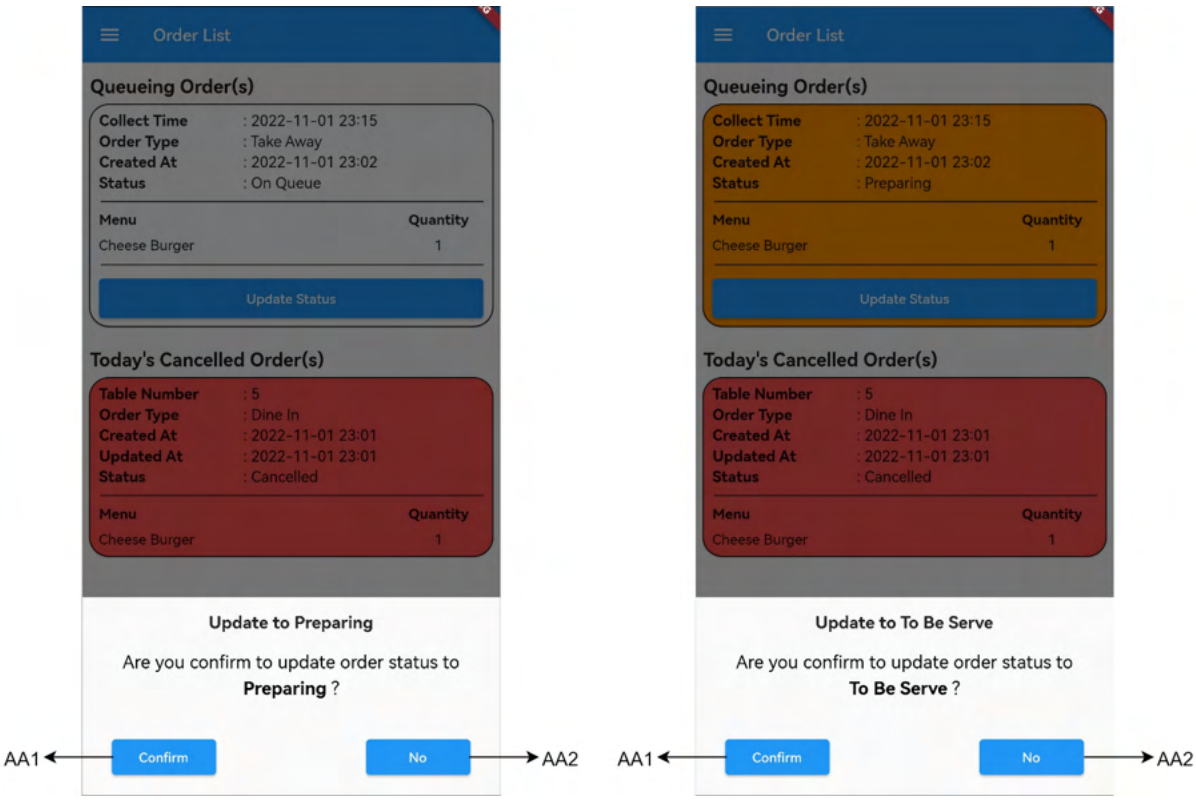

Figure 3.27 Kitchen Staff Update Order Status Panel

Table 3.27 Description of Elements in Kitchen Staff Update Order Status Panel

Label	Description
AA1	Update Order Status to “Preparing” If Current Status is “On Queue” or Update Order Status tot “To Be Serve” If Current Status is “Preparing”
AA2	Close Kitchen Staff Update Order Status Panel

3.5 Waiter Interfaces

3.5.1 Waiter Delivery List Page

Figure 3.28 illustrates the homepage of waiter which is the waiter delivery list page. The waiter can view the queuing order and today's cancelled order. They need to update the order status by pressing the "Update Status" button if the prerequisite is satisfied. Only order with status "To Be Serve" will be displayed in this page. Table 3.28 shows the description of the function of each element in Figure 3.28.

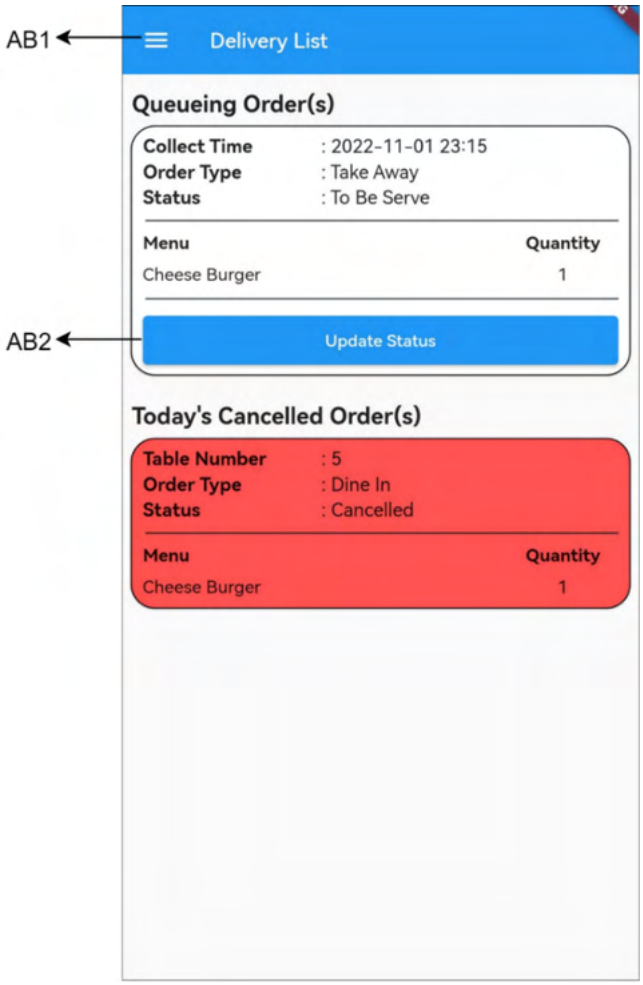

Figure 3.28 Waiter Delivery List Page

Table 3.28 Description of Elements in Waiter Delivery List Page

Label	Description
AB1	Access Waiter Navigation Menu
AB2	Display Waiter Update Order Status Panel

3.5.2 Waiter Navigation Menu

Figure 3.29 depicts the waiter navigation menu. Waiter can redirect to the homepage which is the delivery list by pressing the button labelled with “AC1” or logout by pressing the logout button. Table 3.29 describes the function of each element in Figure 3.29.

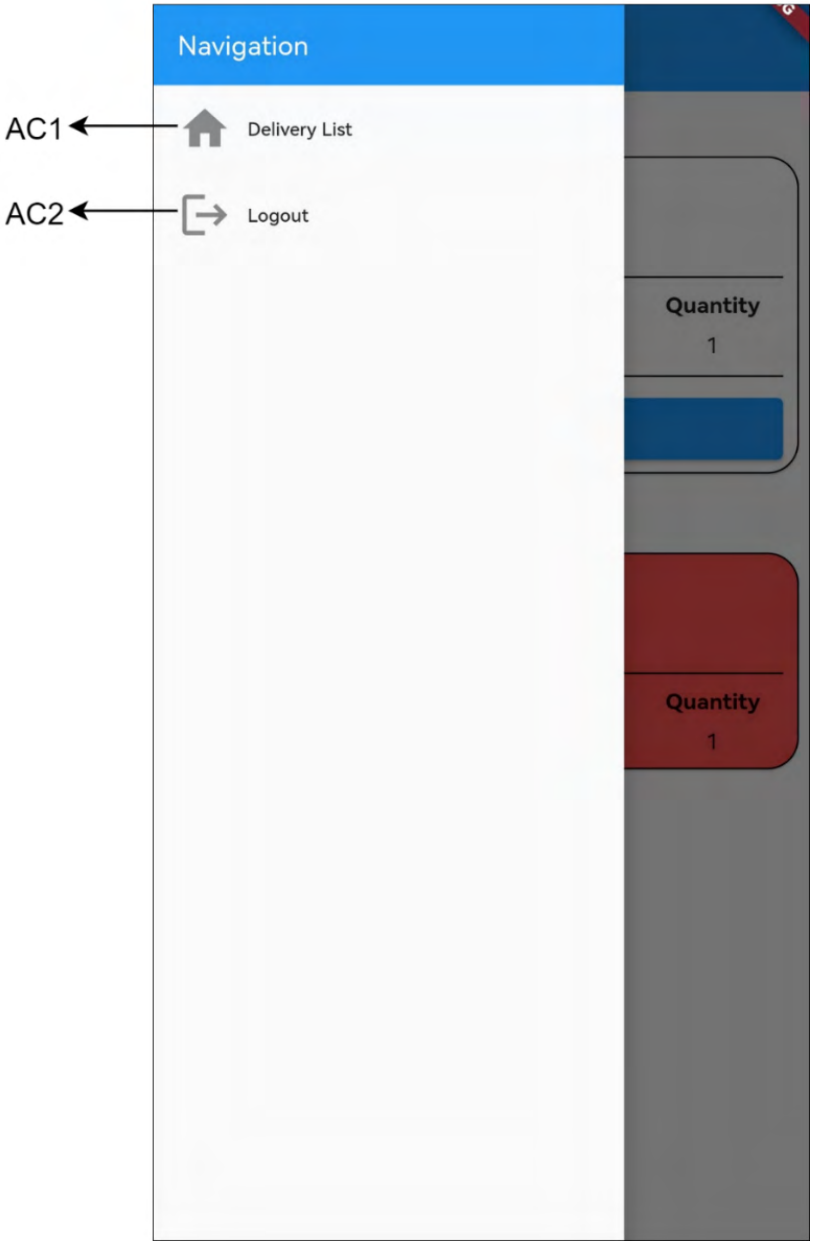

Figure 3.29 Waiter Navigation Menu

Table 3.29 Description of Elements in Waiter Navigation Menu

Label	Description
AC1	Redirect to Waiter Delivery List Page
AC2	Logout Waiter and Redirect to Login Page

3.5.3 Waiter Update Order Status Panel

Figure 3.30 shows the update order status panel for waiter after they pressed the "Update Status" button in Figure 3.28. The waiter needs to confirm the update of order status from "To Be Serve" to "Delivered" by pressing the "Confirm" button labelled with AD1 in Figure 3.30. Table 3.30 explains the function of each element in Figure 3.30.

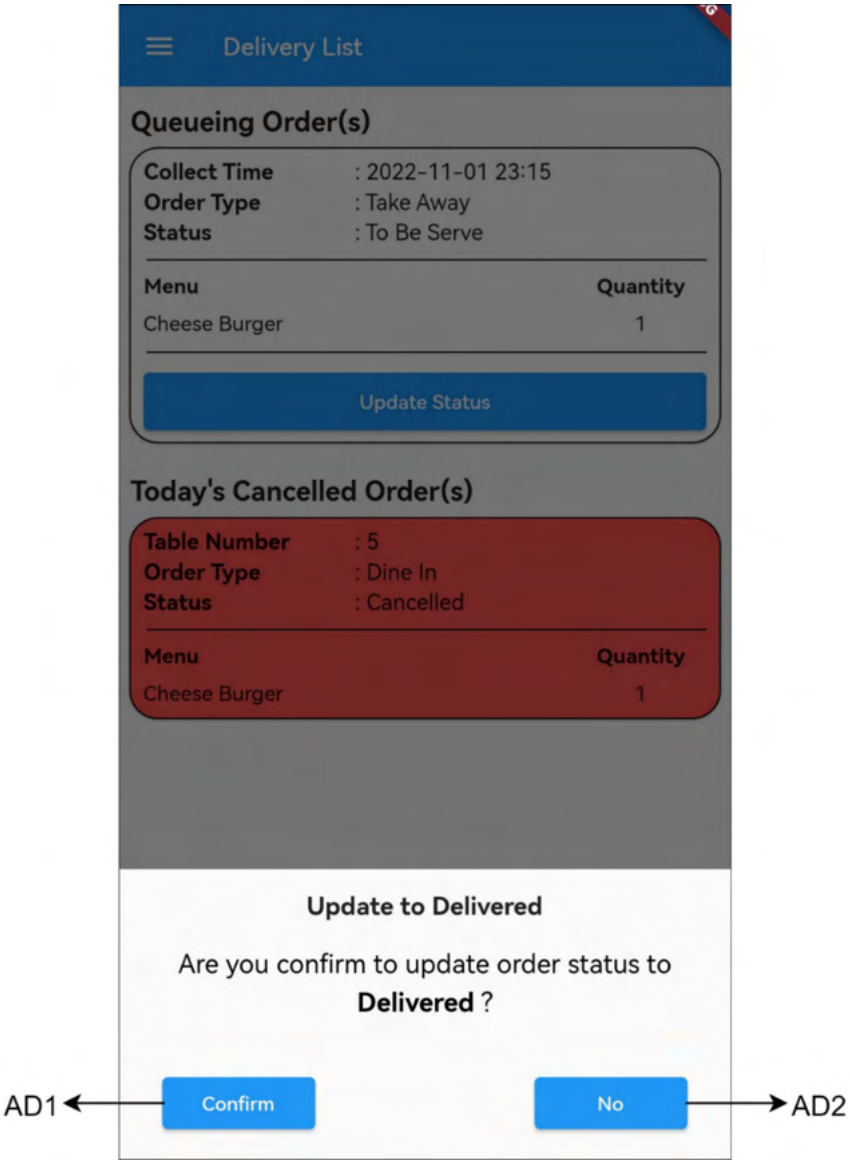

Figure 3.30 Waiter Update Order Status Panel

Table 3.30 Description of Elements in Waiter Update Order Status Panel

Label	Description
AD1	Update Order Status to "Delivered"
AD2	Close Waiter Update Order Status Panel

3.6 Cashier Interfaces

3.6.1 Cashier Payment List Page

Figure 3.31 illustrates the cashier payment list page. The cashier will be able to view the queueing order and today's cancelled order. Only order with status "Delivered" will be displayed to the cashier. They will be responsible to update the order status to "Completed" once they received the payment from the customer. Table 3.31 shows the explanation of the function of each element in Figure 3.31.

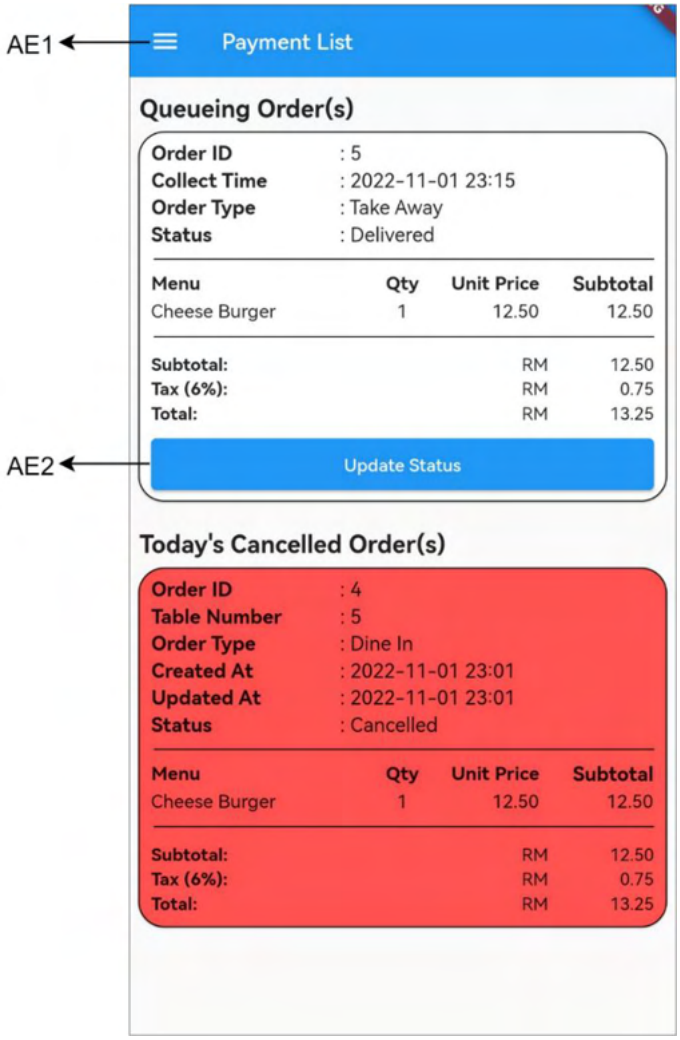

Figure 3.31 Cashier Payment List Page

Table 3.31 Description of Elements in Cashier Payment List Page

Label	Description
AE1	Access Cashier Navigation Menu
AE2	Display Cashier Update Order Status Panel

3.6.2 Cashier Navigation Menu

Figure 3.32 shows the navigation menu of the cashier. The cashier can be redirected to the payment list and logout by using this navigation menu. Table 3.32 describes the function of each element in Figure 3.32.

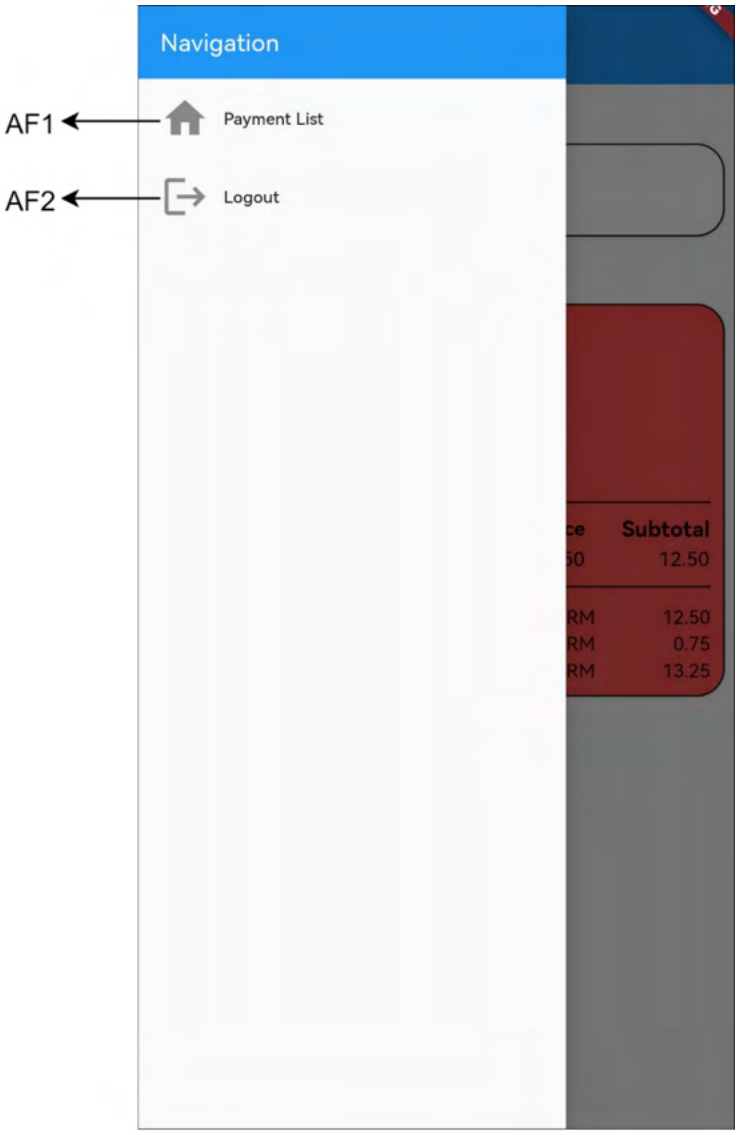

Figure 3.32 Cashier Navigation Menu

Table 3.32 Description of Elements in Cashier Navigation Menu

Label	Description
AF1	Redirect to Cashier Payment List Page
AF2	Logout Cashier and Redirect to Login Page

3.6.3 Cashier Update Order Status Panel

Figure 3.33 shows the cashier update order status panel which will appear when the cashier pressed the “Update Status” button in Figure 3.31. Table 3.33 shows the description of the function of each element in Figure 3.33.

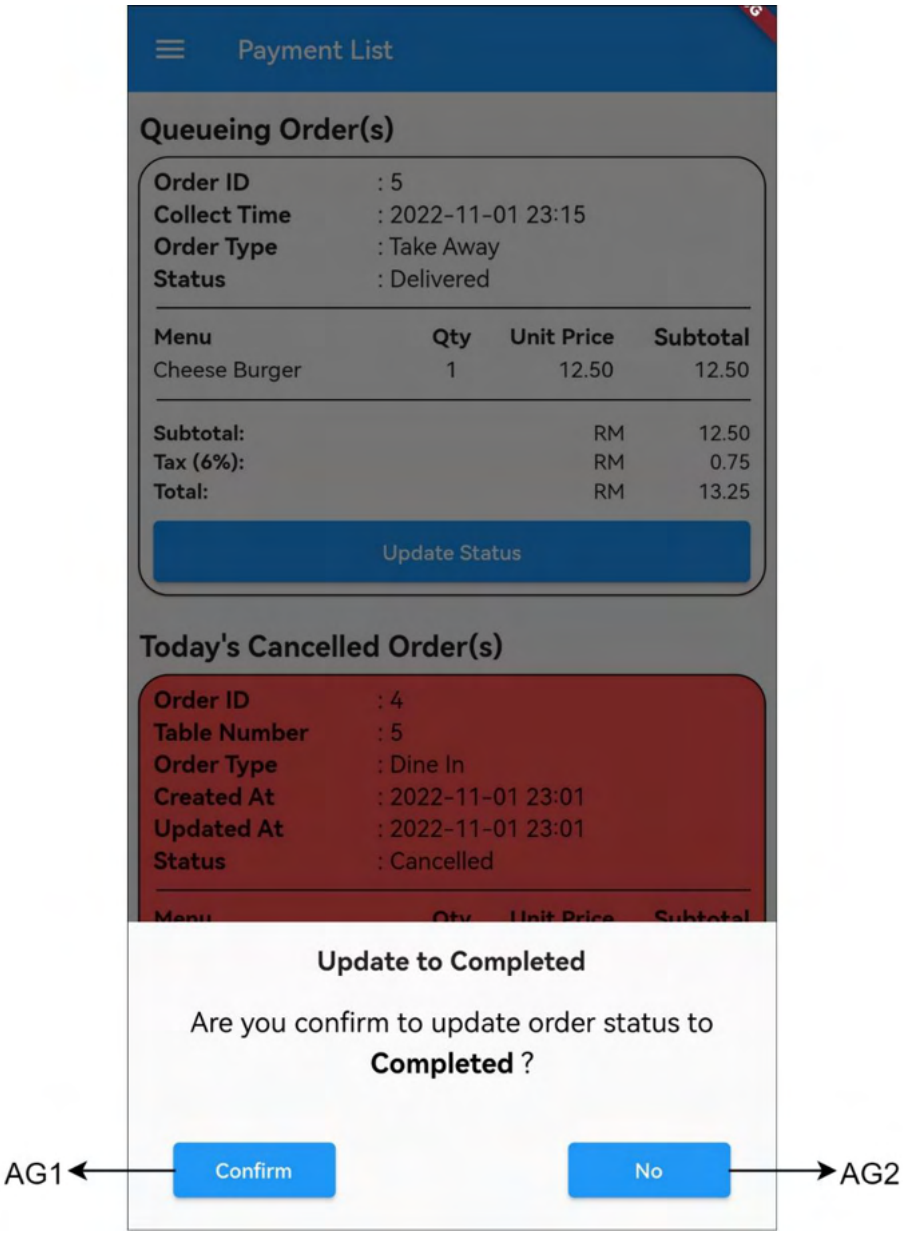

Figure 3.33 Cashier Update Order Status Panel

Table 3.33 Description of Elements in Cashier Update Order Status Panel

Label	Description
AG1	Update Order Status to “Completed”
AG2	Close Cashier Update Order Status Panel

APPENDIX D
USER ACCEPTANCE TESTING (UAT) FORM

For Appendices Heading, use TITLE AT ROMAN PAGES style.

TABLE OF CONTENT

1.0 Testing Report

This section will describe the User Acceptance Test (UAT) procedure which will be conducted to the developed application. Approval of this testing indicates the testers are sure that, the developed application will be regarded as having undergone comprehensive testing and being ready for implementation as the test plan has been performed. Table below shows the details of testers involved in UAT:

	Name	**User Type**
	Wong Sung Sum	Customer
	Tan Chee Meng	Business Owner
	Poon Ju Yen	Kitchen Staff
	Woon Chun Cheong	Waiter
	Ronald Lim Sheng Wei	Cashier

1.1 User Acceptance Testing (UAT) Form for Customer Role

Figure 1.1 shows the test case that will be performed by Wong Sung Sum as the customer of the restaurant.

User Acceptance Test Form for OSMR System (Customer)

Name: Wong Sung Sum Position/Occupation: Customer Date: 13-12-2022

No.	Event	Pre-conditions	Test Data/Steps	Expected Result	Pass/Fail (✓ - Pass, X – Fail)	Comments
Login Page						
1	Login an existing account	Account is not verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Verify Email page	✓	
2	Login an existing account	Account is verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Customer Homepage	✓	
3	Access Registration page		1. Press "Register" button	Redirected to Registration page	✓	
Registration Page						
1	Register an account		1. Fill in Required Data for Each Input Field 2. Press "Register" button	Redirected to Verify Email page	✓	
2	Access Login page		1. Press "Login" button	Redirected to Login page	✓	
Verify Email Page						
1	Resent verification email		1. Press "Resent Email" button	Verification email is received in registered email address	✓	
2	Logout / Terminate verification process		1. Press "Cancel" button	Redirected to Login page	✓	
Customer Homepage						
1	Access Navigation Menu		1. Press "Menu" icon button	Display Customer Navigation Menu	✓	
2	Access Order Cart page		1. Press "Cart" icon button	Redirected to Order Cart page	✓	
3	Access Place Order Module		1. Press "Place Order" button	Redirected to Menu List page	✓	
4	Access View Order Status Module		1. Press "Order Status" button	Redirected to Order Status page	✓	
5	Access Order History Module		1. Press "Order History" button	Redirected to Order History page	✓	
6	Access Provide Feedback Module		1. Press "Feedback" button	Redirected to Feedback Form page	✓	
Customer Navigation Menu						
1	Access Homepage		1. Press "Homepage" button	Redirected to Customer Homepage	✓	
2	Access Place Order Module		1. Press "Place Order" button	Redirected to Menu List page	✓	
3	Access View Order Status Module		1. Press "Order Status" button	Redirected to Order Status page	✓	
4	Access Order History Module		1. Press "Order History" button	Redirected to Order History page	✓	
5	Access Provide Feedback Module		1. Press "Feedback" button	Redirected to Feedback Form page	✓	
6	Logout		1. Press "Logout" button	Redirected to Login page	✓	
Place Order Module / Menu Type Page						
1	Access Menu List page of chosen menu type		1. Press any menu type label	Redirected to Menu List page of chosen menu type	✓	
Menu List Page						
1	Return to Menu Type Page		1. Press "Return" icon button	Redirected to Menu Type page	✓	
2	Add desired menu to order cart	Desired menu is not existed in order cart	1. Press "Add to Cart" button of desired menu	Number appends to the cart icon button increases / appear	✓	
Order Cart Page						
1	Return to Menu List Page		1. Press "Return" icon button	Redirected to Menu List page	✓	
2	Increase quantity of desired menu		1. Press "+" icon button	Quantity of desired menu increase by 1	✓	

Figure 1.1 User Acceptance Testing (UAT) Form for Customer Role (1)

Order Cart Page						
1	Return to Menu List Page		1. Press "Return" icon button	Redirected to Menu List page		
2	Increase quantity of desired menu		1. Press "+" icon button	Quantity of desired menu increase by 1		
3	Decrease quantity of desired menu	Current quantity of desired menu is greater than 1	1. Press "-" icon button	Quantity of desired menu decrease by 1		
4	Remove desired menu from order cart	Current quantity of desired menu is equal to 1	1. Press "-" icon button	Menu removed from order cart		
5	Remove desired menu from order cart		1. Press "Delete" icon button	Desired menu is removed from order cart		
6	Proceed the items in order cart as an order		1. Press "Order" button	Display Select Order Type panel		
Select Order Type Panel						
1	Proceed the order with type "Dine In"		1. Press "Dine In" button	Display Dine In panel		
2	Proceed the order with type "Take Away"		1. Press "Take Away" button	Display Take Away panel		
Dine In Panel						
1	Proceed the order with table number		1. Fill in the table number 2. Press "Confirm" button	Redirected to Order Message page		
Take Away Panel						
1	Proceed the order with collect time		1. Select collect time for the order 2. Press "Confirm" button	Redirected to Order Message page		
Order Message Page						
1	Return to Homepage		1. Press "Go Homepage" button	Redirected to Customer Homepage		
View Order Status Module / Order Status Page						
1	View order details		1. Press "View Order" button	Redirected to Order Details page		
2	Cancel Order	Current order status is "On Queue"	1. Press "Cancel Order" button	Display Cancel Order Panel		
Cancel Order Panel						
1	Proceed the cancel order action		1. Press "Confirm" button	1. Order status updated to "Cancelled" 2. Order removed from Order Status page		
2	Abort the cancel order action		1. Press "No" button	Close Cancel Order Panel		
Order History Page						
1	View Order Details		1. Press "View Order" button of desired order	Redirected to Order Details page of desired order		
2	Re-order	Current order status is "Cancelled" or "Completed"	1. Press "Re-Order" button	Redirected to Order Cart page with menu of previous order		
Order Details Page						
1	Return to previous page		1. Press "Return" icon button OR 1. Press "OK" button	Redirected to previous page		
2	Re-order	Current order status is "Cancelled" or "Completed"	1. Press "Re-Order" button	Redirected to Order Cart page with menu of previous order		
Feedback Form Page						
1	Submit Feedback		1. Select rating 2. Fill in comments 3. Press "Submit" button	Redirected to Feedback Message Page		
Feedback Message Page						
1	Return to Homepage		1. Press "Go Homepage" button	Redirected to Customer Homepage		

Figure 1.2 User Acceptance Testing (UAT) Form for Customer Role (2)

Remarks / Feedback

N/A

Signature:

Date: 13-12-2022

Figure 1.3 User Acceptance Testing (UAT) Form for Customer Role (3)

1.2 User Acceptance Testing (UAT) Form for Business Owner Role

Figure 1.2 shows the test case that will be performed by Tan Chee Meng for user role "Business Owner".

User Acceptance Test Form for OSMR System (Business Owner)

Name: Tan Chee Meng Position/Occupation: Business Owner Date: 3-12-2022

No.	Event	Pre-conditions	Test Data/Steps	Expected Result	Pass/Fail (✓ - Pass, X – Fail)	Comments
Login Page						
1	Login an existing account	Account is verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Business Owner Homepage	✓	
Business Owner Homepage						
1	Access Navigation Menu		1. Press "Menu" icon button	Display Business Owner Navigation Menu	✓	
2	Access View Sales Report Module		1. Press "Sales Report" button	Redirected to Sales Report List page	✓	
3	Access View Feedback Module		1. Press "View Feedback" button	Redirected to Feedback List page	✓	
Business Owner Navigation Menu						
1	Access Homepage		1. Press "Homepage" button	Redirected to Business Owner Homepage	✓	
2	Access View Sales Report Module		1. Press "Sales Report" button	Redirected to Sales Report List page	✓	
3	Access View Feedback Module		1. Press "View Feedback" button	Redirected to Feedback List page	✓	
4	Logout		1. Press "Logout" button	Redirected to Login page	✓	
Sales Report List Page						
1	View sales report details		1. Press "View Detail" button	Redirected to Sales Report Details page	✓	
Sales Report Details Page						
1	Return to previous page		1. Press "Return" icon button OR 1. Press "OK" button	Redirected to previous page	✓	
Feedback List Page						
1	Filter feedback according to date created		1. Press button labelled date	Display Select Date Range page	✓	
Select Date Range Page						
1	Select date range for filtering feedback		1. Select Start date 2. Select End date 3. Press "SAVE" button	Redirected to Feedback List Page with feedback filtered	✓	
2	Abort filter feedback action		1. Press "X" icon button	Redirected to Feedback List Page	✓	

Remarks / Feedback

--

Signature:

Date: 3-12-2022

Figure 1.4 User Acceptance Testing (UAT) Form for Business Owner Role

1.3 User Acceptance Testing (UAT) Form for Kitchen Staff Role

Figure 1.3 shows the test case that will be performed by Poon Ju Yen with the user role "Kitchen Staff".

User Acceptance Test Form for OSMR System (Kitchen Staff)

Name: Poon Ju Yen Position/Occupation: Kitchen Staff Date:27-11-2022

No.	Event	Pre-conditions	Test Data/Steps	Expected Result	Pass/Fail (✓ - Pass, X – Fail)	Comments
			Login Page			
1	Login an existing account	Account is verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Order List Page	✓	
			Order List Page			
1	Access Navigation Menu		1. Press "Menu" icon button	Display Kitchen Staff Navigation Menu	✓	
2	Update order status to "Preparing"	Current order status is "On Queue"	1. Press "Update Status" button	Display Update Order Status panel	✓	
3	Update order status to "To Be Serve"	Current order status is "Preparing"	1. Press "Update Status" button	Display Update Order Status panel	✓	
			Kitchen Staff Navigation Menu			
1	Access Order List page		1. Press "Order List" button	Redirected to Order List page	✓	
2	Logout		1. Press "Logout" button	Redirected to Login page	✓	
			Update Order Status Panel			
1	Update order status to "Preparing"	Current order status is "On Queue"	1. Press "Confirm" button	1. Order status updated to "Preparing" 2. Background of the order changed to orange colour	✓	
2	Update order status to "To Be Serve"	Current order status is "Preparing"	1. Press "Confirm" button	1. Order status updated to "To Be Serve" 2. Order removed from Order List page	✓ ✓	
3	Abort update order status action		1. Press "No" button	Close Update Order Status panel	✓	

Remarks / Feedback

No.

Signature:

Date: 27-11-2022

Figure 1.5 User Acceptance Testing (UAT) Form for Kitchen Staff Role

1.4 User Acceptance Testing (UAT) Form for Waiter Role

Figure 1.4 shows the test case that will be performed by Woon Chun Cheong with the user role "Waiter".

User Acceptance Test Form for OSMR System (Waiter)

Name: Woon Chun Cheong Position/Occupation: Waiter Date: 25-11-2022

No.	Event	Pre-conditions	Test Data/Steps	Expected Result	Pass/Fail (✓ - Pass, X – Fail)	Comments
Login Page						
1	Login an existing account	Account is verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Delivery List Page	✓	
Delivery List Page						
1	Access Navigation Menu		1. Press "Menu" icon button	Display Waiter Navigation Menu	✓	
2	Update order status to "Delivered"	Current order status is "To Be Serve"	1. Press "Update Status" button	Display Update Order Status panel	✓	
Waiter Navigation Menu						
1	Access Delivery List page		1. Press "Delivery List" button	Redirected to Delivery List page	✓	
2	Logout		1. Press "Logout" button	Redirected to Login page	✓	
Update Order Status Panel						
1	Update order status to "Delivered"	Current order status is "To Be Serve"	1. Press "Confirm" button	1. Order status updated to "Delivered" 2. Order removed from Delivery List page	✓	
2	Abort update order status action		1. Press "No" button	Close Update Order Status panel	✓	

Remarks / Feedback

None.

Signature:

WoonCC

Date: 25-11-2022

Figure 1.6 User Acceptance Testing (UAT) Form for Waiter Role

1.5 User Acceptance Testing (UAT) Form for Cashier Role

Figure 1.5 shows the test case that will be performed by Ronald Lim Sheng Wei as a cashier of the restaurant.

User Acceptance Test Form for OSMR System (Cashier)

Name: Ronald Lim Sheng Wei Position/Occupation: Cashier Date: 26-11-2022

No.	Event	Pre-conditions	Test Data/Steps	Expected Result	Pass/Fail (✓ - Pass, X – Fail)	Comments
Login Page						
1	Login an existing account	Account is verified via email	1. Fill in login credentials 2. Press "Login" button	Redirected to Delivery List Page	✓	
Payment List Page						
1	Access Navigation Menu		1. Press "Menu" icon button	Display Cashier Navigation Menu	✓	
2	Update order status to "Completed"	Current order status is "Delivered"	1. Press "Update Status" button	Display Update Order Status panel	✓	
Waiter Navigation Menu						
1	Access Payment List page		1. Press "Payment List" button	Redirected to Payment List page	✓	
2	Logout		1. Press "Logout" button	Redirected to Login page	✓	
Update Order Status Panel						
1	Update order status to "Completed"	Current order status is "Delivered"	1. Press "Confirm" button	1. Order status updated to "Completed" 2. Order removed from Payment List page	✓	
2	Abort update order status action		1. Press "No" button	Close Update Order Status panel	✓	

Remarks / Feedback

N/A

Signature:

Date: 26-11-2022

Figure 1.7 User Acceptance Testing (UAT) Form for Cashier

APPENDIX E
USABILITY TESTING FORM

For Appendices Heading, use TITLE AT ROMAN PAGES style.

OMSR - Usability Test Evaluation Form

Hello and good day. I'm Tan Chee Kin, an undergraduate student from Faculty of Computing (FKOM), University Malaysia Pahang (UMP). I'm concerned about your views on using Order Management System for Restaurant (OMSR).

Thank you so much for taking the time to respond to my questions. I'm grateful.

Sign in to Google to save your progress. Learn more

*Required

1. What is your user type? *

- ◯ Customer
- ◯ Business Owner
- ◯ Kitchen Staff
- ◯ Waiter
- ◯ Cashier

2. Does the components of the interfaces are well organized? (eg. button, text, icon, images) *

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

3. Does the interfaces is ease of use and consistent? *

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

4. Do you think the words (eg. label of buttons) and icons used in the application is understandable? *

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

5. Is it difficult to learn how to use the application? *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

6. Is it easy to find the desired information in the application? *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

7. Overall, how satisfied are you in the system? *

	1	2	3	4	5	
Not Satisfied At All	○	○	○	○	○	Very Satisfied

8. Is there any comments that you would like to inform the developer?

Your answer

Submit

Clear form