Digital Menu Application For Accessibility In Restaurants

Final Year Project Report

DT282

BSc in Computer Science International

Angela Youyi Peng

Supervisor: Dr. Oisin Creaner

School of Computing

Dublin Institute of Technology

12th April 2019

Abstract

This project is for the creation of an accessible restaurant ordering application for mobile devices. The method of communication will be in two parts one for the user and the other for restaurant owners, both will be built using android studio SDK.

This application has the ability to integrate technology with dining, allowing restaurant owners to manage their business in a modern and flexible manner. When correctly implemented this technology has the ability to not only increase customer satisfaction but also increase restaurant revenue.

Not only does this project focus on the use of QR codes but also allow users to integrate different languages whether for educational purposes or translation purposes. With this accessible feature tourists or language enthusiast can optimize this functionality when ordering food.

The goal of this application is to create an accessible food ordering system that enhances performance and efficiency. By researching the role of technology in the food industry, this application may allow users to take control and create a comfortable dining experience.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Angela Youyi Peng

12th April 2019

Acknowledgements

Firstly, I would like to express my deep gratitude to my supervisor Dr. Oisin Creaner for his advice and support throughout the progression of this project. My grateful thanks also extends to your weekly feedback, without this the outcome of this project would not have been possible.

I am particularly grateful for all the translation assistance given by my international friends offering their time and feedback.

Finally, I would also like to thank my friends and family for their support and initiative for testing this application.

Table of Contents

Table of Figures

List of Tables

List of Code Snippets

1. Introduction

1.1 Project Background

The aim of this project is to assist individuals and businesses with flexible communication in the workplace. Ordering food through an application can be of great efficiency for both users and businesses with the aid of technology [1]. This application can assist users that may have anxiety in social environments to feel comfortable and enjoy their time at a restaurant without the need to interact directly with a waiter.

The real-time notification interaction between users and business provides a firm bridge of communication. Kitchen staff that can see orders placed are able to determine when the order is available to be served, this can be of great advantage to both the customer and the staff. The data from orders allows customers to keep on track of time and allows restaurant owners to manage their business.

This project will consider many ways of user accessibility in design, implementation and development. A key feature to this project for efficiency is the use of QR codes. QR codes, also known as Quick Response codes have existed for over 25 years [2]. It was used to enhance decoding speed and the ability to hold large amount of information. QR codes not only have the function of holding data but it is also a great marketing tool to draw in tech savvy audiences.

Internationalisation is a problem we face on a daily basis, with the aid of technology we are able to break that language barrier. By implementing a language setting into this application, we are giving a chance to open up the many possibilities of internationalisation. Allowing multiple languages can be of great benefit to broaden the business into the global market.

1.2 Project Description

This project will be divided into two parts, client and server with an additional administration role. Client and Server application that communicates together to perform client-server interaction. The client application is used by a customer, the server application is used by a restaurant staff member and the administrator

provides system maintenance.

The client application uses many API libraries to perform tasks such as reading QR code, make order, make payments through the application and receive live updates on order status. Users can make payments through the app after an order is placed. Payments can be received by PayPal, credit card or cash transactions. Once the order is placed the details of the order are sent to the kitchen (Server Application), here the restaurant staff are able to control the server side functionalities.

The server application will be able to manage order status and restaurant menus. Restaurant staff are able to receive a notification, notifying them of new orders, payment methods and extra details the customers may have noted in their order. Restaurant managers are able to add new dishes, edit current dishes and view orders.

The role of an administrator is to be able to oversee all functionalities from the front and back end. They can provide new updated versions, fix bugs and set up new restaurants and staff.

1.3 Project Challenges

Through this report a detailed illustration of each process and technologies used are studies. During the progression of developing this android application there were many challenges faced. While many of these challenges occurred during the progression of the development of this project, they were all overcame with determination and further research. Below are some of the major challenges and issues that were encountered.

1.3.1 Appropriate API Libraries

Choosing a precise API for certain functionality was a challenge as there are both pros and cons that needs to be considered. While choosing a third-party barcode scanner there where many libraries to choose from. Both Google Mobile Vision and ZXing barcode scanner were researched. At the end, ZXing API [3] was chosen because of its simplicity in use compared with google vision.

Retro Fit Representational state transfer (REST) API [4] was a challenge to implement as data serialisation is performed by using a GSON converter which is a java serialisation library. This took some time to understand, perfect its usage and syntax. In section 2.4.3.1, a detailed explanation of its use and operations in this project are illustrated.

1.3.2 Back-End Database

Android applications allow the use of many different backend services including built-in data storage and third party services. Choosing a well performed backend service that fits this project's intentions slowed down a significant amount of time in the development phase.

Although the main data storage is stored in Firebase [5], SQLite Database Browser [6] is also implemented in this system. Both databases have its own individual job and role in this application.

But with determination of producing real-time data and notification services, Firebase was the best service to implement. In section 2.4.3.1, further research on the performance of firebase is detailed with the addition of other back end services.

1.3.3 Accessible User Interface Design

To implement a user-friendly interface that is accessible to users with colour blindness, choosing a colour palette was a great difficulty. Considering the different types of colour blindness and its statistics, deuteranomaly is one of the most common kind which is a reduced sensitivity to green light. Research shows that one in twelve men [7] are colour blind. After testing out different colour combinations with a colour blind tool a simple solution is to set the contrast of colours and text to a 4.5:1 ratio [8].

Another accessibility that is implemented is the usage of a multilingual application. Localising application texts and data was a challenge that was faced. Since the menu items are stored in firebase, a second set of data needed to be added for a second language to be integrated into an application. This took an extensively long time as each data is added manually to enhance the accuracy of human translated

texts. Similar for any string texts that are shown throughout the application, strings of each language is manually added.

1.4 Project Objectives

The main objective of this project is to create an interactive order management system with the aid of QR codes. By using QR codes we are able to retrieve data efficiently from firebase backend. The objectives of the final product will be two application systems with the first one being for the restaurant manager and staff while the second primarily focuses on the customer. Additionally the administrator has the role to control the performance of the overall system which includes system updates and maintenance. Below are the details of the objectives that both applications require to produce an interactive ordering system.

Businesses should be able to perform basic management functions in this application. Management functions such as adding new categories, food items and viewing incoming orders etc. In order for a productive and successful business to operative the following objectives are needed:

- Developing an interactive system between the client and the server to provide customers with live notifications of their order status.

- Developing a fast and efficient menu management system which updates a specific restaurant's menu in real-time.

Customers should be able to view menu details, place order and view order status. Like any food ordering applications the basic functions apply. Payment transactions and table management details are some accessibility features that are implemented to this application. To accommodate user accessibility features for a smoother dining experience the following objectives are required:

- Developing an application that has the ability to use the device camera to capture and decode a QR code.

- Developing a graphical user interface that fits a certain contrast ratio to assist users with visual impairments.

- Integrating a descriptive language to buttons, text view and image view to aid talkback and braille back accessibility features.

1.5 Structure of The Document

- Section 2 – Research

In this section, research in background, similar applications and technologies are conducted.

- Section 3 – Design

In chapter 3, the design implementation is discussed and how complementing colours play a major visual impact for colour blind users.

- Section 4 – Architecture & Development

In this section, the architecture and implementation is discussed backed up with sample codes.

- Section 5 – System Validation

Testing will be conducted by users of little technical knowledge and tech savvy users.

- Section 6 – Project Plan

This section discusses the projects life cycle and any changes that were made during the course of this piece of application.

- Section 7 – Conclusion

To conclude, a reflection of the overall project, future plans and learning curve are discussed.

2. Research

2.1 Background Research

In order to create a real-time client/server food ordering application, a deep understanding of the food and business industry is reviewed. The role of technology [9] has influenced the food industry drastically in recent years and it has become a part of most modern restaurants.

For a business, it is important to understand its market and acknowledge the connection with its consumers. Flexibility is the key to a successful business, by implementing technology into the food and drink industry we are able to establish this connection. An application that captures user detail and functions subjected to flexibility can build a trusted foundation between the producer and consumer.

Technology can be of benefit to both potential customer and restaurant operators. Customers benefits include improvements in convenience and control, while restaurants can take advantage of improvements of service speed, reduction of labour costs, increase in revenue and enhancement in overall product quality.

Making the customer's experience more convenient relates to the customers wish to sustain their time and effort. With an increase of accessibility, substantially it increases satisfaction. Technology can aid this convenience [10] by providing functionalities to reduce wait time for customers when placing an order, transaction convenience can reduce customers wait time post dining and management convenience to increase the pace of an overall dining experience. When users have a certain control over a service, they are most likely to be satisfied with the final product.

Restaurants can benefit with the use of technology to increase the speed of service which in return more customers can be served [11]. Table management, communication management and transaction management are some services that can increase the velocity of service which can almost guarantee a rise in customer satisfactory.

Technology in restaurants can also improve the business revenue. Adding accessibility in restaurants for example, a self-ordering system will attract more customers resulting in an increase in revenue.

2.2 Dining Experience

Research shows how long customer's dining experience should last. According to Noone, Kimes and Wirtz [12] study, customers reactions change based on the different stages from when a customer initially walks into a restaurant and the type of restaurant. The cycle of dining experience starts from when a customer walks into the restaurant.

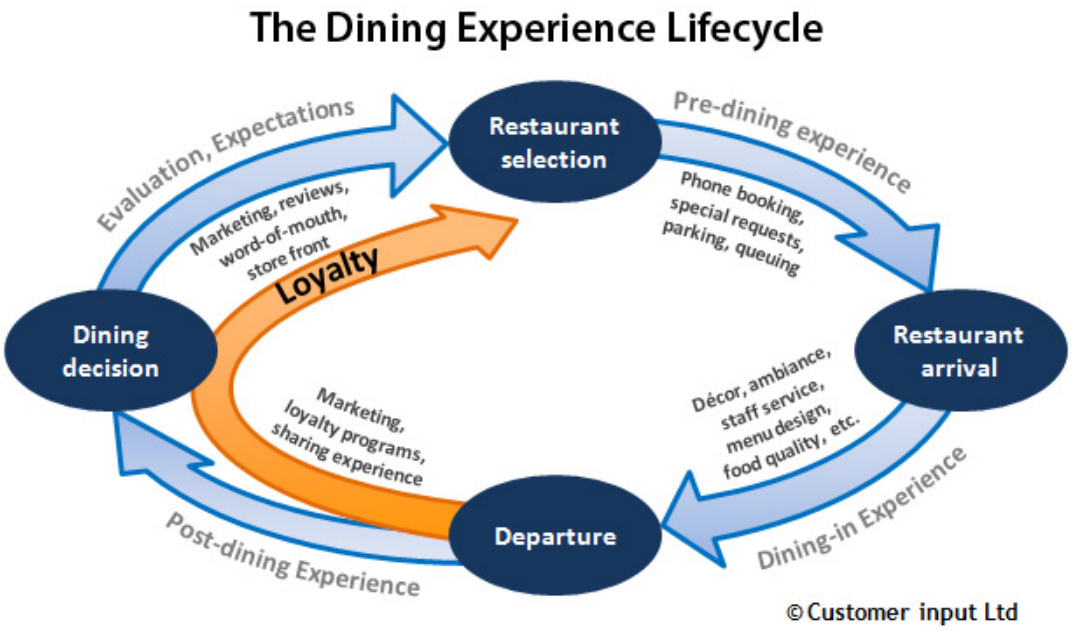

Figure 1 The Dining Experience Lifecycle [13]

In casual environments customers prefer a fast pre-dining and post dining stage yet during dining-in a slower pace is more desirable. Whereas customers at luxury restaurants are more likely to enjoy a slower paced dining experience.

Based on Noone, Kimes and Wirtz [12] study, restaurant managers should focus on these 5 methods to enhance revenue management:

1. Reduction of time during pre-dining phase.

This phase is integrated by the self-service QR code scanner which the user can

access through their own mobile device to retrieve a specific restaurant.

2. Reduction of time during post dining phase.

As the user has pre-paid for their order through this application before their order was sent to the kitchen, customers won't have to waste their time waiting for a waiter to pay for their meal post dining. This stage considers reducing customers wait time post dining which in result shows an efficient impact that this application will provide.

3. Avoid reduction of time during dining-in phase.

During the in-dining phase, customers are able to manage their own time based on their order through the application. As the server application will send status updates to the users at a moderate pace, it allows customers to feel satisfied and comfortable in the environment.

4. Give customers the control of their dining experience

This phase gives customers the ability to customize their own dining experience. As this application allows customers to control their own dining pace it creates a trusted platform for users which in return can increase the level of returning customers.

5. Focus on the importance of a consistent cycle throughout the whole lifecycle.

This application shows consistency and can be used in many restaurants by many customers. A constant cycle can provide loyal customers a feeling of familiarity and can by combining technology into a regular dining experience can attract new customers.

By acknowledging this cycle businesses can build a firm foundation to perform a smooth and satisfied experience for their customers.

2.3 Similar Technologies

This section consists of similar technologies that exists and are in use in the market today. There are a few applications that have similar functionalities as this project that have many advantages. These technologies are studied and researched as

illustrated below.

2.3.1 Order Wizard

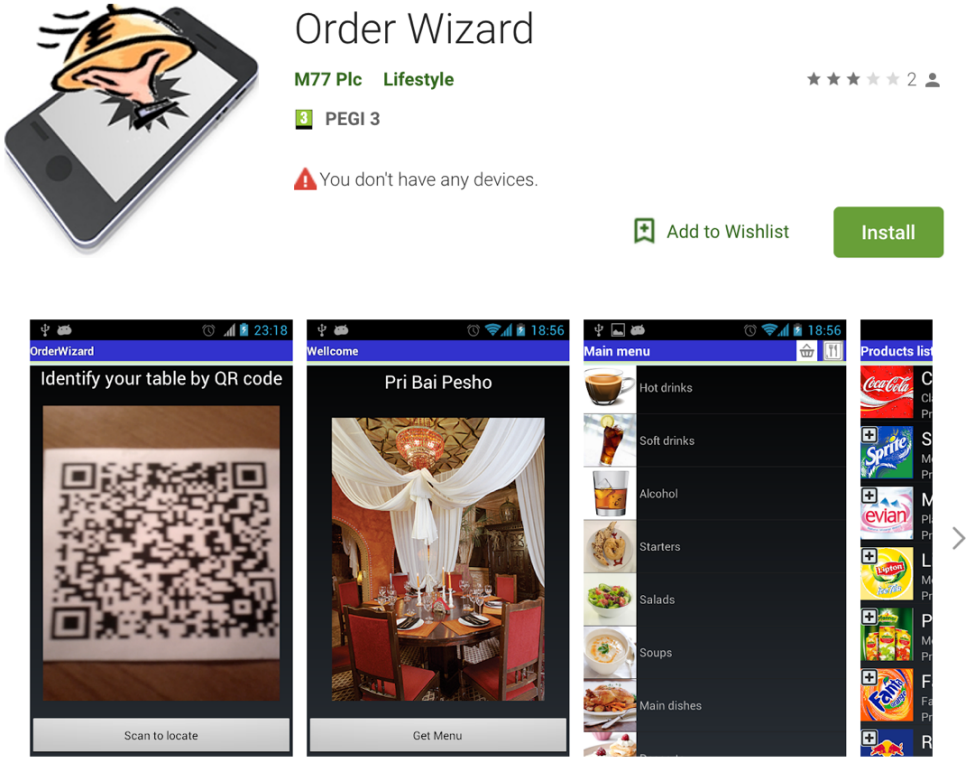

Figure 2 Order Wizard [14]

Order Wizard [14] has similar functions to this project including QR code scanning and processing payments through the application. This application supports both android and IOS systems. The system will ask the user to scan a QR code to retrieve a restaurant, then depending on the restaurant a menu will display. From the menu, the user is able to add items to order, when they are finished they can then edit, delete or add extra details. Payments are completed by PayPal, offline cash or offline card transaction.

A benefit that this application has is that the users can call the waiter if they want to process the transaction with cash.

A disadvantage is that any extra details that the user adds is not by item but by the overall order. This application also does not ask for customer details this may cost a loss in clientele and security breach. User authentication provides secure transactions and personal information without this it becomes a vulnerability. The

system will become exposed to cyber criminals from gaining personal information.

This application is available on google play but there hasn't been any updates since 2013 when the application was first created.

2.3.2 Quick Orders

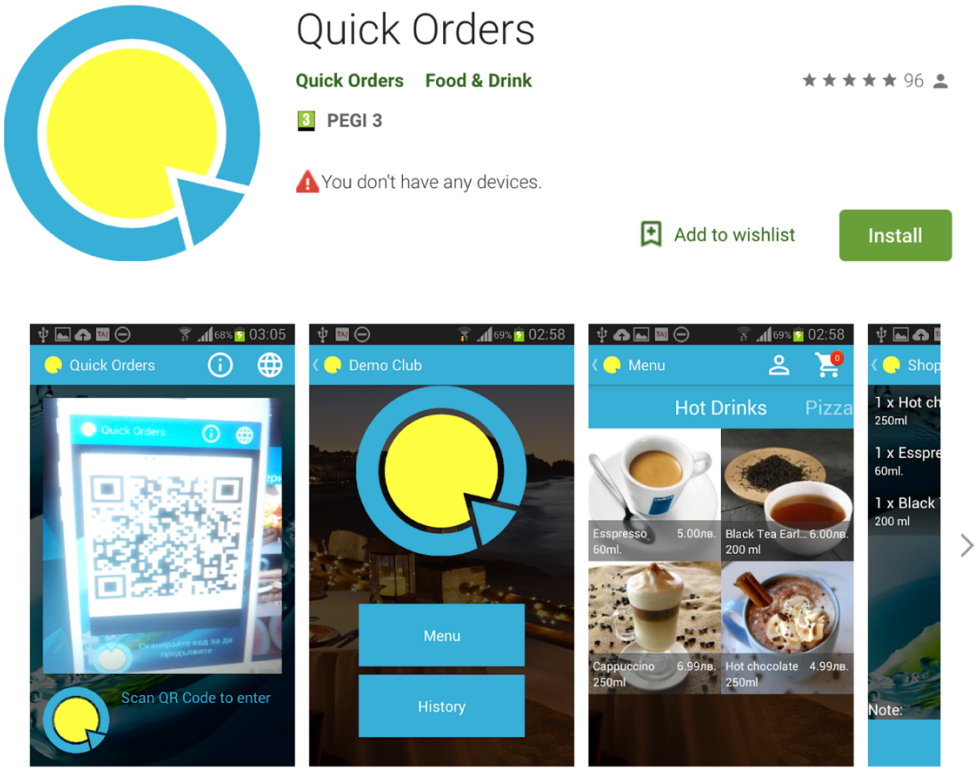

Figure 3 Quick Orders [15]

Quick Orders [15] is available in the google play store and it has the functionality of a barcode scanner that supports all mobile operating systems. Similar to this project it has the ability to order through the application, add personal comments and see order histories. A key feature to this application is the multilingual functionality which is a great accessibility characteristic. With this feature, this application can be opened up to a larger global market.

A major advantage of this application is that it is multilingual. Being able to change and set the language according to the user. This feature provides access to users internationally which can help enterprises and businesses to expand their market

globally.

A downside to this application is that there is no payment transaction through the app. Once the user places an order it doesn't ask for confirmation which means any individual that has access to the QR code can make an order and not pay for it, this is a major flaw in the application. If the developer implemented a payment process it can allow quick transaction process and increase user authentication.

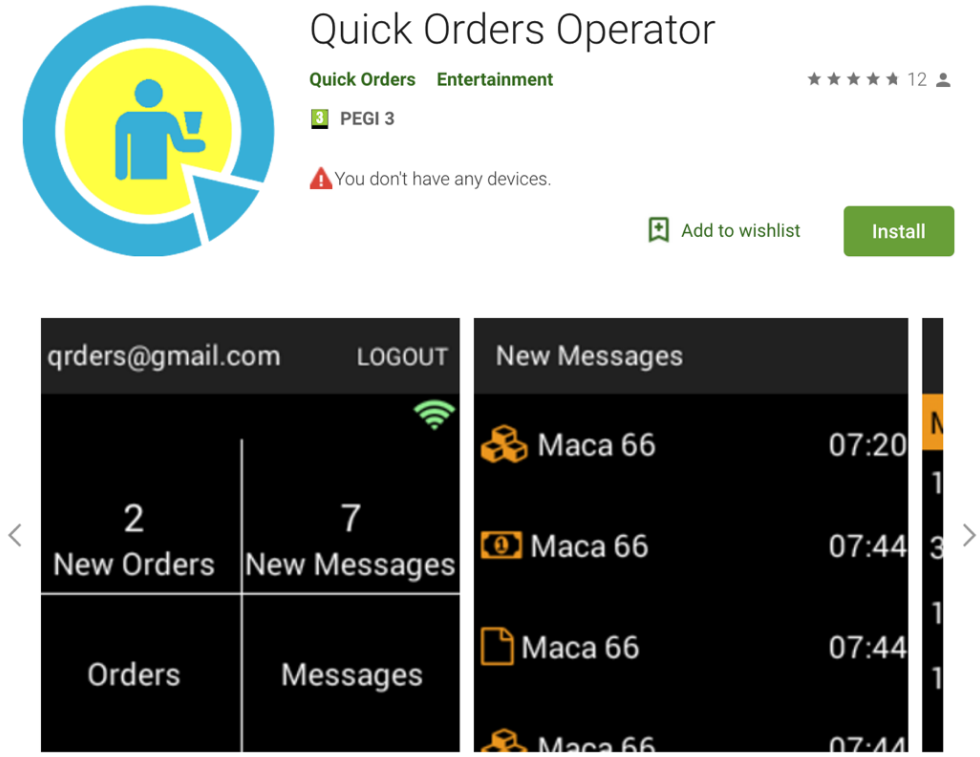

Figure 4 Quick Orders Operator [15]

The image above shows merchant side application which complements the customer app. This Quick Order Operator [15] shows restaurants or bars of incoming orders. It provides information about each order, item ordered, quantity and total amount.

A disadvantage to this application is that there are no functionalities for business management which is not as accessible for merchants. Through this application restaurant managers are not able to add, update or control dishes to their menu.

2.3.3 eHopper POS

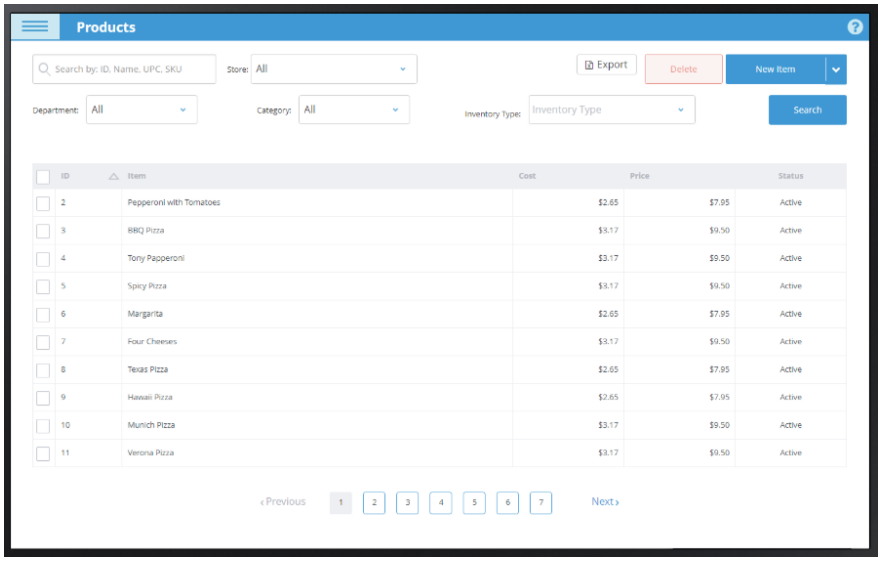

Figure 5 eHopper Application [16]

eHopper [16] is a point of sale management system. This program is available in IOS, android and windows operating systems. According to their webpage and documentation this system possesses many operations from Multi-store management, POS reports, employee management, order tracking to tax management and ingredient management.

An advantage of this POS is it provides the business with an extensive amount of functions to manage their performance. The system supports many devices and it provides a self-explanatory user interface.

A disadvantage of eHopper is that for a business to access its full potential, expenditure of the system increases.

2.4 Technologies Researched

During the progression of this project, there were many new technologies to be studied. In this part, different technologies used were and a thorough research was conducted in aid of developing a digital menu application system

2.4.1 Application Platforms (Android Vs. IOS)

Both platforms are supported on many devices and SDKs. While Android and IOS are at a constant competition [17] with one another, there are pros and cons that

need to be considered when developing an accessible application. Below discusses the differences between Android and IOS in modern applications.

2.4.1.1 Android Platform

The android platform is set out for global usage eminently for areas of lower income. About 75% [18] of the population worldwide use an android platform device, as android is available to many different manufacturers even with different SDK versions consumers can experience various hardware devices but still remain as a loyal customer to android. This is also known as Device Fragmentation [19]. While developing applications developers must consider limited devices that the application can support otherwise it will cause complexity.

Advantages of Android platform [20] includes its availability and flexibility to support various devices. Androids can be developed on different machines including Macintosh operating system using a popular development language such as JAVA.

Disadvantages of Android platform development [20] is that it takes a substantial amount of time to develop which can cost a lot of money. To support many android devices, the high level of device fragmentation usage can be overwhelming.

2.4.1.2 IOS Platform

Benefits of developing an IOS application [21] includes is the limited use of device fragmentation and less configuration is needed. Development costs sufficiently less than developing an Android application.

Down side of IOS development consists of its primary programming language of SWIFT [22]. As SWIFT is still a language that is fairly new in the technology world, it has its limitations. Changes and updates of new versions of SWIFT is uncertain and can cause a drastic change in overall functionality of applications and compatibility issues with Objective-C.

Having analysed the difference between both Android (java) and IOS (swift), a decision was made to implement this project using Android operating system. Undeniably, android is still evidently more flexible in use and allows the support of many APIs to enhance applications performance.

2.4.2 Front End Development (Native Vs. Web Vs. Hybrid)

Developing a graphical front-end user interface native, web and hybrid development [23] were considered. They each contain their own pros and cons list of usability features. The three types of front end development that were researched are discussed in detail below.

2.4.2.1 Native Application

As this project is concerning the development of an android application, android API native contains many tools that can be directly accessed through the IDE [24]. Android studio depends on XML files for activity views and layouts. As the application is native to its platform XML files can be accessed in a fast and responsive manner.

Specific UI design which can be developed straight from the IDE can provide an overall better user experience. With additional device built-in features a more flexible interactive experience can be accomplished, giving the final product a much smoother transition and user input/output transactions.

Native application [25] can also benefit users with mobile devices as it is developed and optimized for a specific device. Developing an application that can be used on a daily basis can increase revenue for businesses and a mobile phone application can provide that portability and efficiency in use.

Disadvantage of native applications [25] includes the need of an extensive amount of knowledge in the programming language during development and maintenance phase. Native applications need constant software updates to fix any bugs that may occur or arise. Vast amount of time spent to implement the large amount of work that is expected to develop a flawless native application can in return sequentially cost a fortune to maintain.

2.4.2.2 Web Application

Web application [26] hosted by web browsers are condensed information from the website to produce a more functional application that acts similarly to a native application. A wide range of technologies can be used to implement a web

application this includes, open source platforms like angular.js and Node.js.

Advantages in developing a web application is its trouble-free methods to maintain and fix bugs without the user having to constantly update the application for new versions. Web applications can be accessible by all platforms through an internet connection which allows cross-platform application service. Cross-platform application service can benefit both the user and the developer. Users have that flexibility to view the application on any device, developers can primarily focus on perfecting the web application functionalities and usability.

Security is a priority in web applications because unlike other simple websites, web applications are utilized on the machines dedicated server. Internet access is needed at all times, without it users cannot experience the application to its full potential.

2.4.2.3 Hybrid Application

A hybrid application [27] is a combination of a native app and a web app. This is built by using a combination of JavaScript, HTML, CSS wrapper in a native container like, web view. The hybrid method is simple to develop and development time is significantly faster than native apps.

The key benefit of utilizing a hybrid application is that it allows access to different device platforms to be integrated which is therefore cost efficient. Constant maintenance can be done without having the user update the application unless there are any major changes.

After comparing these three types of front end development, it came to the conclusion that this application will be developed as a native application. Users are not able to experience a native application's full potential if the application was developed as a web or hybrid application.

2.4.3 Back End Research (Firebase Vs. SQLite)

There are many back end servers that are available to use for developing an android application. The two backend data storages that are studied for this project consists of firebase which can provide real-time data and android studio built in SQLite

database. Below discusses the these databases usages and limitations in detail.

2.4.3.1 Firebase

Firebase [5] is a free cloud hosted service which can be used for application development as a backend server. Firebase offers many abilities to perform an interactive client-server operation.

Firebase database offers storing and synchronising real-time data to any connected devices of all platform. With this real-time data functionality notifications can be sent to targeted users using Firebase Cloud Messaging (FCM) [28] . By allowing REST API to make a call request to the firebase server a message is then sent or received.

Because firebase follows the concept of JSON structured data, this makes implementation simpler so that the back end development is not as complex as relational databases. Although the implementation cycle is simple, the design of the structure is far more complicated.

By using firebase authentication [29] with phone/email and password, Facebook or google account that can produce a more secure application to identify a real user. It also allows the use of google analytics to provide a statistical view of the efficiency of the application.

2.4.3.2 SQLite

SQLite [6] is an open source database service and it can be accessed using android studio IDE. SQLite requires no installation for this service because it is an in-process relational database management system (DBMS). Its lightweight structured file that uses relational query language enables SQL to retrieve data.

This database does not rely on a server which means no configuration is required and internet usage is needed. Communications are directly with the application for this instance Android studio IDE. SQLite can only handle low to medium traffic which is not suitable for large data storing application.

2.4.4 Application Programming Interface and Libraries

With the introduction of google play service 7.8 [30], Mobile vision API released a

new API which enables barcode scanner to read and decode its contents. Google barcode API is able to detect barcodes in real time on devices in any orientation. By implementing this with permission to use Camera API, an android framework which supports various camera features. This allows the application to access the device camera and scanning any barcode it detects.

ZXing [3] also known as “Zebra Crossing” is a google operated open source service which is used to read and decode barcodes using an image processing library. This library allows the operation of a barcode scanner, it has the ability to decode two dimensional black and white grid then output the encrypted message. ZXing API can capture images from a distance and at different angles. Compared to Google Mobile Vison API, the ZXing API is more capable and reliable to capture QR codes.

A migration from android to android jetpack [31] also known as AndroidX was performed to improve android studios library performances. AndroidX is a more improved set to utilize libraries, tools and frameworks to perform uniquely according to its developer.

By performing this migration on the client application during development stage, many support libraries and packages can be maintained separately from the main project files.

Figure 6 Android Jetpack [31]

2.5 Other Relevant Research

Other relevant research in relation to customer relationship management, customer controlled point of sale and vision impairments were studied in this section. To fully understand users' needs, the following areas of relevant research are essential.

2.5.1 Quick Response Code (QR codes)

Quick Response codes [2] also known as QR codes are becoming extremely popular in the technology industry. QR codes is defined by the oxford dictionary as a machine-readable code consisting of an array of black and white squares, typically used to store URLs or other information for reading by the camera of a smartphone.

Quick Response codes were first introduced by Denso Wave during 1994. It was used to enhance decoding speed and the ability to hold large amount of information. These codes are growing rapidly in the marketing and ecommerce community. According to an article by OpenJaw [32] the rise of QR code usage is targeted in Asian countries especially China. Unlike any other country China has 900 million active consumers a month using a popular worldwide application called WeChat [33]. This application contains many functions one in particular is its use of QR codes for payment transactions, marketing and user identification. A downside to this application would be its functionality outside of China. This app will only operate at its full potential if the user has a Chinese credit card and a Chinese phone number otherwise it is just another social media application.

By integrating quick response codes in to this application can provide users a sense of control and modernity. This can lead to the expansion of the use of technology in restaurants, making users dining experience more convenient.

2.5.2 Customer Relationship Management (CRM)

Mobile customer relationship management (CRM) [34] is extremely important in today's food industry. With the increase in the use of technology and internet services businesses ensure customers are receiving the best service at all times. Capturing client details allows employees to provide real-time data and information to clients immediately.

CRMs acts as a communication connection between restaurant owner and their

customers. By asking customers for their details, businesses can send user sale offers and deals of an item to capture returning customers or new customers.

Benefits of customer relationship management [35] is its ability to deliver performance reports, analytics in sales and organizational activities in businesses.

2.5.3 Customer Controlled Point of Sale

Point of Sale (POS) systems contains tools that are essential to businesses that handles sales reporting, customer management, Inventory management and employee management. POS can provide the store manager with information that increase business efficiency.

Implementing this into a mobile application, results in a customer controlled point of sale system [36]. This provides users a sense of authority when making an order. The user can manage their own order and process payments directly through the application by card transactions.

2.5.4 Localizing User Interface

Advancement of technology made it possible to communicate with individuals of different nationalities to interact with each other at ease. By localising an e-commerce application, it is reaching out and beyond to a greater community of potential users [37].

For instance, if a user that has many allergies is travelling and ordering food in a foreign restaurant becomes a difficulty an application that supports different languages can benefit these individuals.

Android studio is able to localize the user interface to support different languages. By utilising translation editor in the IDE, it will allow the application to change from one language to another and not affecting the system's default language.

To implement translation properties, machine translation and human translation [38] mechanisms are studied. Machine translation provides many benefits, for one, it is less expensive than to hire a professional translator. Machine translation tools are becoming more and more reliable as data is fed into these software's constantly but

these improvements is still no match for a human translator. Human translators are relatively more reliable to translate texts into human readable terminology although it may cost dearly.

2.5.5 PayPal

PayPal sandbox provides developers a business account and a personal account both includes virtual money to test payment transaction within the application.

The most secure way of electronic transactions is through PayPal [39]. PayPal accepts payments using credit card and PayPal account transactions. By using HTTP based RESTFUL APIs, we can retrieve real-time data to authenticate credentials. Responses and requests are structured in JSON. PayPal Sandbox allows developers to create virtual businesses and personal credit card accounts with virtual money.

2.5.6 Vision Impairment

Developing an application that allows flexibility and change in user interface can ease any conflict and intimidation for individuals with vision impairment. Colour schemes, amount of information displayed and navigation should all be considered when developing.

Many vision impairment accessibility features includes Talk-Back [40] and Braille-Back [41] requires applications to be designed in a simple layout to accommodate these functions. When developing an accessibility application all these services are researched thoroughly.

Talk-Back functionality is a built in accessibility operation in android devices that allows all characteristics of an activity to repeat certain buttons, texts and views to users using the speaker. This functionality allows users with visual impairment to hear where they navigate to throughout the mobile device.

Braille-Back is an independent machine that translates texts to braille which helps users that are blind to read text from this machine to operate a mobile device. This device gives an extra accessibility feature for the use of android devices.

Users with colour blind impairment also face difficulties when using an application with high volume of imagery and texts it can be difficult to differentiate the texts. By using adobe experience designer a prototype was designed and then modified to optimize the user interface. In section 47 , a detailed illustration of the design process is discussed.

3. Design

3.1 Methodology

There are many different types of methodologies that are available to integrate into a successful project and a working product. Methodology itself is defined to be a system of methods used in a particular area of study or activity [42]. Below discuss the methodologies that are researched into this project which include the waterfall, agile and user centred system design (UCSD) methodologies.

3.1.1 Waterfall Methodology

The waterfall model methodology [43] is a simple structured organizational method. The advantage of using this method is its linear structure making this approach understandable and manageable. Applications that follow the steps of the waterfall methodology are more likely to face errors as each phase must be perfected until the next stage can be performed. This linear methodology can cause time loss during the progression of development. Below illustrates a thorough explanation of each step taken to utilize this methodology.

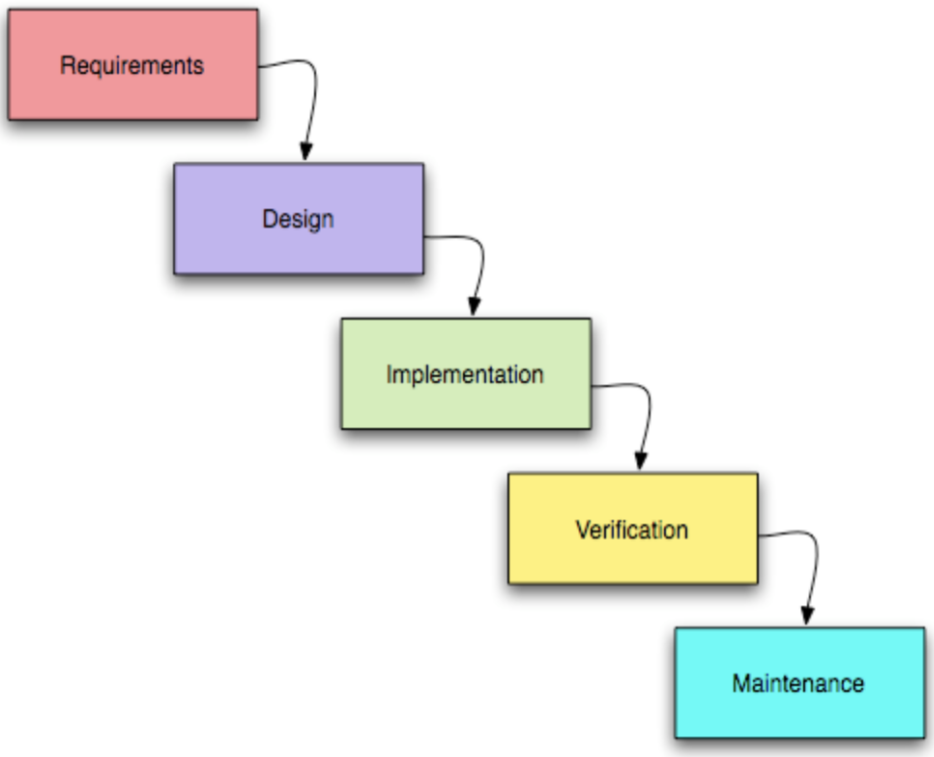

Figure 7 Waterfall Methodology [43]

- Requirements

Requirements phase gathers information of what the system should be able to perform. Information is written down and documented in detail that includes any research conducted. This phase is a crucial part of project planning as it provides the basic foundation of the project, without this phase carefully studied it will cause a lot of complications in future development.

- Design

During the design phase it primarily focuses on technical architecture design, programming language used and diagrams to assist in implementation. A prototype of the user interface is designed to assist the implementation phase.

- Implementation

The coding stage of the project puts the above two phases into action. Creating and implementing the requirements which includes both front-end and back-end development.

- Verification/Testing

Testing phase, uses testing tools to resolve any issues that emerge throughout the application functions. The waterfall methodology shows that testing is performed at the final stage of development which can cause major complications because testing should be conducted throughout the whole development of the project's implementation phase.

- Maintenance

Last phase is maintenance, once the application is ready to be formalized to the market. System refurbishment and debugging are still maintained and kept under control by system updates. Some modifications can be done with internet access and won't cause major effects on the system usability and performance.

3.1.2 Agile Methodology

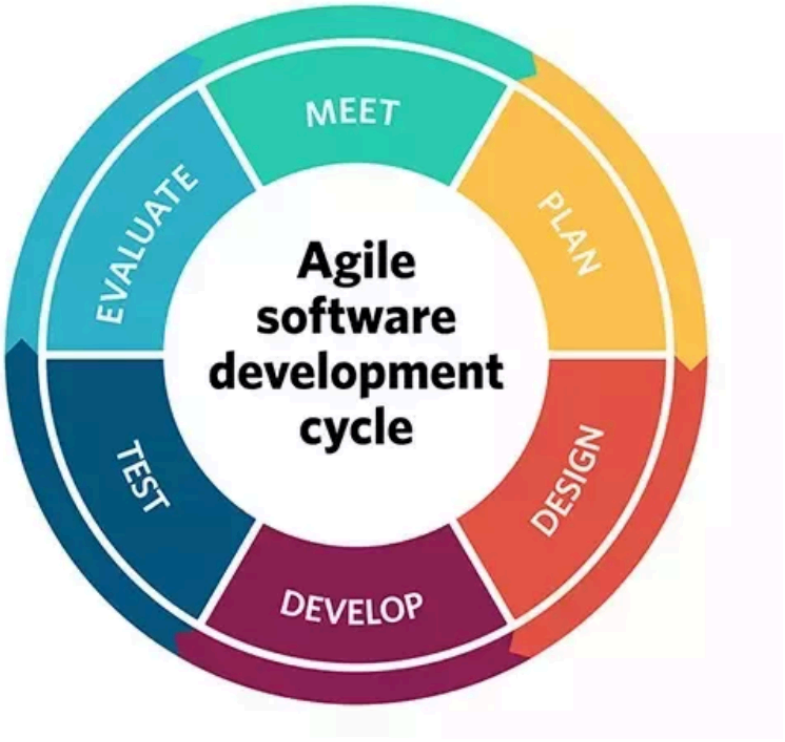

Figure 8 Agile Software Development Cycle [44]

The agile methodology [44] is a method that provides quick and responsive feedback to developers in the direction the project is headed. During the development phase it has many opportunities to assess the project, unlike the waterfall methodology which is tested at the final stage of development. The agile manifesto has four fundamental values that are considered:

"(1) Individuals and interactions over processes and tools

(2) Working software over comprehensive documentation

(3) Customer collaboration over contract negotiation

(4) Responding to change over following a plan"

By implementing some aspects of agile methodology and principles, a final product was developed in a time efficient and organised manner. Agile process is customer focused and aligned with businesses needs which defines the overall aim of this project.

Each step of agile methodology is taken into account and thoroughly performed to produce the final application. To maintain a consistent workflow daily, new targets and goals are set and previous bugs were resolved. Following the cycle of agile methodology, goals were achieved on a daily basis which in result increased productivity performance.

1. Meet

This stage of the agile methodology set out requirements and objectives that the final product possess. Background research are studied including the platform and technologies used, by gathering data before development is time efficient. This phase plays an important role in project management as it provides the groundwork of software development.

2. Plan

The planning stage consists of daily tasks that are set out. Each day a new plan is created in aid of time management when developing this application. For this project planning requires all goals and tasks to be completed in a consistent order to maintain a constant flow of work.

3. Design

This stage includes the design aspect of this project which follows the User-Centred System Design Methodology illustrated in section 3.1.3 below. This phase play an important role in the development of this project as the main design is targeted to users of all ages and users with visual impairments.

4. Develop

As the agile methodology follows a continuous cycle of actions, the development phase is returned and many aspects to this project have been re-developed to fit a more accessible application for users. This phase took a considerable amount of time as it is the most important stage of the overall project. Development stage consists of activities, functions and operations that produced a working application.

5. Test

Tests are performed continuously during the development of this project, if not by a test subject then by the developer itself to ensure that each function implemented is working. The testing phase in agile methodology can be performed regularly in parallel to the development phase, contradicting the waterfall methodology which performs testing at the very end of the project lifecycle.

6. Evaluate

After testing, the evaluation phase evaluates the errors and issues occurred. Each stage is repeated in the agile cycle until the evaluation possess success for all sections of the application. Once all operations are successful during testing, the evaluation phase takes the role of maintenance and system updates.

3.1.3 User-Centred System Design Methodology

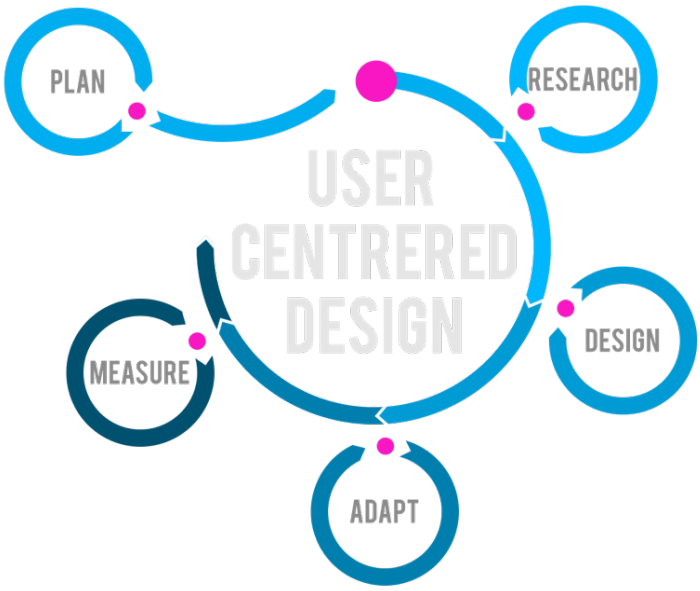

Figure 9 User Centred Design [45]

User-Centred System Design (UCSD) [45] methodologies focal point is usability throughout the entire development cycle and how it impacts users. Usability measures the productivity of the product and its satisfactory level to a targeted

audience. There are ten key principles of UCSD to remain focused on usability:

1. Design for the users and their tasks

This principle takes into account the straightforward navigation system in this project. When designing the layout of this application clear labelled buttons and descriptive string texts where implemented to assist users with visual impairments when using talk back functionality.

2. Maintain consistency

Consistency is maintained throughout the application providing a familiar environment for users. The colour of the application is consistent along with activity layouts. To naturalize the environment, a simple theme is designed to help users feel comfortable while using this application.

3. Usage of simple and natural dialogue

All texts are implemented in a very distinctive and straightforward plain text. Error messages are displayed in simple casual format to assist users when using the application.

4. Reduce unnecessary mental effort by users

Reducing unnecessary activities and functionalities can provide users a more efficient usage and shows the potential of what this application can offer.

5. Provide adequate feedback

Feedbacks can provide users a sense of assurance to trust this application. This application allows users to give feedback of certain menu items which in return give restaurant owners the ability to analyse their businesses performance.

6. Provide adequate navigation mechanism

Navigation is designed to assist users throughout the application in an uncomplicated manner. Consistency is maintained by developing a simple navigation

mechanism.

7. Let the user take charge

By letting the user obtain control of the application, the user can grow confident and reuse this application again. Allowing the user to take charge when using this application can be of great benefit including giving customers their freedom.

8. Present information clearly

All information in this application including order history, status, menu items and restaurants details are clearly displayed for users to access.

9. Offer assistance

Each step of the ordering process is clearly displayed by text hints and labelled to assist users when making an order. This design stage helps users with the aid to operate functions that may be unfamiliar.

10. Error-free

It is guaranteed that there will be many errors during the development of any application system. Nevertheless, attentive implementation during development avoids any careless mistakes that may occur.

While developing this application all of the above principles are involved, especially when designing the user interface. This was a challenge and countless research and tests were conducted. As this application focuses on the accessibility usage for users, many features of visual impairment designs were considered. By following the principles of UCSD a clear visual and navigation mechanism was designed.

In conclusion, the use of both agile methodology and UCSD played a significant role in the development of this project. The agile methodology provides a consistent and repetitive method of development compared to the waterfall methodology. This results in the waterfall method to be inefficient with its linear method of actions. As for UCSD methodology, this method provides design principles that centralises user accessibility characteristics.

3.2 Use Cases and Sequence Diagrams

Use case diagrams and sequence diagrams aided the development of this project by providing a flexible user interface. This section comprises of use case diagrams and their corresponding sequence of actions diagrams. To understand each actors' responsibilities, the following use case diagrams, use case scenarios and sequence diagrams are designed and explained below.

3.2.1 Customer Interaction to Client Application

The first interaction includes customers actions with the client application. Customers are able to perform various operations when using this application. The process of the customer upon registering and making an order will involve at least 2 actors, the customer itself and the restaurant staff.

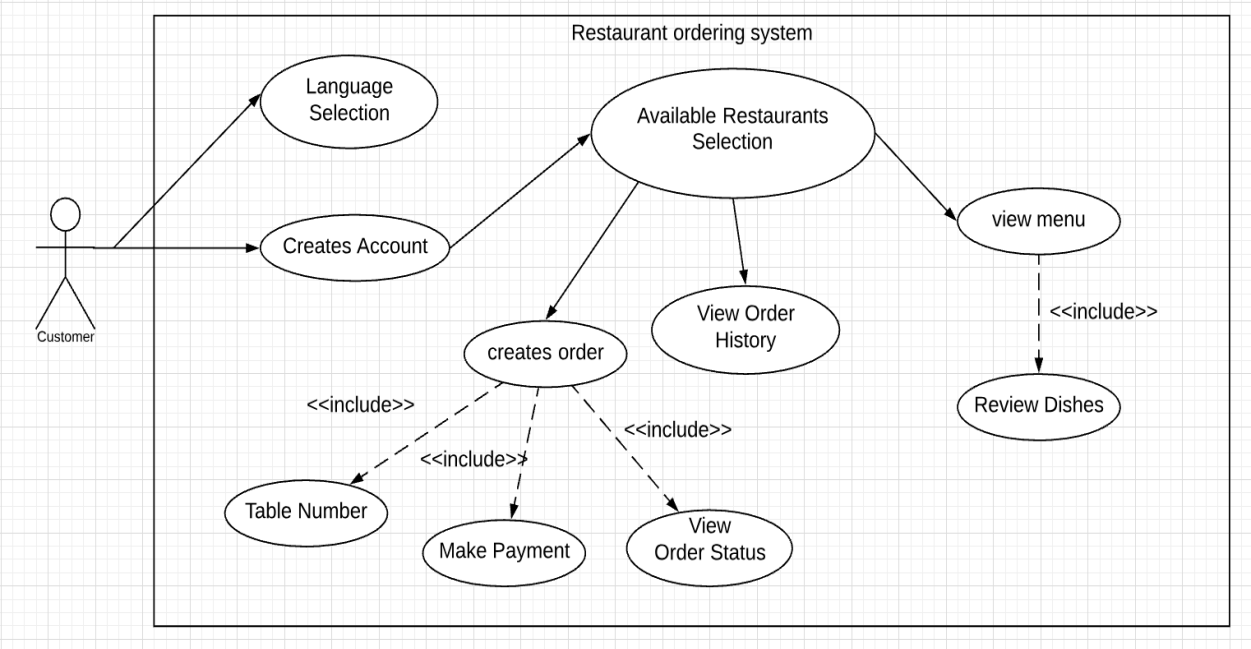

Table 1 Use Case of Customer Interactions

Main Flow of First Iteration	
1.1	Customer creates account
1.2	Customer are able to change the language to their preferred
1.3	Customer are able to scan a QR code
1.4	Customer are able to create order from menu items

1.5	Customer would need to input table number, comments, method of payment
1.6	Customer can make payment through the app by PayPal
1.7	Customer can view order status on their current and previous orders
1.8	Customer can view order history on selected restaurant
1.9	Customer are able to review dishes of menu items

Conditions of First Iteration	
Description	From the first iteration, we can see that the Customer is able to perform many tasks such as create account, make order, view menu and view order status. Customer must select an available restaurant in order perform certain tasks.
Pre-Condition	We know nothing about the user until the user has logged in or signed up.
Post-Condition	We know information about the user, selected language and order details.
Exception Errors	Some menu items cannot be translated.
Alternative variations	Users are validated by unique phone number and a corresponding password. Restaurant menu to view is validated by a QR code containing restaurant ID.

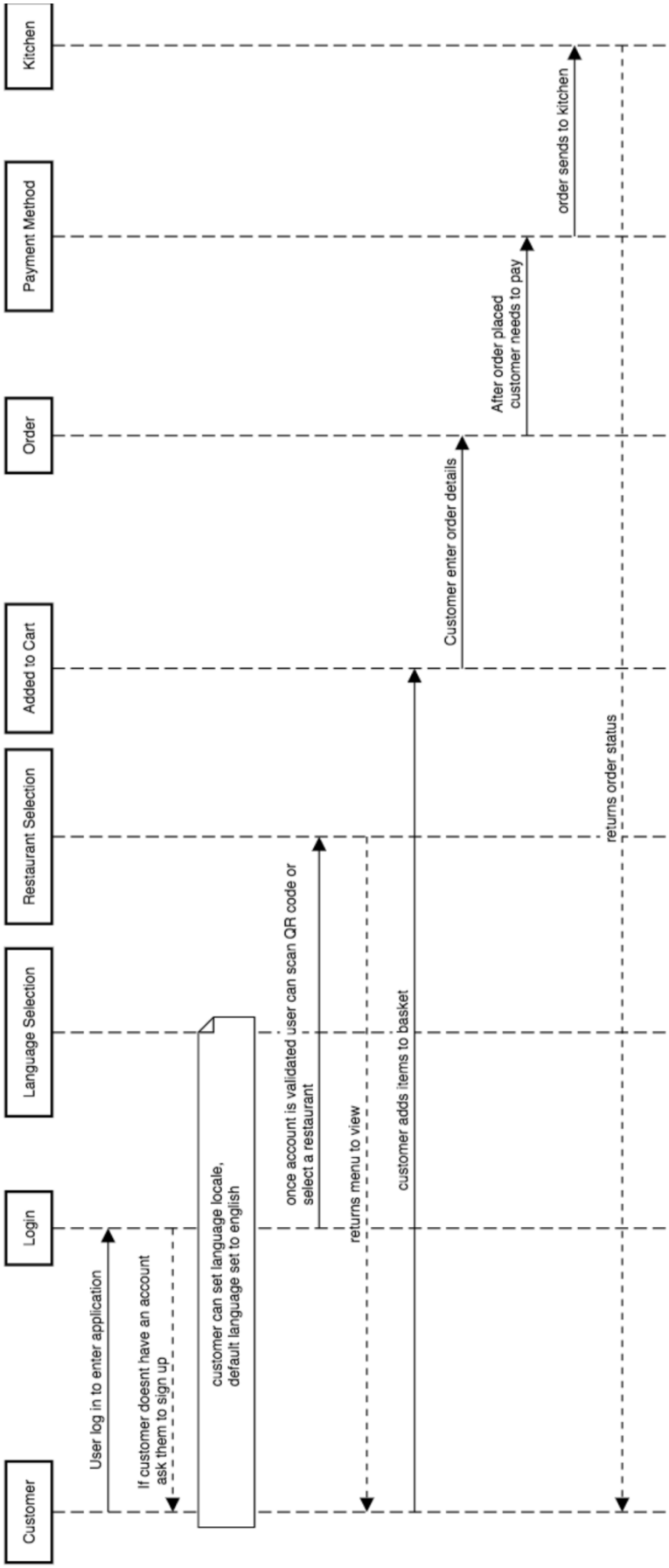

Table 2 Sequence Diagram of Customer and Restaurant Interaction

The sequence diagram shown above shows the progression of actions performed by the customer when using this application. Upon using the application the user is able to set their preferred language or leave the default language to English. Users must log in to an already existing account or create a new account to be able to scan a QR code. Inputs will be validated using firebase email and password authentication.

Once the user is able to access their account they are able to use the application freely. The customer will be directed to the QR code activity, they must first allow the application to access their mobile device camera to perform the barcode scanner capability otherwise users won't be able to view the restaurant's menu. After selecting a restaurant, the menu activity is shown and from here the user can add items into their cart.

Once the user is finished with their order they must specify the table number they are seated at and pay the total amount that is owed before the order goes to the kitchen. Users will be asked to select either cash or PayPal methods of payment. If the user wishes to pay through PayPal they will be redirected to the PayPal payment activity otherwise, a member of staff will be called.

Subsequently, when the transaction process is successful the order will be sent to the kitchen waiting to be confirmed. The user then will be able to track their order based on order status. This status is changed manually by restaurant staff until the food is served.

3.2.2 Customer to Staff Interaction

Second use case diagram demonstrates the actions performed between the restaurant staff and the customer. Staff members have limited access to information about the restaurant itself and customers. Staff role includes payment confirmation, order status change and cook/serve dishes.

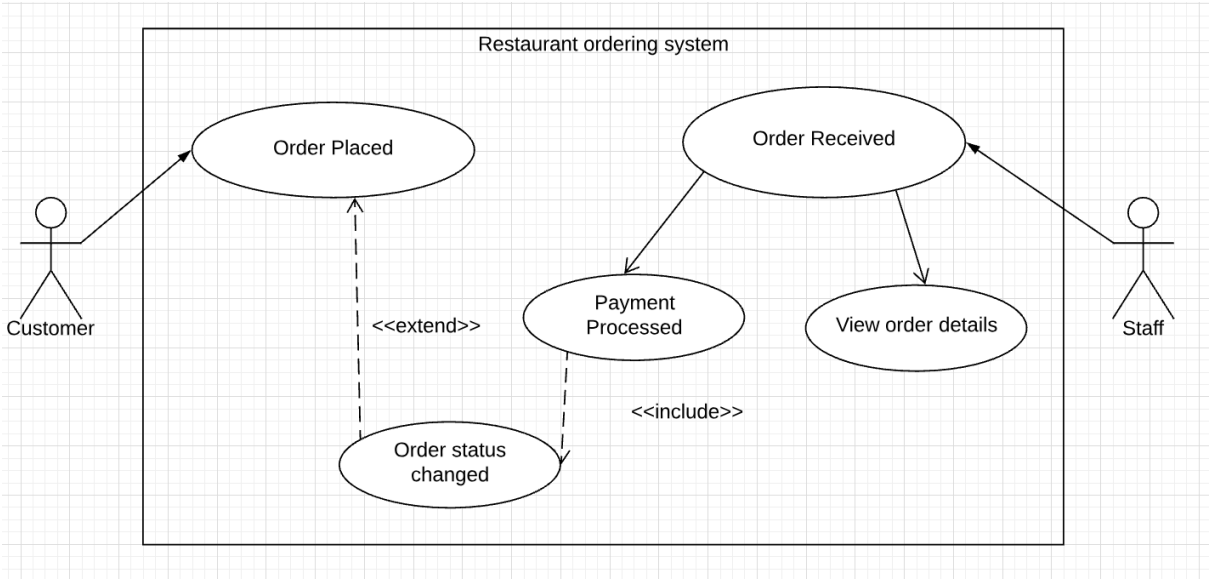

Table 3 Use Case Diagram of Customer to Staff Interaction

Main Flow of Second Iteration	
2.1	Customer places an order that get sent to staff
2.2	Staff receives order detail
2.3	Staff confirms payment process is complete
2.4	Staff are then able to view item quantity and comments from order
2.5	Staff can change order status
2.6	Order status when changed notifies customers

Conditions of Second Iteration	
Description	The second iteration shows the staff and customer. The staff can see the order placed by customers and transaction progress. Staff can change order status.
Pre-Condition	We know information about each staff. Staff does not know any information about customer.
Post-Condition	Staff knows customers order details. Staff knows order status, payment method and payment process.
Exception Errors	The chef can only start cooking if transaction process is completed.

Alternative variations	All staff must log into the staff account to keep track of incoming orders and status.

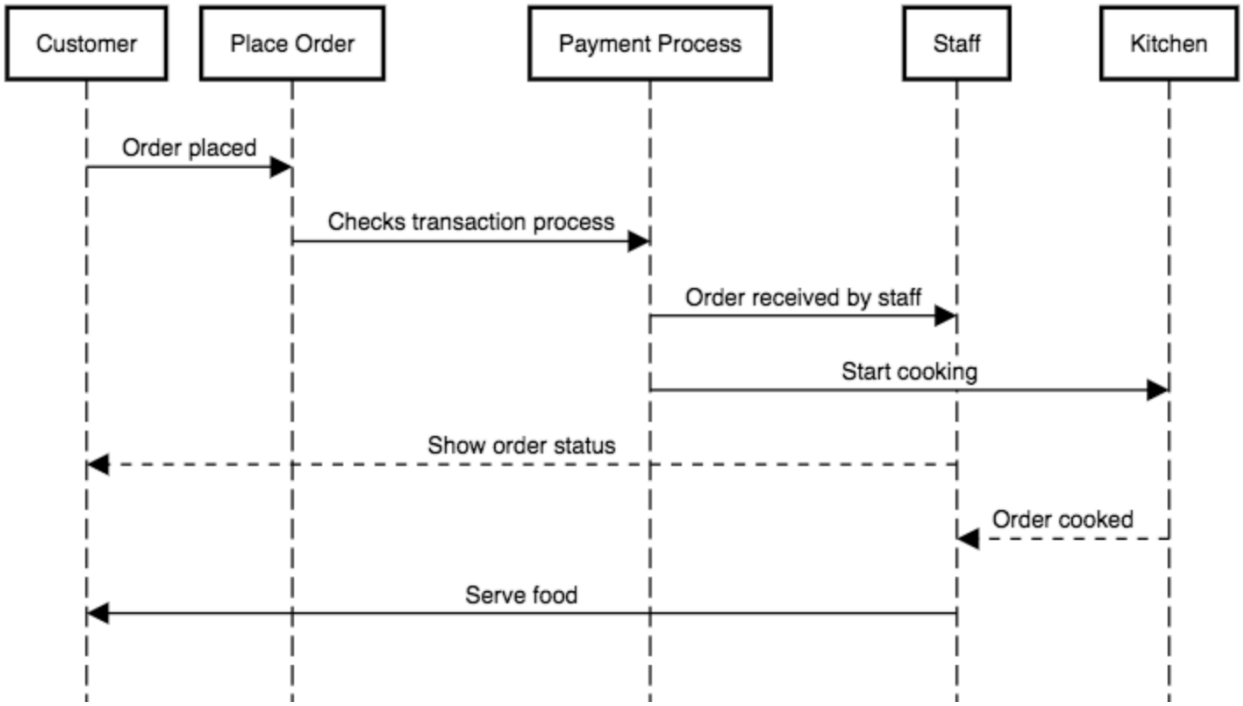

Table 4 Sequence Diagram of Customer to Staff Interaction

Sequence diagram shown above sets the sequence of actions that occurs between the customer and staff. When customers make an order, payment verification is validated to ensure the payment is successful. Once the transaction process is successful, the restaurant staff will start cooking orders placed. The staff will send the order status of each stage of preparation, cooking and delivery process. Alerts sent to customer, notifies them of new order status. Once kitchen staff have finished preparing the order, waiter will then serve the food to the customer.

3.2.3 Staff Interaction to Server Application

The third use case diagram involves restaurant manager only. They have certain roles to control and manage information about their business. They are able to perform create, read, update and delete (CRUD) operations to dish categories and menu items. Managers are able to view dish ratings that customers make, acknowledging of what to improve about their dishes. The manager runs the

restaurant therefore, they have the ability to control the environment.

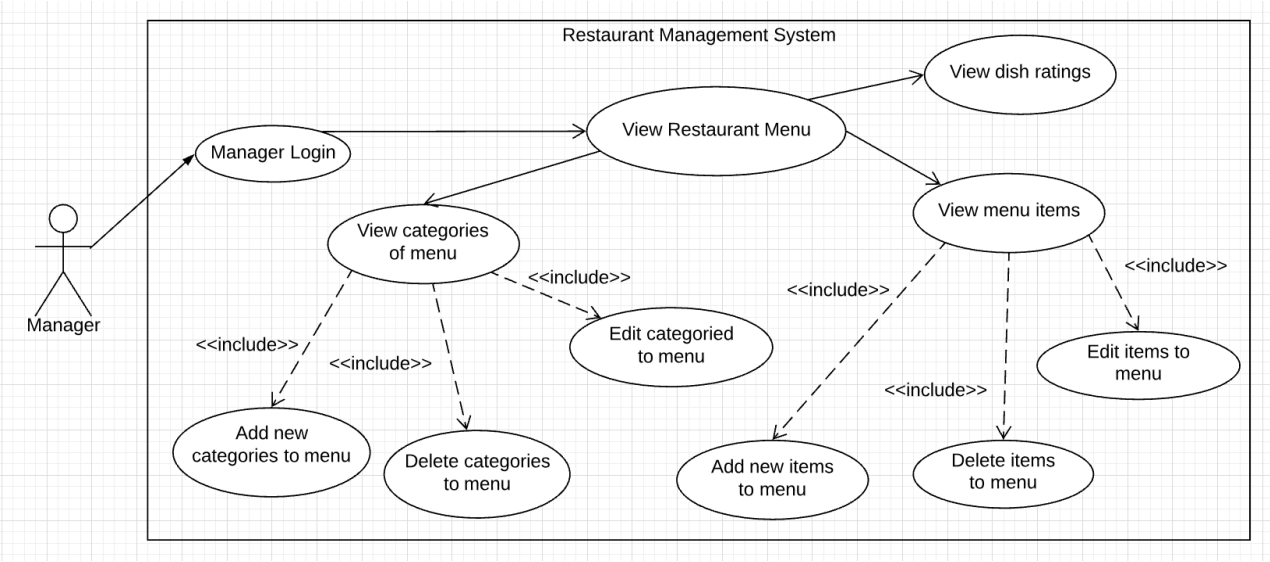

Table 5 Use Case Diagram of Staff Interaction

Main Flow of Third Interaction	
3.1	Managers must login as manager to access management system
3.2	Managers can view restaurant menu
3.3	Managers can view dish ratings by customers
3.4	Managers can add new categories
3.5	Managers can delete categories including dishes that are in a particular category
3.6	Managers can edit name and image of categories
3.7	Managers can view menu items
3.8	Managers can add new dishes
3.9	Managers can delete menu items
3.10	Managers can edit menu items price, name, image and description

Conditions of Third Interaction

Description	This use case will contain the details of roles the restaurant manager. The manager can create, view, update and delete menu items. Managers are responsible for menu management. By capturing customer details in the first iteration managers can view customers review on particular dishes.
Pre-Condition	Managers knows information about menu items.
Post-Condition	Managers can view customer reviews and new menu items added.
Exception Errors	Managers cannot control QR code generation
Alternative variations	Managers must log in as restaurant manager

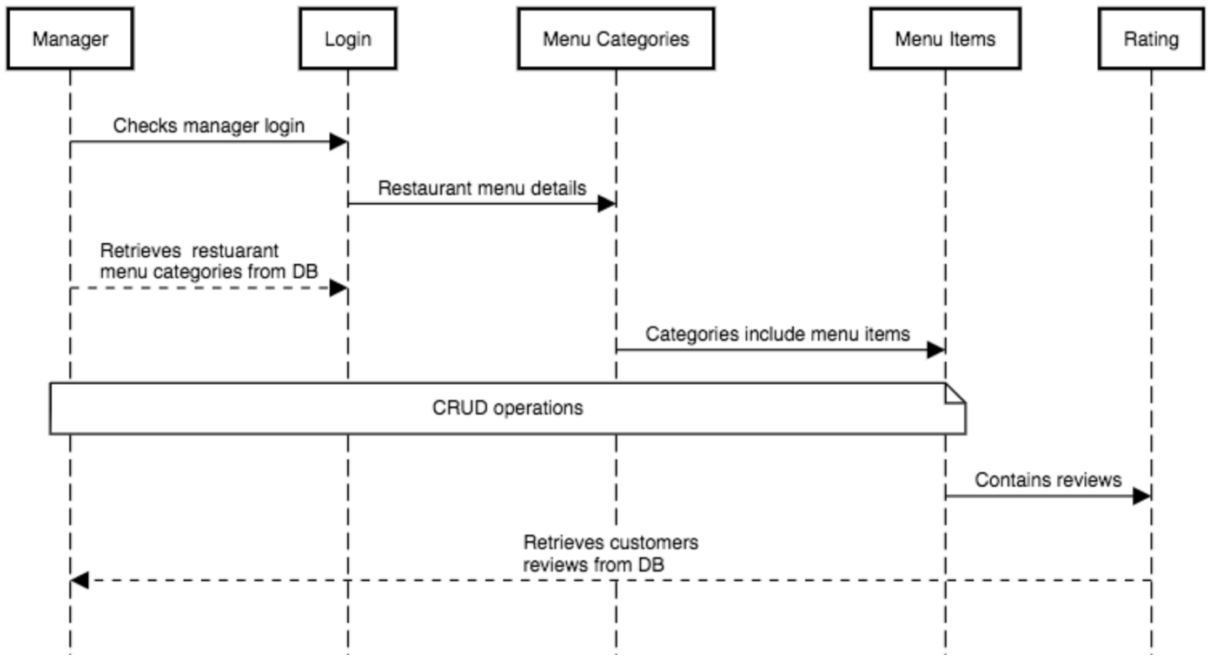

Table 6 Sequence Diagram of Staff Interaction

Managers upon login is greeted with restaurant menu categories which can perform the CRUD operations. Categories also includes menu items which also has functionality of CRUD on each dish.

Each dish contains reviews and ratings made by customers that can be accessed by managers. This can improve restaurant performance and provides feedback mechanism for restaurant managers.

3.2.4 Use Case Scenarios

3.2.4.1 Scenario A

User A comes across a modern casual restaurant of interest. But this user suffers from an inability to speak which makes it difficult to communicate to other individuals. From pre-dining the restaurant advertises a new form of dining with the aid of technology. User A can feel more confident with the use of an application to order food in store while not having to interact with a waiter but still being able to experience a nice meal at a social environment.

User A upon entering the restaurant scans the QR code that is available to scan on a table which will guide them through the restaurant menu application. Once user A has made an order they can make their payment through the application, again relieving user A with any miscommunications and allows user A to control the duration of their dining experience. After the payment is processed the kitchen staff will keep the user updated with their order status.

User A received the dishes that were ordered and feels satisfied with certain dishes they can rate their preferred dish to give the restaurant business feedback. As the order is already paid, user A can directly leave the restaurant after dining, making the post-dining experience convenient.

3.2.4.2 Scenario B

Assuming user B is tech savvy and has prior knowledge on using this application. User B is an extremely busy student that wants to use their meal time to be productive and not wanting any interruptions. User B uses this application in need of a quick and simple meal while still working. User B scans the QR code that is available and orders the items in the menu lists. With the in-app transaction system, user B doesn't have to worry about waiting for the bill post-dining.

User B receives notifications from the kitchen staff of their order. User B's dining experience allows them the time to both enjoy a meal at a restaurant at their own pace while still focusing on their college work.

3.3 Project Structure of Client Application

This section discusses the project structure of the client application and its components. A breakdown of each package is defined and an illustration of the impact of adobe experience designer had on the development of this project. Colour blindness testing software was used to aid in the design of a user friendly interface.

3.3.1 UI Components of Client Application

3.3.1.1 Layout Structure

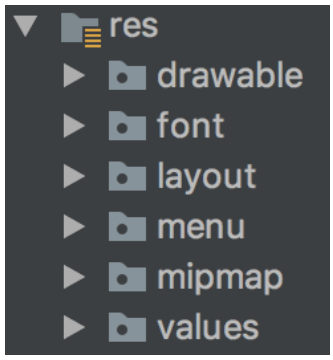

Figure 10 Client Application UI Layout Structure

Above shows android resource directory containing the packages of user interfaces structures. Each package contains different files to indicate each value and layout of certain activities. Below is a detailed description of each package:

"`drawable`": Contains vectors/icons, images and background images that is used throughout the application

"`font`": Main Activity layout contains a cursive font called Lucida Calligraphy.

"`layout`": Contains android XML files that sets the graphical user interface that users can view.

"`menu`": Contains XML files that are used within other layouts. This consists of selected menu items in the navigation drawer and app toolbar drop down menu.

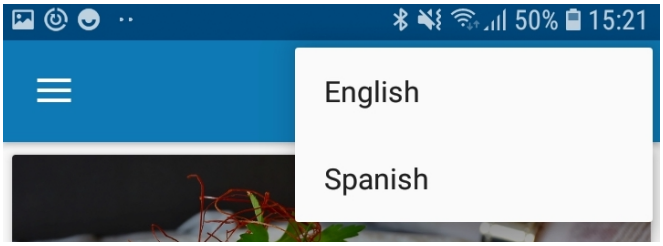

Figure 11 Language Menu

"`mipmap`": Contains app launch icon xml files.

"`values`": Values package contains strings and styles of this project. Strings include different language strings. Styles include app theme, collapsing toolbar, expanded toolbar and other unique customized styles.

3.3.1.2 Experience Designer

Adobe Experience Designer [46] is a software that assists developers to design and deliver a straightforward visual prototype of their end product. This software is extremely beneficial to compare complementing colours and view colour to text ratio contrasts. A simple software can aid developers to design a user friendly graphical user interface

Figure 12 Adobe Experience Design [46]

Using adobe extreme design the initial user interface was outlined. Colour and layouts of button were designed which sets a outlined prototype. Adobe XD provided a user interface testing when designing the initial prototype, the result of the initial design has drastically changed during the progression of development.

There are many benefits in utilizing this tool to design a graphical user interface. These benefits includes the ability to test on a live user interface kit like IPhone or an android device and adobe XD provides usability testing during prototyping before the

application is in its development phase, this ensures that the design is fit for usability and accessibility. This tool can be a great advantage in application development as it reduces an incredible amount of time and workload during implementation.

Below shows a comparison of the initial design using XD and the final design implemented with android XML. It is clear that there were many changes made from the initial design before a more acceptable and user friendly design was created.

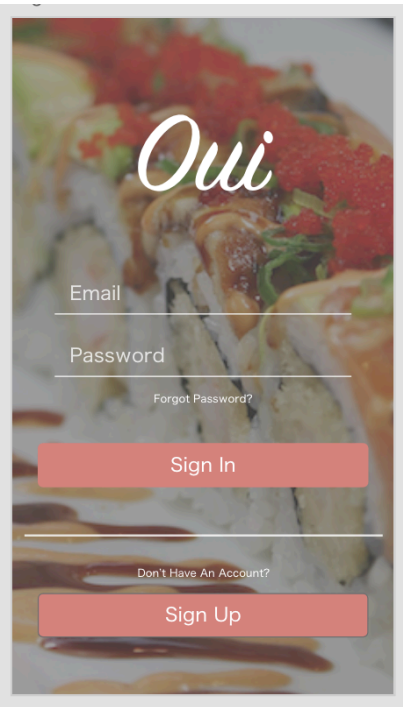

Figure 13 Client UI Before Implementation

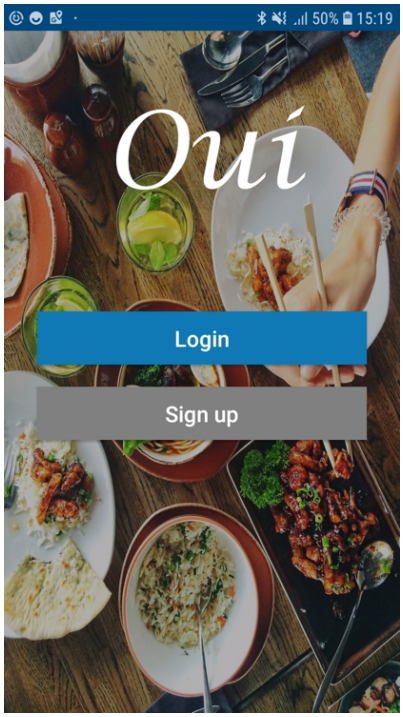

Figure 14 Client Application After Implementation

The comparison of the initial and final design both have similarities and differences. The main difference is the colour contrast. The prototype version seems too grey toned and is difficult to view the edit text inputs whereas the final product has distinctive, clearer images to differentiate the buttons.

3.3.1.3 Colour Blind Accessibility

While choosing the colour theme of this application, colour blind accessibility were taken into account. Users with colour blind impairments consider the colour contrast to be significant. Colour contrast between background colour and overlaid text should follow a 4.5:1 ratio [8] to maintain a high but not too harsh colour contrast. Using a colour blind filter program [47], views of different types of colour blindness

was tested.

Below are the results of the three categories of colour-blindness that were tested:

Type A: Users that are blind to warm red tones shows more cooler colours

Type B: User that are blind to green tones shows a bit of red and blue undertones

Type C: Users that are blind to blue tones shows high contrast of red and green

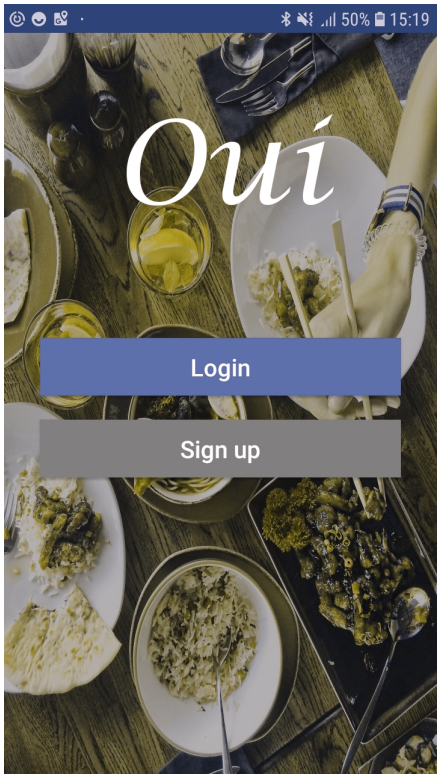

Figure 15 A - Red Blind Protanopia

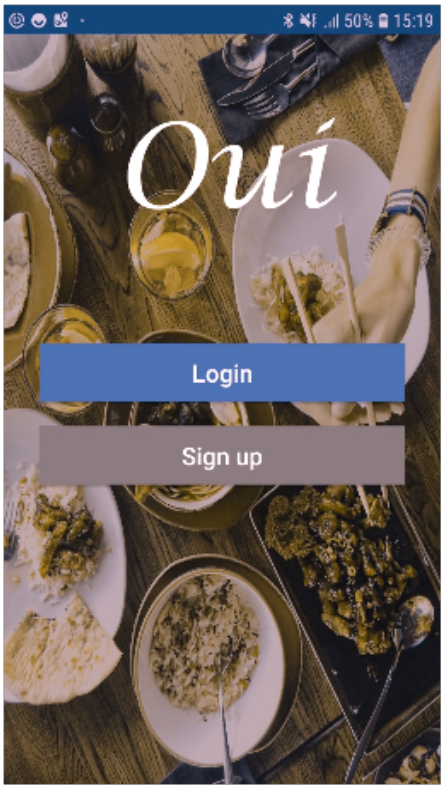

Figure 16 B - Green Blind Deuteranopia

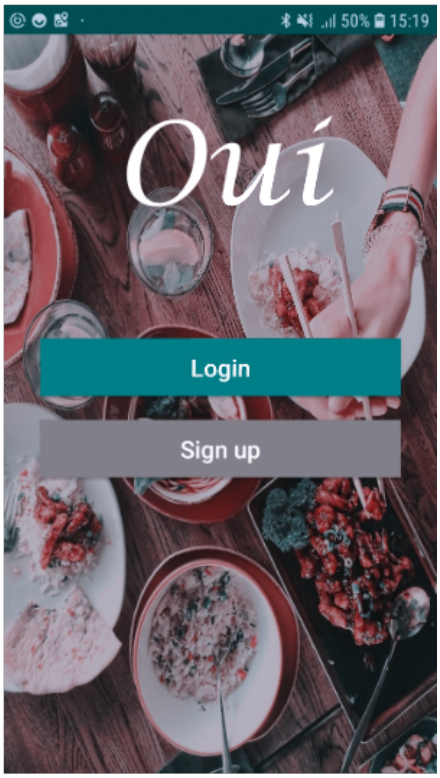

Figure 17 C - Blue Blind Tritanopia

3.3.2 Package Structure of Client Application

In the main java resource directory of the client application it contains several of packages with common functionalities. Each package contains java files to perform certain roles in the application, below is a list of each package sorted in alphabetical order.

3.3.2.1 Package Structure of Client App

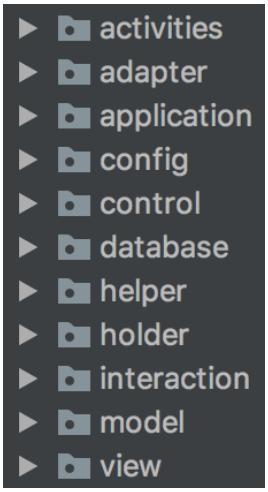

Figure 18 Package Structure of Client Application

“activities”: Contains all activities including its corresponding layout files.

“adapters”: Contains the java class for basket items that the user has added to their basket. This adapter package acts as a bridge between UI components and its functionality.

“application”: Contains the java class to integrate translation functionalities.

“config”: Configuration package short for configuration contains retrofit REST client service. Acts as a communication bridge between the two applications

“database”: Contains SQLite database helper that performs the SQLite queries.

“helper”: This helper package holds the java class to notification manager and localisation helper which holds the java class for translation properties.

“interaction”: This package carries the classes for FCM, firebase cloud messaging services.

“model”: Model package holds classes that calls the data items. Using getters and setters.

“view”: View package contains an interface to set a listener.

3.4 Project Structure of Server Application

This section of project development shows the progression of implementing the server application. The server application follows a similar format as the client application.

Initially the server application was primarily focused on designing to aid businesses to operate their restaurants as a point of sale system but after prototyping with adobe XD, a more portable and flexible application was developed.

3.4.1 UI Components of Server Application

3.4.1.1 Layout Structure

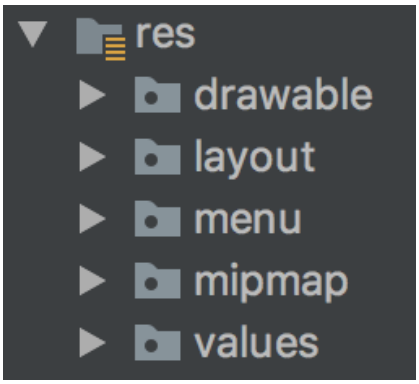

Figure 19 Server Application UI Layout Structure

Package layout of the resource directory are similar to client application with the exception of font package. Font is not an essential part of the server application, as implementing a clear and simple visual design to the merchant application provides uncomplicated guidance for clients. A simple and sophisticated design is implemented in the server application to increase business productivity.

3.4.1.2 Experience Design Server App

Using Adobe XD [46] software an initial prototype of the server application was designed. Compared to the client application there is a substantial loss in visuals with a limited colour palette. Choosing a simple theme can stimulate productivity and using a white background representing neutrality. The result of a clean and minimalistic front-end design allows efficiency and smooth navigation which in return can increase business opportunities.

The images below shows a comparison between the initial design and the final product of the server application. It is evident that there was a drastically change to the user interface. To complement the client application similar colours were used to implement the user interface of the server application, cerulean blue and a soft background colour produces a basic yet elegant layout.

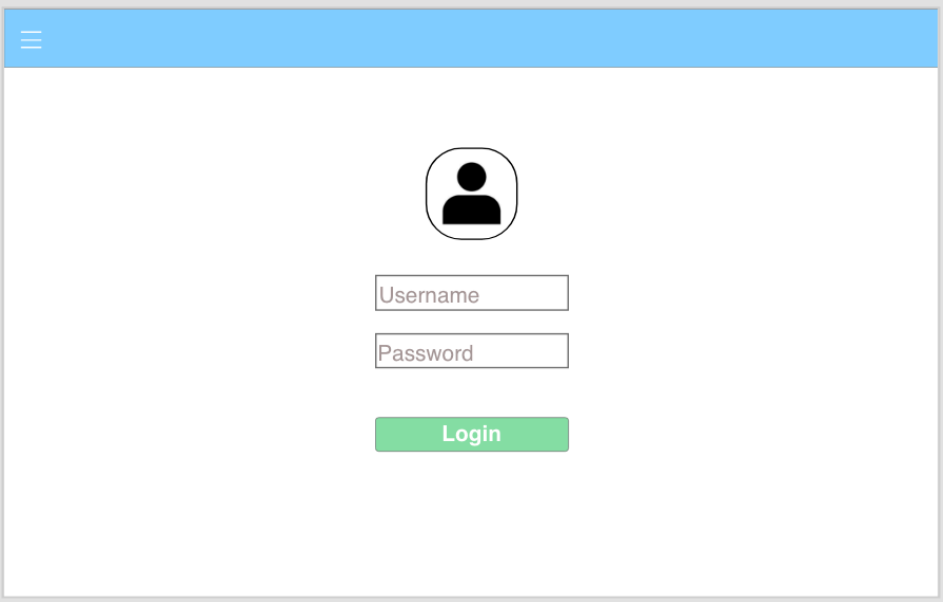

Figure 20 Server Application Before Implementation

Figure 21 Server Application After Implementation

Unlike the client application there is little graphics and visuals involved, this is because it enhances and increases productivity. To manage a business successfully there is no time to waste on irrelevant designs and implementations. Staying on track with as little distraction as possible can assure business owners productivity and

success.

By keeping the user interface design to a minimal can enhance its main role and functionality as a restaurant management system. Staff has fast access to incoming orders without any distractions and complications.

3.4.2 Package Components of Server Application

3.4.2.1 Package Structure of Server App

The structure of the server application is similar to the client application minus database package and application package. Database package is not needed because the main backend of the server application is firebase database. The application package is also not needed as the default language of restaurant management system is set in English although multilingual is not implemented in the server application it can be integrated in the future.

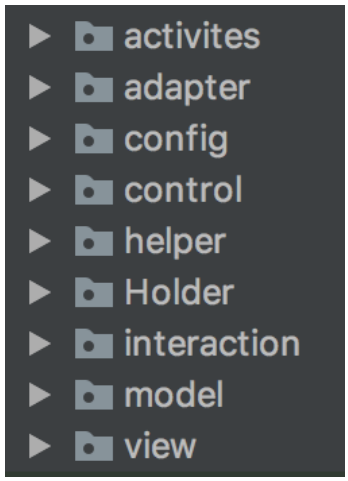

Figure 22 Package Component of Server Application

3.4.3 Overall Application Structure

An overall application structure was designed to assist the development in sections. Each section provided roles and actions that is to be performed by a certain user. The diagram below shows the overall project structure. It consists of three actors user, staff and administrator.

- The user actor operates on the client side application which has direct connection to the firebase database, also has a direct link to a temporary database storage (SQLite DB Browser [6]). The client application uses third

party APIs to perform certain functionalities.

- Restaurant staff operates the server application which performs menu management functionality and it also communicates directly with firebase database.

- The administrator actor, which is most likely to be the developer, will have access to the whole system. The administrator performs system maintenance, updates and provide assistance to both the client and server application users.

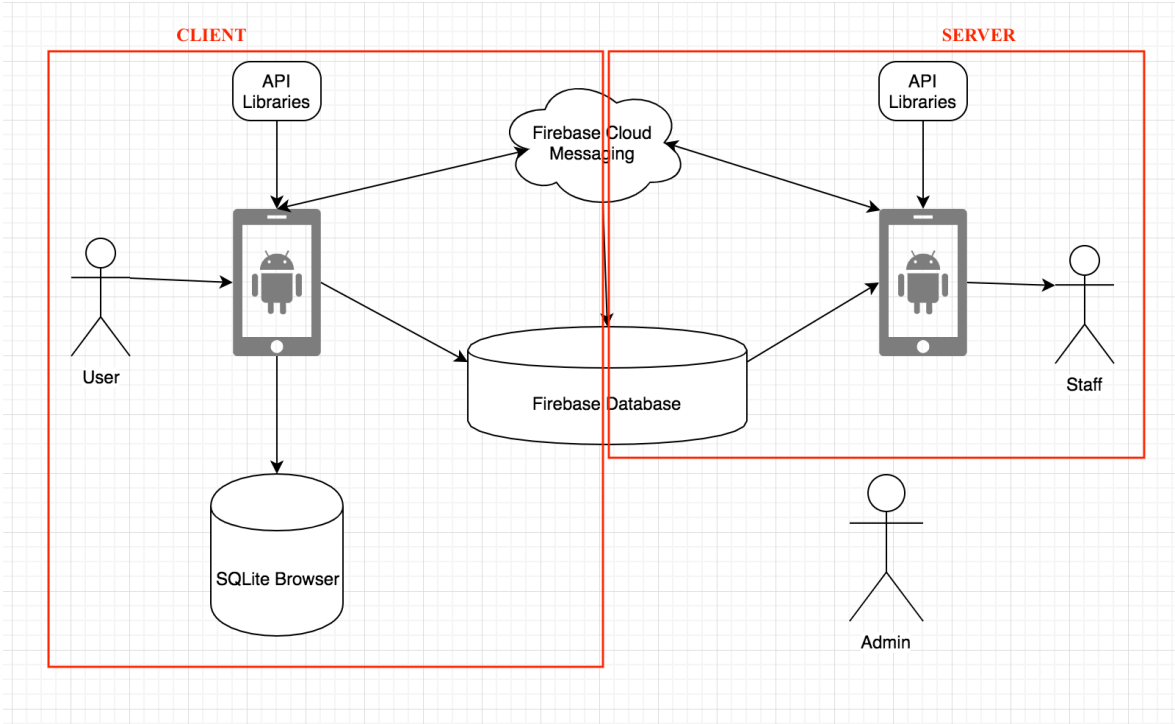

Table 7 Application Structure Diagram

4. Architecture & Design

4.1 System Architecture

Following the application structure diagram in the section above, a complete system architecture design is created to equip the Model, View, Controller (MVC) [48] pattern during development.

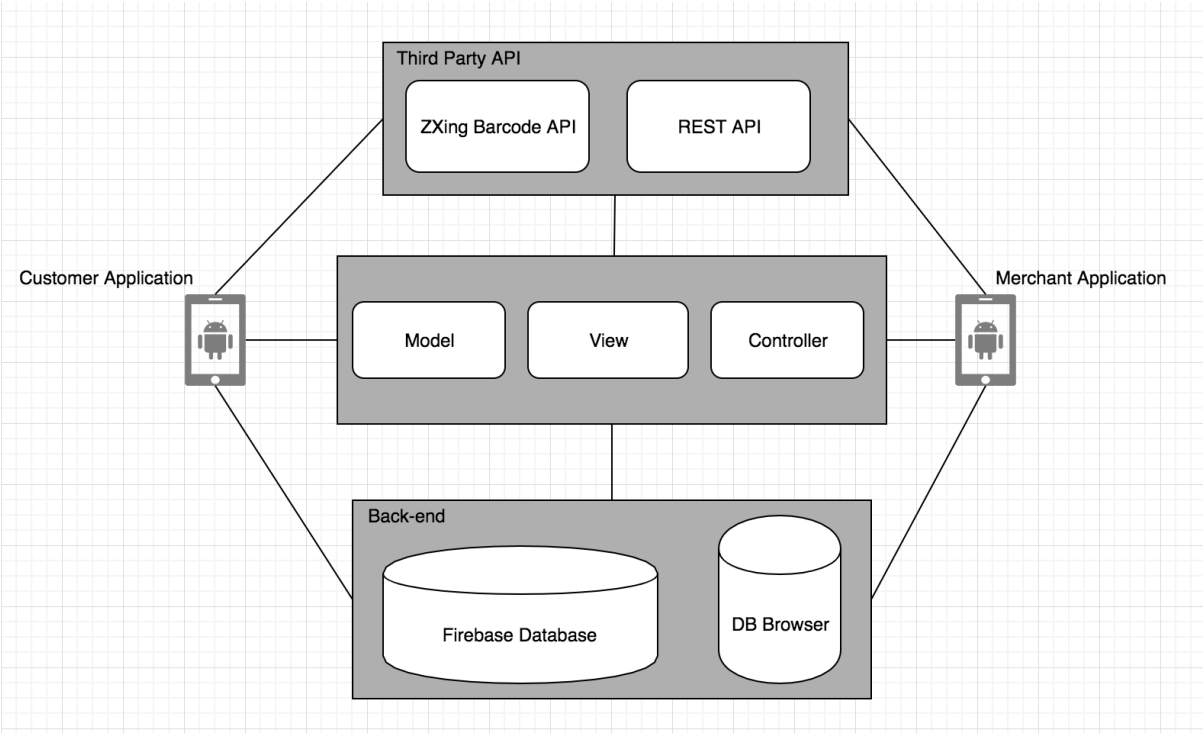

Table 8 System Architecture Diagram

The diagram above outlines the overall system architecture with the top layer consisting of any third-party APIs used, the second layer shows MVC pattern of which the android environment is executed. Finally, the last layer is the back end where all the data is stored using firebase [5] real-time database and DB Browser [6] for SQLite. The diagram also shows the two applications that are developed for customers and merchants. The whole system is then executed with an android operating system device.

4.1.1 Third Party APIs

4.1.1.1 ZXing Barcode Scanner

ZXing scanner or also known as Zebra Crossing [3] is a fast and straightforward way

to integrate barcode scanner library. This library is then called when the application detects and decodes a barcode or QR code. By declaring the dependencies below in the gradle file will allow the functions to implement a barcode scanner.

```
implementation 'com.google.zxing:core:3.3.3'
implementation 'me.dm7.barcodescanner:zxing:1.9.13'
```

Code Snippet 1 ZXing Barcode Scanner API

In android manifest the following permissions were granted to authorize the device camera to be used. This permission will be asked to the user upon their initial use of this application.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Code Snippet 2 Android Manifest Camera Permission

4.1.1.2 RetroFit2 REST API

Retrofit API [4] is an open source library to enable HTTP communication for android applications developed by Square Inc. Retrofit uses OkHttp to make requests to get a response from the server, it also translates JSON responses to POJO (Plain Old Java Object). In order to do this, translated data is sterilized by using a GSON converter which will respond to the object that is needed. By initializing the following gradle dependencies enable the use of this API with the aid of GSON converter.

```
implementation 'com.squareup.retrofit2:retrofit:2.5.0'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Code Snippet 3 RetroFIt2 REST API

4.1.1.3 Facebook API

The Facebook API [49] used in this project allows users to share a selected restaurant and post on the customer's Facebook page. This API provides the functionality of customer made advertisement, by implementing this feature into this application can allow the restaurant to promote itself without spending an extra expenses in marketing. The Facebook share dependency is applied to the gradle file to allow the use of this function.

4.1.1.4 YouTube Android Player API

The YouTube player API [50] can call a specific link to retrieve a video from YouTube which can be played within the application. This API provides the user with a video demonstration that can be viewed in a YouTube Player View. This video which is implemented in the about activity page, demonstrates the application's usability and functions, providing customers with a guide to operate the application.

4.1.2 Model View Controller (MVC)

The model view controller architecture (MVC) [48] assisted to the design of an interactive system that can be flexible to change. MVC consists of three parts, the first would be the model. The model communicates directly with firebase database which contains the data of the application. Any changes made to the model correspondingly changes the view to show the user of a new update. The second part is the view, this is the user interface displayed to the user which is developed in Android XML. The last part is the controller, this controls the user input and according to the inputs the view and/or model might change.

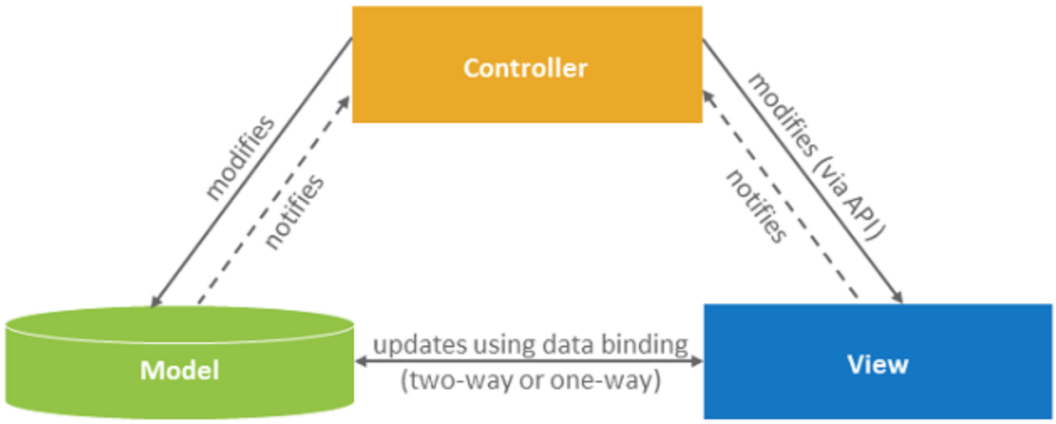

Figure 23 Model, View, Controller [48]

From this architecture, the system will consist of the customer (view), the manager (view), database(model). The controller will update any data that has been changed by both views to model, once the update is done the views will automatically update. With the three environments working together it creates a smooth application.

4.2 Client Application Implementation

The client application is implemented to assist users with their dining experience in a restaurant. The following sections below demonstrates the functionalities of activities that assisted in creating the final product.

4.2.1 Activities

4.2.1.1 AboutActivity.java

About activity contains brief information about the selected restaurant and this application. Information including restaurant name, address and phone number are displayed here. A demonstration video is embedded into this activity using YouTube player API [50] to provide users with a guidance when using this application. This can be beneficial to first time users who are unfamiliar to the application.

4.2.1.2 BasketActivity.java

Basket activity contains functions and information that is in the users shopping cart. The main ordering functionality takes place in this java file. Operations include an Alert Dialog which requests the customer to enter the table number, custom order comments and select a payment method.

PayPal transactions are performed by the try and catch exception that allows a new payment confirmation function to be called which opens the PayPal payment activity. PayPal will then confirm the customers credentials before processing the new order made by users. Once the payment transaction is approved by PayPal Sandbox [39], it returns the user back to the client application and a new order is made by retrieving customer details, a list of dishes ordered, total price, JSON object response state from PayPal, the restaurant and the current date and time.

`notifyServer(order_num)` is called to send a notification to the server application notifying the restaurant staff on new incoming orders. This function will request the remote API service to create a token, if not already, to notify the server application that an order has been placed using the call and response operation provided by the Retrofit REST API [4].

```
Token token = postSnapShot.getValue(Token.class);
Notification notification = new Notification( title: "OUI", body: "New Order "+order_num);
Sender sender = new Sender(token.getToken(), notification);
remoteAPIService.sendNotice(sender).enqueue(new Callback<Response>() {
    @Override
    public void onResponse(Call<Response> call, retrofit2.Response<Response> response) {
        if(response.body().success == 1){
            Toast.makeText( context: BasketActivity.this, text: "Order Placed", Toast.LENGTH_SHORT).show();
            finish();
        } else{
            Toast.makeText( context: BasketActivity.this, text: "Error something happened!", Toast.LENGTH_SHORT).show();

        }
    }

    @Override
    public void onFailure(Call<Response> call, Throwable t) {
        Toast.makeText( context: BasketActivity.this, text: "Error something happened!", Toast.LENGTH_SHORT).show();
        Log.e( tag: "Error", t.getMessage());
    }
});
```

Code Snippet 4 BasketActivity.java, Client App

4.2.1.3 HomeActivity.java

In the home activity the restaurant ID is passed from the previous activity, scan activity, and stored in a temporary string which is called repeatedly to retrieve the data that is enclosed with a restaurant ID.

```
Intent intent = getIntent;

Control.restID = intent.getStringExtra(Control.Restaurant_scanned);
```

Home activity contains a navigation drawer which is implemented to guide users throughout the application. In the main content of the home activity, a recycler view showing the restaurant menu categories. Users are able to see images of categories of foods. The content of the home activity also contains the search function which is discussed in section 64, this search function searches all dishes that are in a selected restaurant. The top application bar also contains a drop down menu were users can change the language of the application to either English, Spanish or simplified Chinese.

In the navigation drawer, customers can direct to other activities which include the scan QR activity, profile activity, about activity and share restaurant activity. Users can also end their log in session by clicking the logging out button that is implemented in the navigation drawer.

4.2.1.4 *LoginActivity.java*

Login Activity is a basic logging in functionality. Retrieving users data from firebase and compare this data with user input to authenticate users password. This activity also searches if the user exists. If user exits the enters to scan activity, else request customers to register in the sign up activity.

4.2.1.5 *MainActivity.java*

Main activity consists of three buttons. The first button will bring users to the login activity on click, the second button brings the user to the sign up activity on click and lastly the third button opens an alert dialog that allows users to change the default language of the application. The code snippet below shows the alert dialog that contains the language selection feature. `SetSingleChoiceItems` allows the user to only select one language to be displayed.

```
dialog.setTitle("Choose Language...");
dialog.setSingleChoiceItems(listLang, checkedItem: -1, (dialogInterface, i) → {
        if(i == 0){
            Paper.book().write("language", "en");
            updateLanguage((String)Paper.book().read( key: "language"));
        }
        if(i == 1){
            Paper.book().write("language", "es");
            updateLanguage((String)Paper.book().read( key: "language"));
        }
```

Code Snippet 5 MainActivity.java, Client App

`updateLanguage((String)Paper.book().read("language"));`

This function is called to update the default language using Paper DB [51] to retrieve the language codes. Paper DB [51] is a NO-SQL object storing option for android, an ideal DB storage to retrieve different in-app locales at a fast pace.

4.2.1.6 *MenuDetailActivity.java*

Menu detail activity contains information about each dish including the price, description and its reviews. This activity allows users to add items to their cart and set the quantity of each item added. The quantity is implemented using an elegant number button and stored in the SQLite DB [6] browser temporarily.

Customers are also able to rate each menu item using app rating dialog as shown in

the code snippet below. This dialog will ask the user to rate a menu item from one to five stars and leave a review comment.

```
new AppRatingDialog.Builder().setPositiveButtonText("Submit")
        .setNegativeButtonText("Cancel")
        .setNoteDescriptions(Arrays.asList("Poor","Not Great","Meh","Great", "Amazing"))
        .setDefaultRating(1).setTitle("Write a review")
        .setDescription("Please give this a rate and comment")
        .setHint("Comments here")
        .setHintTextColor(R.color.grey)
        .setCommentTextColor(android.R.color.black)
        .setCommentBackgroundColor(R.color.white)
        .setWindowAnimation(R.style.RatingAnim)
        .setTitleTextColor(R.color.colorAccent)
        .setDescriptionTextColor(R.color.colorAccent)
    .create(MenuDetailActivity.this).show();
```

Code Snippet 6 MenuDetailActivity.java, Client App

This rating bar is calculated by retrieving the counting the amount of reviews placed from firebase and setting a value listener, it listens to changes that may alter a new average. `DataSnapshot` retrieves the number of reviews and calculates the star rating average based on the sum of all ratings which is shown in menu detail activity.

```
if(count != 0 ) {
    float average = sum / count;
    ratingBar.setRating(average);
}
```

Code Snippet 7 MenuDetailActivity.java, Client App

4.2.1.7 MenuListActivity.java

Menu list activity contains a brief description and price of each dish on the menu. This activity is entered once the user clicked a certain category in the previous home activity. Firebase [5] will get the reference of:

`db.getReference("Restaurant").child(Control.restID).child("details").child("menu")`

This reference's the child of the current restaurant ID which contains the details of the restaurant's menu. To view a list of items a recycler view Firebase Recycler Options and Firebase Recycler Adapter are declared to adapt a recycler view to ensure data is shown in this layout. An adapter listener is implemented to listen and operate the recycler view.

```
options = new FirebaseRecyclerOptions.Builder<Menu>().setQuery(query, Menu.class).build();
adapter = new FirebaseRecyclerAdapter<Menu, MenuHolder>(options) {
```

Code Snippet 8 MenuListActivity.java, Client App

4.2.1.8 *OrderPlacedActivity.java*

Order placed activity retrieves order history data from database, showing users of all orders placed previously at the selected restaurant. Users are able to cancel an order before the status of the order changes to '`cooking in kitchen`'. Once the order has been processed and confirmed the user will not be able to cancel their order. This can avoid any scams and is a security precaution.

```
if(adapter.getItem(position).getStatus().equals("0")){
    cancelOrder(adapter.getRef(position).getKey());
}else{
    Toast.makeText( context: OrderPlacedActivity.this,  text: "Order can not be cancelled", Toast.LENGTH_SHORT).show();
}
```

Code Snippet 9 OrderPlacedActivity.java

4.2.1.9 *ProfileActivity.java*

Profile activity shows the current user's details such as their name, email and phone number. Users are also able to update their old password and delete their account with a simple click of a button.

4.2.1.10 *ScanActivity.java*

This activity performs the barcode scanner using ZXing API library [3]. The unique feature of this application is developed by implementing a QR code scanner. This activity allows users to scan the QR code at a restaurant which will then retrieve the restaurant's menu items. By following the instructions on the activity, users can operate this function freely.

```
    IntentIntegrator intentIntegrator = new IntentIntegrator( activity: this);
    intentIntegrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES);
    intentIntegrator.setCameraId(0);
    intentIntegrator.setBeepEnabled(true);
    intentIntegrator.setOrientationLocked(false);
    intentIntegrator.setBarcodeImageEnabled(false);
    intentIntegrator.initiateScan();
```

Code Snippet 10 ScanActivity.java, Client App

Scan QR function is called when the scan button is pressed. `IntentIntegrator` opens the mobile device's camera which will then read in the barcode. For this function, the desired barcode format is set to `QR_CODE_TYPE`. Intent Integrator will ask for the user's permission to allow the use of their device camera to enable this scanning functionality. In the android manifest file this permission is asked, when the user has installed this application initially the application will ask the user for permission of their camera.

```
protected void onActivityResult(int requestCode, int grantResults, Intent data) {
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode, grantResults, data);
    if (result.getContents() != null) {
        String restID = result.getContents();
        Intent intent = new Intent( packageContext: this, HomeActivity.class);
        intent.putExtra(Control.Restaurant_Scanned, restID);
        startActivity(intent);
```

Code Snippet 11 ScanActivity.java, Client App

The content of the scanned result will be parsed into `onActivityResult` function shown above and stored in the control activity. In the control activity the string of `result.getContent` is stored for a period of time until the customer logs out or has scanned another QR code. Once the scanned string is stored it can then be accessed by other activities just by calling `Control_restID` string.

4.2.1.11 SearchFoodActivity.java

This activity performs the search functionality on data that is stored in firebase [5]. Material search bar is integrated into the design and implementation. Using material search bar, searching in large data storages can be more efficient and more interactive for the users. This search bar provides recent searches, auto-complete and listener for single value events.

```
implementation 'com.github.mancj:MaterialSearchBar:0.8.2'
```

Code Snippet 12 Material Search Bar Library

4.2.1.12 SeeReviewsActivity.java

See review activity, shows a list of reviews and comments about a certain dish at a certain restaurant. Usernames are shown, identifying each user who has reviewed an item on the menu.

4.2.1.13 *ShareRestaurantActivity.java*

This activity allows the use of Facebook API [49] to share a restaurant post on the customers Facebook page. By creating a target, a bitmap share photo is initialized which retrieves a Picasso image from the activity and parsed to be displayed in Facebook. The following statement performs this function

```
Picasso.get().load(model.getImage()).into(target);
```

The executed output of this activity shows this result.

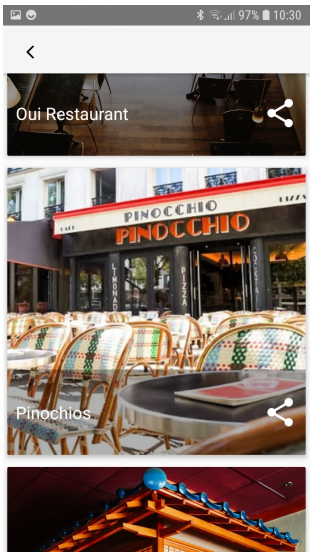

Figure 24 Share Restaurant Activity

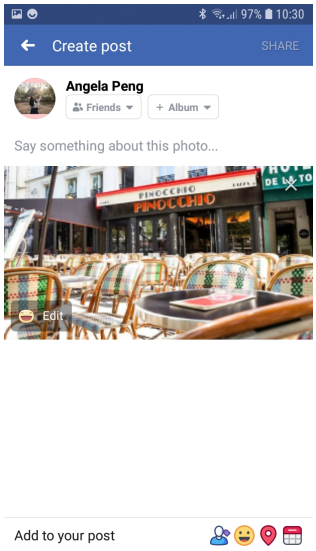

Figure 25 Share Restaurant Facebook Activity

4.2.1.14 *SignUpActivity.java*

Sign up activity provides a basic registration form. Once the user is signed up a verification code is needed to authenticate a user. This is sent by implementing firebase email and password authentication [29].

4.2.2 Adapter

4.2.2.1 *BasketAdapter.java*

Basket adapter extends the recycle view <`basket_holder`>. Basket adapter holds a list of menu items that there were added. Users can then delete an item from the basket by on hold click on a selected item. Along with functions in basket activity, illustrated in section 59, the operations to perform the role of a shopping cart is implemented.

```
@Override
public void onCreateContextMenu(ContextMenu contextMenu, View view, ContextMenu.ContextMenuInfo contextMenuInfo) {
    contextMenu.setHeaderTitle("Choose Action");
    contextMenu.add( i: 0, i1: 0,getAdapterPosition(), Control.delete);

}
```

Code Snippet 13 BasketAdaper.java, Client App

4.2.3 Application

4.2.3.1 LanguageApplication.java

This file extends the application which sets the default language to English. This acts as a base application file and is implemented throughout the activities that assist translation operations. `attachBaseContext` defines the default language to English.

```
@Override
protected void attachBaseContext(Context base) {
    super.attachBaseContext(LocalHelper.onAttach(base, defLanugage: "en"));
}
```

Code Snippet 14 LanguageApplication.java, Client App

4.2.4 Config

4.2.4.1 RemoteAPIService.java

HTTP requests are made in this interface from a remote API service, in this case Retro Fit [4]. As declared in the header, the type of application and authorization key are set. Authorization key is the key that is set for this application on firebase cloud messaging(FCM) [28] and can be found in firebase console.

```
public interface RemoteAPIService {
    @Headers(
            {
                    "Content-Type:application/json",
                    "Authorization:key=AAAAZd_f7b0:APA91bGriF-XKzzLN04lyznsuAmPTyvpZPlS80K3h
            }
    )
    @POST("fcm/send")
    Call<Response> sendNotice(@Body Sender body);
```

Code Snippet 15 RemoteAPIService.java, Client APP

To send notifications HTTP requests directs all responses to FCM [28]. This connection between FCM, client application and server application is connected by REST Retrofit API [4] defined by a base URL link:

```
https://fcm.googleapis.com/fcm/send
```

4.2.4.2 *RetroClient.java*

In this java class conversion to GSON is created. This file provides the Retrofit [4] instance if it exists, else a new instance is created and returned.

```
if(retrofit == null){
    retrofit = new Retrofit.Builder().baseUrl(baseURL).addConverterFactory(GsonConverterFactory.create()).build();
```

Code Snippet 16 RetroClient.java, Client App

4.2.5 Control

4.2.5.1 *Control.java*

This file controls and stores temporary strings like current user ID and current restaurant ID. Control file is used throughout the activity files to set the data of users that is currently logged in and the restaurant ID that the user has scanned.

```
public static boolean checkConnectivity(Context context){
    ConnectivityManager connectivityManager =  (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
    if(connectivityManager != null){

        NetworkInfo[] infos = connectivityManager.getAllNetworkInfo();

        if(infos != null){
            for(int i =0; i < infos.length; i++){
                if(infos[i].getState() == NetworkInfo.State.CONNECTED);
                {
                    return true;
                }
            }
```

Code Snippet 17 Control.java, Client App

This file also implements the functionality of connectivity manager that is used to check for internet connection. As this application uses firebase as a back-end server, data is retrieved through an internet connection. HTTP calls and responses are also performed with the use of internet, without this connection this application cannot perform to its potential.

4.2.5.2 *Paypal.java*

Using PayPal sandbox developer [39] an application is created. PayPal sandbox developer initializes a unique key which identifies a certain business application. This key is then implemented into the PayPal java class to allow payments to be made to a certain business application. Virtual payments can be made directly through the PayPal payment activity from a virtual personal account to the business account.

```
public static final String paypal_ID = "AcWbZVQ6Cvl0SwaVRxsi5Bs7UK-y_9dbIgwBd_wWXQELQF-HvS-dTkFxZh2hbsGj40svWxa0PXJJGOe0";
```

Code Snippet 18 Paypal.java, Client App

4.2.6 Database

4.2.6.1 Database.java

The database file extends `SQLiteAssetHelper`, this java file performs simple SQLite queries that loads basket items from SQLite DB Browser [6] to the activity for users to access. Functions include add, delete and update basket items. The sample code below shows add to basket function that performs SQLite query. The query performs a simple `INSERT` statement into the `OrderDetail` table.

```
public void addToBasket(Order order){
    SQLiteDatabase sqLiteDatabase = getReadableDatabase();
    String query = String.format("INSERT OR REPLACE INTO OrderDetail(UserPhone, ProductId, ProductName, Quantity, Price) " +
                    "VALUES('%s', '%s', '%s','%s','%s');",
            order.getUserPhone(), order.getProductId(), order.getProductName(), order.getQuantity(), order.getPrice());
    sqLiteDatabase.execSQL(query);

}
```

Code Snippet 19 Database.java, Client App

The `getOrderBasket` function in database.java requests the database to retrieve information of a certain users shopping cart items. This is identified when the user ID is the current users ID and restaurant ID is the key that is encrypted into the current scanned QR code. To simplify the SQLite query below, it selects all data from the order detail table where the user ID is the current user and where rest ID is the current restaurant.

```
public List<Order> getOrderBasket(String userPhone, String restID){
    SQLiteDatabase sqLiteDatabase = getReadableDatabase();
    SQLiteQueryBuilder sqLiteQueryBuilder = new SQLiteQueryBuilder();

    String[] select = {"UserPhone", "ProductId", "RestID","ProductName","Quantity", "Price"};
    String sqlTable = "OrderDetail";

    final List<Order> result = new ArrayList<>();
    sqLiteQueryBuilder.setTables(sqlTable);
    Cursor cursor = sqLiteQueryBuilder.query(sqLiteDatabase, select,  selection: "UserPhone = ? AND RestID = ?",
```

Code Snippet 20 BasketActivity.java, Client App

4.2.7 Helper

4.2.7.1 LocalHelper.java

To set the locale of the application shared preferences is implemented using persistent data to store a selected language. Strings are updated by configuring the

locale of the application.

```
private static void persist(Context con, String lang) {
    SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(con);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SELECTED_LANGUAGE, lang);
    editor.apply();
}
```

Code Snippet 21 LocalHelper.java, Client App

Shared Preferences [52] are key-value pairs of primitive data types that are saved in a file within an app file structure. In this case, the file that is stored contains string texts of different languages.

4.2.7.2 NotifyMeHelper.java

Notification helper contains functions to create notification channels which enables android devices to receive incoming notifications from the server. Notifications are identified by notification channel variables that includes the ID and application name.

```
@TargetApi(Build.VERSION_CODES.O)
private void channel() {
    NotificationChannel notificationChannel = new NotificationChannel(OUI_ID, OUI_NAME, NotificationManager.IMPORTANCE_DEFAULT);
    notificationChannel.enableLights(false);
    notificationChannel.enableVibration(true);
    notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

    getNotificationManager().createNotificationChannel(notificationChannel);
}
```

Code Snippet 22 NotifyMeHelper.java, Client App

4.2.8 Holder

4.2.8.1 MenuHolder.java

Menu holder sets the data of recycler view in menu detail activity which identifies a custom layout. Information includes dish name, image, description and price of the item. This information is displayed when the menu list activity is shown in the client application.

4.2.8.2 OrderHolder.java

Order holder contains the recycler view layout of order history activity. List of information includes order ID, order status, total price and table number.

4.2.8.3 ProductHolder.java

Product holder, holds recycler view layout of menu list activity. Menu list holds

categories of menu items including name and image. Which can be retrieved when the home activity is called.

4.2.8.4 *RestaurantHolder.java*

Restaurant holder, holds restaurants that are available to view menu items from. Recycler view contains a list of available restaurants including its name and image.

4.2.8.5 *ReviewHolder.java*

Review holder contains information about reviews users made on a particular dish. Star rating, comment and username. This information is then displayed to view in see review activity by customers.

4.2.9 Interaction

4.2.9.1 *FirebaseMessage.java*

Firebase messaging class extends Firebase cloud messaging service [28]. The code snippet below shows the function `sendNotice` which opens the main activity when the user clicks on a notification. The ringtone manager defines the default alert tone for a certain device.

`NotificationCompat.Builder` builds the layout of the notification that will appear on the users device. The code snippet below shows the notification is designed to show a small application icon with the content title and text. The content title and texts are defined in figure Figure 26 Advanced Rest Client. Once an order is placed the remote API will send a notification to the server, alerting them of new orders in the layout of the notification builder.

```
PendingIntent pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, intent, PendingIntent.FLAG_ONE_SHOT);
Uri uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder( context: this).setSmallIcon(R.mipmap.ic_launcher)
        .setContentTitle(remoteNotification.getTitle())
        .setContentText(remoteNotification.getBody()).setAutoCancel(true).setSound(uri).setContentIntent(pendingIntent);

NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify( id: 0, notificationBuilder.build());
```

Code Snippet 23 FirebaseMessage.java, Client App

Notification title and body are stored in a JSON format and defined by using advanced rest client software as followed:

```
{
  "notification": {
    "title": "Hello",
    "body": "Hello this is body message"
  },
  "to": "<Token receiver>"
}
```

Figure 26 Advanced Rest Client

4.2.9.2 *FirebaseService.java*

Firebase [5] service class creates new tokens for new users to enable a connection for FCM to communicate to the server application. This token provides an authentic communication link between client and server application when making call requests and responses.

```
private void updateToken(String refreshToken) {
    FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
    DatabaseReference databaseReference = firebaseDatabase.getReference( path: "Tokens");
    Token token = new Token( serverToken: false,refreshToken);
    databaseReference.child(Control.currentUser.getPhone()).setValue(token);

}
```

Code Snippet 24 FirebaseService.java, Client App

4.2.10 Model

Model package contains java files to get and set data from database. It acts as a temporary storage facility before the data is set to a new value or retrieves a value. There are twelve java classes implemented within the model activity package in this application. Below discusses some of the model files that are implemented.

4.2.10.1 *MakeOrder.java*

Contains getters and setters of orders that are made to retrieve data from firebase. A list of items that makes an order are listed below. This make order class also includes a list array to store each item that is ordered by the customer.

```
public MakeOrder(String phone, String email, String table,
    this.phone = phone;
    this.email = email;
    this.table = table;
    this.name = name;
    this.total = total;
    this.status = status;
    this.notes = notes;
    this.paymentProcess = paymentProcess;
    this.paymentMethods = paymentMethods;
    this.restaurantId = restaurantId;
    this.listOfOrderPlaced = listOfOrderPlaced;
}
```

Code Snippet 25 MakeOrder.java, Client App

4.2.10.2 Menu.java

Menu model contains getters and setters of menu details activity which can be retrieved by the application to display information that are stored in firebase.

```
public Menu(String name, String image, String description, String price, String foodId) {
    Name = name;
    Image = image;
    Description = description;
    Price = price;
    FoodId = foodId;
}
```

Code Snippet 26 Menu.java, Client App

4.2.10.3 Response.java

This java class shows implementation of and usage of advanced REST client API. This API tests interaction between client and server giving a success true or success false response.

```
public class Response {
    public long multicast_id;
    public int success;
    public int failure;
    public int canonical_ids;
    public List<Result> results;
```

Code Snippet 27 Response.java, Client App

Using advanced REST Client API an authorization key is requested by making a POST call from the server which then a request is made resulting in a response.

4.2.10.4 User.java

User model sets `IsStaff` variable to default false as customers are not staff memebers. Any new user registered will be under the category of customer only which prevents customers to gain access to the server application that is controlled by the restaurant staff members.

```
public User(String name, String email, String password){
    Name=name;
    Email=email;
    Password = password;
    IsStaff = "false";

}
```

Code Snippet 28 User.java, Client App

4.3 Server Application Implementation

The server application is implemented to assist restaurant staff members to manage incoming orders and restaurant menu management. It is developed similarly to the client application with a fewer functionalities. Below discusses some of the main classes that are implemented to the server application.

4.3.1 Activity

4.3.1.1 HomeActivity.java

Home activity shows a recycler view of each categories in a menu, this activity include functions like select image from users device image gallery, upload image and adding new products to the menu.

The home activity performs menu management functions with a touch of a button declared by a floating action button. Select function is implemented as followed, `Intent.createChooser` which opens device's image gallery, allowing users to select a certain image. The code snippet below indicates the select image function. The image selected will then be uploaded.

```
private void selectImage() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent,  title: "Select photo"), Control.PICK_IMAGE_REQUEST);

}
```

Code Snippet 29 HomeActivity.java, Server App

Upload image function uses firebase storage reference to upload a selected image to firebase storage. `addOnSuccessListener` returns a success message of upload completion. `addOnFailureListener` returns a failure message of image. `addOnProgressListener` shows the user a progress bar of the upload status. Once the image is uploaded it is stored as a URL in firebase database storage as declared in the code snippet below. `StorageReference` references the directory that all the images are to be stored.

Error messages are implemented in plain simple text to show the user of a successful or failed image upload.

```
final StorageReference folder = storageReference.child("images/" +imageName);
folder.putFile(saveURL).addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
        progressDialog.dismiss();
        Toast.makeText( context: HomeActivity.this,  text: "Upload completely", Toast.LENGTH_SHORT).show();
        folder.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) → {
                newProduct = new Product_Type(title.getText().toString(), uri.toString());
        });
}).addOnFailureListener((e) → {
        progressDialog.dismiss();
        Toast.makeText( context: HomeActivity.this,  text: "Upload failed", Toast.LENGTH_SHORT).show();
}).addOnProgressListener((OnProgressListener) (taskSnapshot) → {
        double progress = (100.0 * taskSnapshot.getBytesTransferred()/taskSnapshot.getTotalByteCount());
        progressDialog.setMessage("Uploaded" + progress + "%");

});
```

Code Snippet 30 HomeActivity.java, Server App

4.3.1.2 MenuListActivity.java

Menu list activity also performs similar functionality as home activity. Operations include image, name, description and price updates. This management functions offers the general CRUD operations. Users can also delete a dish item by removing menu item ID from firebase database as shown in the code snippet below.

```
menuRef.child(key).removeValue();
```

Code Snippet 31 MenuListActivity.java, Server App

4.3.1.3 OrdersPlacedActivity.java

This activity shows all orders that have been made by the client application. This activity has the ability to change order status from sent to kitchen, cooking and served. Once an order status is changed the send status function is called which will send a notification to the client application.

In the code snippet below indicates the operation to be implement to enable the change status function with the assistance of material spinner design. In the material spinner items are set in terms of status states, in this case, kitchen, cooking and served. Once the user has selected a state it will set the status to the string of the selected material spinner item. Order placed database reference will also update the status key which will then result in a notification to be called if data is changed in firebase.

```
alertDialog.setPositiveButton( text: "Done", (dialogInterface, i) → {
        dialogInterface.dismiss();
        item.setStatus(String.valueOf(materialSpinner.getSelectedIndex()));
        orderPlaced.child(key_STAT).setValue(item);
        adapter.notifyDataSetChanged();
        sendStatus(key_STAT,item);

});
```

Code Snippet 32 OrderPlacedActivity.java, Server App

Notification is sent by call responses using retrofit API service [4]. This allows quick responses between server and client communication.

```
@Override
public void onResponse(Call<Response> call, retrofit2.Response<Response> response) {
    if (response.body() != null) {
        if(response.body().success == 1){
            Toast.makeText( context: OrdersPlacedActivity.this, text: "Order updated!", Toast.LENGTH_SHORT).show();
            finish();
        } else{
            Toast.makeText( context: OrdersPlacedActivity.this, text: "Error something happened!", Toast.LENGTH_SHORT).show();
```

Code Snippet 33 OrderPlacedActivity.java, Server App

4.3.2 Control

4.3.2.1 Control.java

In the control java class current user logged in session is stored, internet connections are checked and order date function is implemented using the calendar

utility. When the user exits the app or logs out the control class variables are set to null again. The date format is set to DD-MM-YYYY HH:MM as shown below.

```
public static String orderDate(long time){
    Calendar calendar = Calendar.getInstance(Locale.ENGLISH);
    calendar.setTimeInMillis(time);
    StringBuilder date = new StringBuilder(android.text.format.DateFormat.format( inFormat: "dd-MM-yyyy HH:MM",calendar).toString());
    return date.toString();
}
```

Code Snippet 34 Control.java, Server App

4.3.3 Holder

4.3.3.1 MenuHolder.java

Menu holder java class contains the recycler view holder of each dish item. This class also implements `onClickListener` which gives user actions to update or delete an item. `onCreateContextMenu` declares these variables in the form of adapter position. If the position is '0', the update function is called otherwise if the position is '1', the delete function is called.

```
@Override
public void onCreateContextMenu(ContextMenu contextMenu, View view, ContextMenu.ContextMenuInfo contextMenuInfo) {
    contextMenu.setHeaderTitle("Choose Action");
    contextMenu.add( i: 0, i1: 0,getAdapterPosition(), Control.update);
    contextMenu.add( i: 0, i1: 1,getAdapterPosition(), Control.delete);

}
```

Code Snippet 35 MenuHolder.java, Server App

4.4 Back End Implementation

4.4.1 Database Structure

Entity relationship diagram (ERD) represents relationships between entity sets that are stored in a database. An entity set [53] is a collection of similar entities which can contain attributes that illustrates its properties. The following ER diagrams shows the relationships between a customer and a restaurant. For simplicity, sections of the entity relationship diagram are separated to illustrate each set of data stored.

Below illustrates two different sections of the entity relationship diagram that this project reflect on when designing the database structure. The first diagram contains the main data structure of the restaurant application (server side). The second diagram shows the data structure of the customer application (client side).

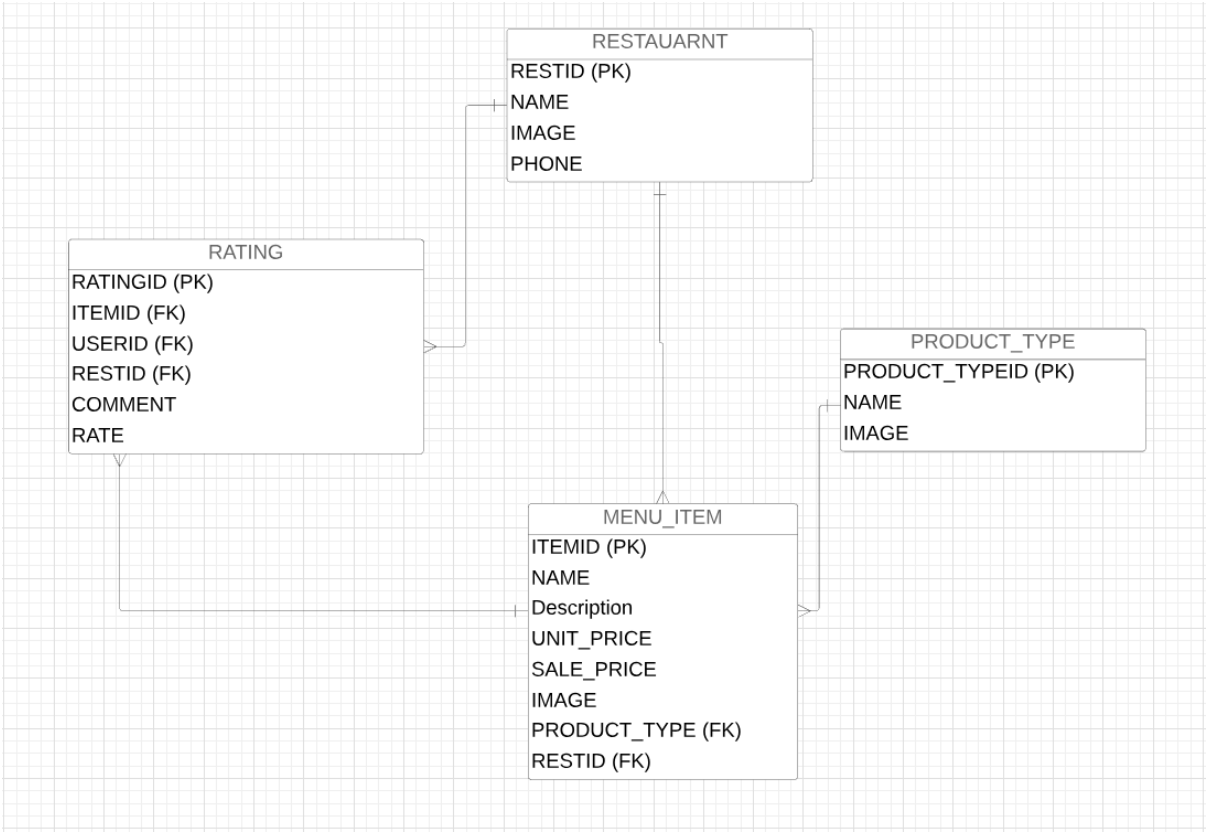

Table 9 Restaurant Data Structure Diagram

Restaurant operation section structure contains tables that include restaurant information, product type, menu item and rating. Each restaurant contains many menu items, product type and rating.

Restaurant entity includes a restaurant ID that uniquely identifies a restaurant when retrieving a certain restaurants data. Restaurant IDs are used to generate the QR code using a QR code generator tool. This entity also contains restaurant name and image that users are able to view when selecting an available restaurant.

Product type entity provides data about different categories that are available on the menu. It contains information of the name and image of different categories of dishes that a restaurant has. A unique identification key is given to product type, this key is then used as a foreign key to menu item's entity.

Menu item entity provides information about each dish that is on the menu. Item ID is unique and is called when users want to view information about each dish. Name, description, price and image are all attributes that are stored in menu item entity.

Menu items also contain the foreign key of restaurant ID and product ID. Each restaurant contains many menu items and each category contains many dishes.

Rating entity stores users input on each dishes. Reviews are set according to star rating, comment and foreign key of users, restaurant and menu items. Each rating has a unique ID, each restaurant contains many ratings for each menu item. Each user can make many reviews for many dishes and each dish contains many reviews.

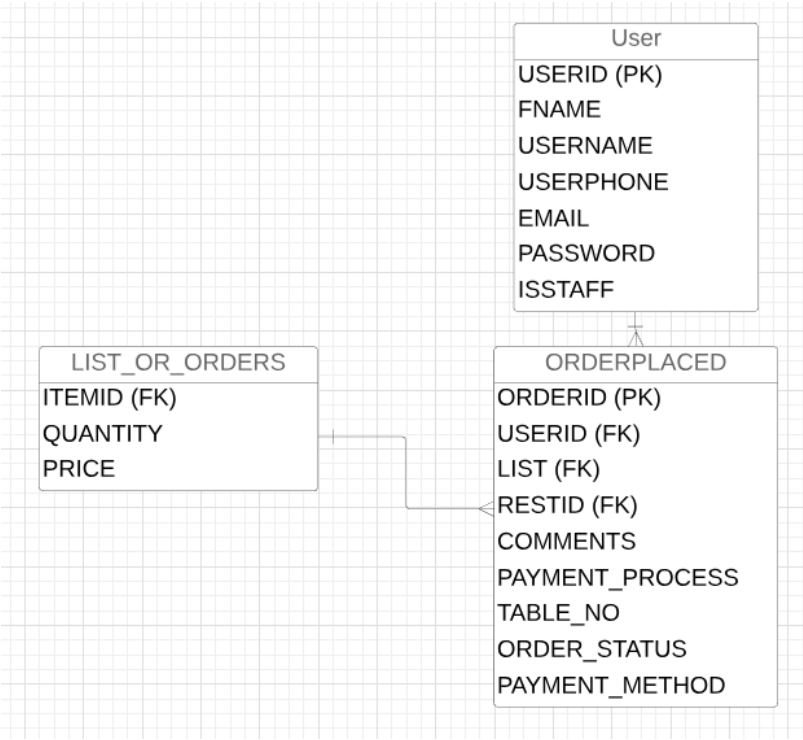

Table 10 Order Placed Data Structure Diagram

Orders placed data structure section contains user, list of orders and order placed entities. Each entity contains a unique identity key, users are identified using these keys. User details can be retrieved by calling a user ID. Users store information including their name, username, user phone, email, password and a Boolean value to indicate if a user is a staff or a customer.

Order placed entity consists of order comments, payment process, table number, order status and payment methods. Additionally this entity incorporates many foreign keys including user ID, list ID and rest ID. Users can write comments into their order to customize their own order. One order can only process one transaction method and process state. Payment process is defined by either paid or unpaid and payment

methods are considered by either PayPal or cash transaction. Table numbers are also set by the user when making an order. Each order status is set to ‘0’ when an order is placed by a user, this status will be changed by a staff member. Users can place many orders, containing many different dishes. Restaurants can receive many incoming orders.

List of orders entity consists of list ID, quantity of item, price and item ID attributes. Quantity depends on the amount of each dish that is placed in an order. Price is calculated by the quantity of a certain dish by adding the price per dish cost.

4.4.2 Firebase

Firebase [5] is a NoSQL database storage facility, it allows data change to be operated smoothly without the need to update the system. Firebase can provide many functionalities including user authentication, real-time data retrieval, firebase messaging and firebase database storage [54] .

Image database storage is used in the server application, this tool allows restaurant managers to upload an image from their device to firebase storage. Once it is stored here data can be read from both client and server application instantaneously. These images can be deleted and re-uploaded.

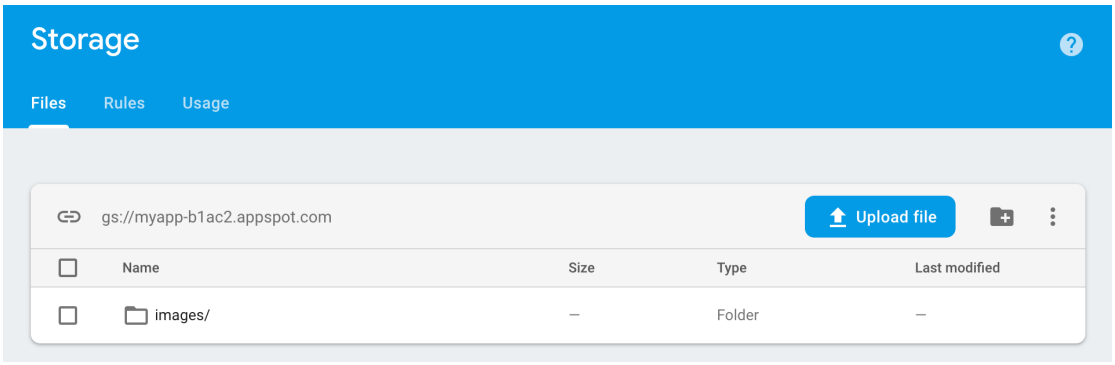

Figure 27 Firebase Storage

The structure of firebase is shown below, each restaurant has a unique ID which contains information about orders placed, menu, product type and reviews. Server tokens are stored based on if the user is a staff or user. If a user is a restaurant customer, server token is set to false while if the user is a staff, server token is set to true.

Figure 28 Firebase Database Structure

Firebase performs multiple functionalities such as authentication [29], firebase cloud messaging [28] and firebase storage. By To integrate these functions the following gradle dependencies are defined.

```
implementation 'com.google.firebase:firebase-database:16.1.0'
implementation 'com.google.firebase:firebase-core:16.0.8'
implementation 'com.google.firebase:firebase-auth:16.2.0'
implementation 'com.google.firebase:firebase-messaging:17.4.0'
implementation 'com.firebaseui:firebase-ui-database:4.3.1'
implementation 'com.google.firebase:firebase-storage:16.1.0'
```

Code Snippet 36 Firebase dependencies

4.4.3 SQLite DB Browser

OrderDetail		CREATE TABLE "OrderDetail" ('
UserPhone	TEXT	"UserPhone" TEXT NOT NULL
ProductId	TEXT	"ProductId" TEXT NOT NULL
RestID	TEXT	"RestID" TEXT NOT NULL
ProductName	TEXT	"ProductName" TEXT
Quantity	TEXT	"Quantity" TEXT
Price	TEXT	"Price" TEXT

Figure 29 DB Browser for SQLite

DB Browser [6] can perform relational database queries which can retrieve data using simple SQL queries including SELECT, UPDATE OR REPLACE and DELETE. Integrating this in the basket activity java class, each customer's cart items are stored temporarily.

SQLite DB Browser [6] is used to store information based on basket items. Each user can add menu items to their cart that is uniquely identified by user ID, product ID and restaurant ID. By storing this data, the application can recognize each users menu item based on a selected restaurant.

4.5 Flow Board of Overall System

The board below shows a series of activities presented by both the client and server application. This work flow is designed by Adobe XD [46], giving a visual representation of the sequence of actions users can operate with a touch of a button. A thorough video demonstration of both applications are available in a link provided in section 5.2.

4.5.1 Client Application

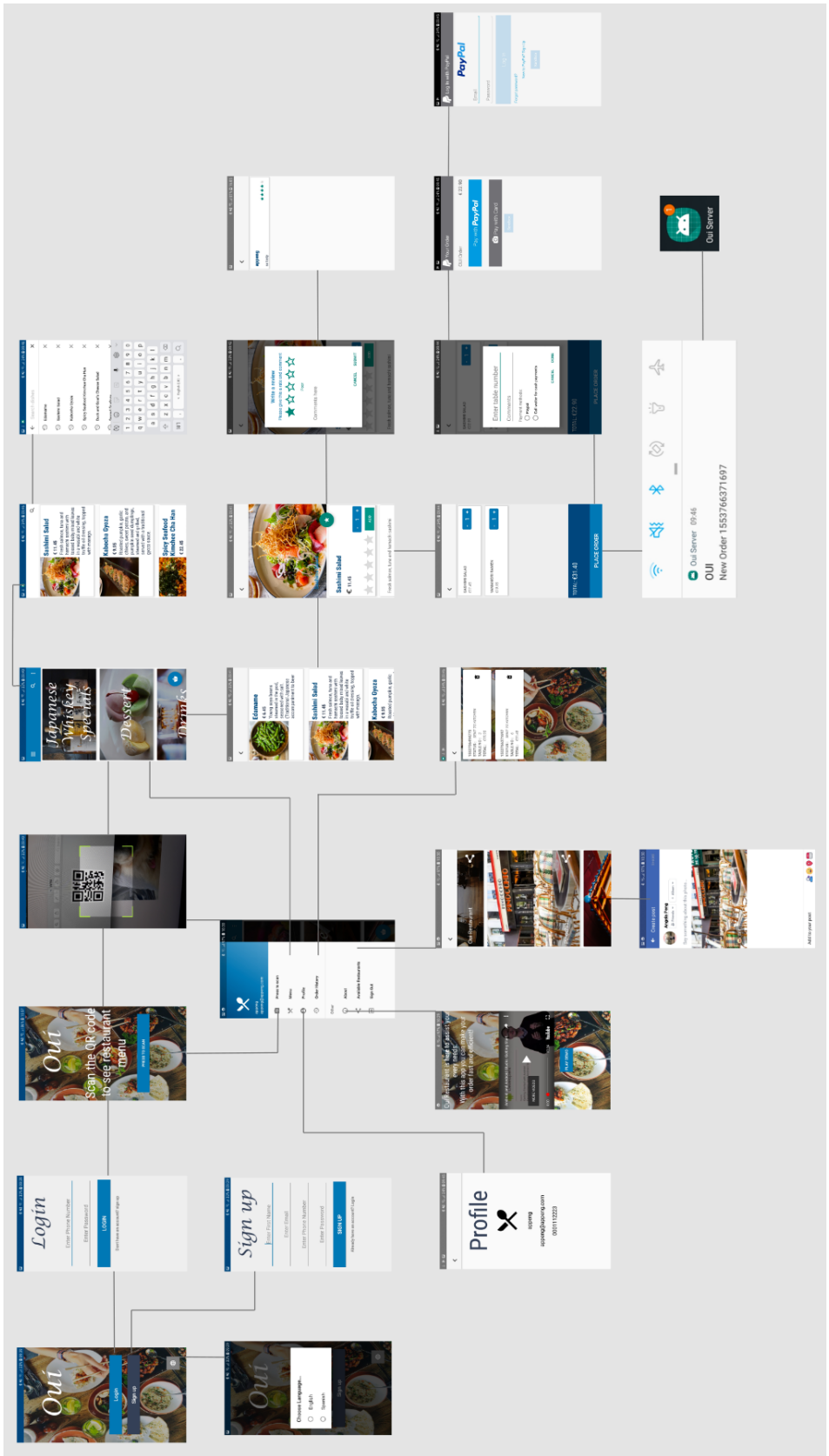

Figure 30 Flow Board of Client Application

4.5.2 Server Application

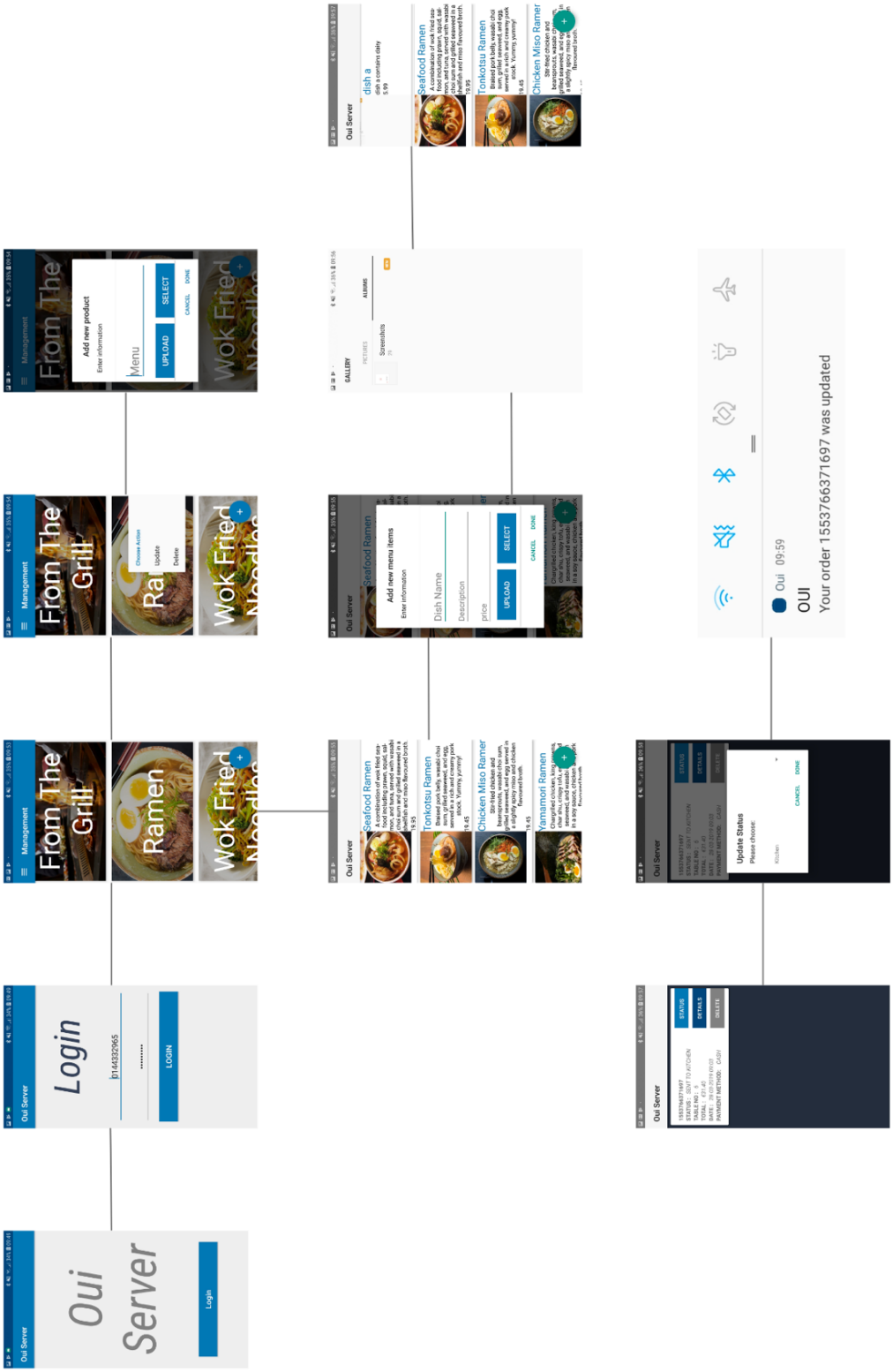

Figure 31 Flow Board of Server Application

5. System Validation

5.1 Testing

This section consists of different types of testing that were conducted during development and after the development phase. System testing includes think aloud protocol, usability and accessibility testing. Below illustrates these testing methods in detail.

5.1.1 Client Application Testing

Testing is conducted throughout the development of this application both front end and back end, to maintain a successful workflow. System testing that were performed on the client application included think aloud protocol, notification testing, usability and accessibility testing.

5.1.1.1 Think-Aloud Protocol

Think aloud (TA) protocol [55] also known as concurrent verbalization which is defined by Jääskeläinen:

"Subjects are to perform tasks and to verbalize whatever crosses their mind during the task performance"

The use of TA protocol concept matured from the concurrent protocol of translation study. By allowing the test subject to verbalise their thoughts during testing, we can gain a sense of the performance of the application in an unbiased aspect.

By following the think aloud protocol, a series of testing is conducted by fellow friends and classmates. Below illustrates the types of testing and validation of the client application performance:

Think aloud protocol was performed to test the final product which included friends and family as test subjects with both technical knowledge and users with little to no knowledge of technical aspects. For this system testing the TA protocol is divided into five sections which includes installation process, activity layout, application terminology, data entry and feedback. Two sample results are shown below, performed by test subject A with technical knowledge and test subject B with no

technical knowledge. The following tests are conducted using Samsung j6 mobile device and a Vodafone ultra 6 device.

Test Subject A *Subject possess technical knowledge.*

Subject is a regular IPhone user.

The test subject is fluent in both English and Spanish.

Date tested: April 2nd 2019

Installation	The subject showed appreciation to a quick and easy installation process. The subject noted that there was no complications during installation phase. Test subject noticed an error when scanning a QR code upon installation which was resolved when scanned again.
Layout	The subject noticed the layout of the application shows consistency and text size are legible. The participant also noted that the colour layouts are persistence with overall theme of the application. The subject demonstrated a quick knowledge of the application navigation system. Showing simple and straightforward sequence of actions.
Terminology	Participant noted that the terminology used in the application is straightforward and easy to understand, even for users that are not tech savvy. Subject tested the translation operation, which showed all texts excluding menu items are translated to a selected language. Language translated was considerably accurate.
Data Entry	The subject noted that number inputs and email inputs are defined by input type based on phone number and email address. The subject noticed that all data entries contains hints to assist users when entering data. The participant commented on the confusion when entering data while using an android device as this

	individual is more used to using an IOS device.
Feedback	This application provides an overall successful functionality. There is no lagging or delays retrieving data everything is very immediate, for example the QR code scanning takes no more than a matter of 2 seconds if even. The design of the user interface provided high contrast of reasonable choices of colours. The introduction to language provided an accessible feature and shows this app could go much further in future by adding more languages.

Test Subject B *Subject possess no technical knowledge.*

Subject is a regular android user.

The test subject is fluent in English

Date tested: April 2nd 2019

Installation	The subject had no reaction with the speed of installation. The subject had a successful installation process.
Layout	The subject noticed the layout was quite appealing and showed befitting imagery. The participant noticed the accessible language feature that is shown in the main activity The subject demonstrated a quick grasp of the application navigation by following activity instructions. The subject proved a good understanding of the application layout after running through the whole application. The subject noted that it the layout is straightforward and are able to navigate without any aid provided.
Terminology	Participant commented on the simple use of uncomplicated every day communication terminology. The subject was intrigued by the translation property which in result provided translated text. After the user was content with this operation they reset the language back to English.

Data Entry	The subject had no trouble entering data during place order activity and demonstrated smooth flow of actions when using this application. The participant entered data without any assistance and followed the entries hints.
Feedback	This application demonstrated a simple food ordering application that allows customers to enjoy their dining experience with the use of technology. The use of QR codes allows customers to retrieve a restaurant's menu without wasting time waiting for a server to perform these tasks.

5.1.1.2 Usability and Accessibility Testing

Usability and accessibility testing is conducted by initially testing the user interface with a colour blindness filter tool which was performed and discussed in section 3.3.1.3. This section evaluated the colour palette chosen for the main theme of this application, the result showed moderate changes when three types of colour blindness filters were tested.

The translation accessibility feature was tested both with a machine translator and human translator. The languages that were available to translate were tested by a machine, which in return provided inaccurate translation terminology because of this the translations were sent to a human translator. The human translator provided a more accurate and conversational translation of terms.

Finally, the last usability testing consists of the use of the talk-back feature. Android devices provide users with accessibility in-app setting which included talk-back functionality. When this setting is enabled users are able to be guided through the application with the aid of audio. Each activity and layout is vocally produced through the speaker provided by descriptive text function. The audio determines the type of text input, text view and button types which assists users with vision impairments.

5.1.1.3 Notification Testing

The notification testing is crucial to this project as it acts as an interactive communication between the client and server application. A demonstration of the

notification operation is available in a video lined in section 0. Notifications are performed by a simple button click by the customer once an order is placed which created a HTTP call and response connection to the server application.

The notification messages are sent by FCM [28], illustrated in section 4.2.9.1. This messaging can only be sent with the access of an internet connection. So, in order to send and receive notifications the mobile device must be connected to a network. This function is implemented in both the client and server application java classes.

5.1.2 Server Application Testing

As testing is performed constantly, the user interface testing is conducted in section 3.4.1.2 using adobe experience design. Further testing after the final development of the server application is conducted using the think aloud protocol. The test subject below is an individual who possess no technical knowledge and is using this application for the first time.

This test was performed using Vodafone ultra 6 mobile device.

5.1.2.1 Think-Aloud Protocol

Test Subject C *Subject possess no technical knowledge.*

Subject is a regular IPhone user.

The test subject is fluent in English

Date tested: April 2nd 2019

Installation	The subject demonstrated a smooth installation process with no complications or errors.
Layout	The participant noted that the application layout is similar to the client application but with less imagery and colours. The subject showed a clear understanding of the layout when using this application the first time. The subject noted that each order history is clearly laid out and can be accessed to view the type of dish and its quantity.
Terminology	The individual noted that the terminology used in this application is all very practical and efficient.

	Low use of long descriptive messages and texts provided a straightforward understanding of each operation.
Data Entry	Data entry was simple to input and select especially when the subject was adding, updating new menu items. The subject commented on the efficiency of adding new items and category of dishes into their restaurant menu. The individual noted that this allows them to operate their business even if they are away. The participant showed particular interest in how effortless this application can be when managing their restaurant.
Feedback	This application showed an efficient and sufficient understanding of the overall system including how it communicates to the customer. This application provides both customer and restaurant owner a sense of control. This can be a great concept to introduce to modern restaurants, providing a friendly and comfortable environment for customers which, as a result can increase the growth of revenue.

5.1.3 Manual Testing on Overall System

Manual testing was conducted to ensure a successful overall system functionality in both client and server application. In Table 11 Manual Testing Table, shows each action on both application executed and a PASS/FAIL result are declared to display each operation output.

5.2 Demonstration

A flow board is shown in section 4.5, the client and server application screen prints shows the sequential movement of the overall system. The source code is available on GitHub with the following link: https://github.com/C15402002/FYP. A README file contains the installation process of this application, provided with guidance.

A video demonstration is also available on YouTube with this link: https://youtu.be/Y6iutf3f_SE

Steps	Component	Description	Sequence of Actions	Expected Results	Status
1	Client App	Customer change language	Select appropriate language before accessing main activity.	Language set to default or changed.	**PASS**
2	Client App	Customer Sign Up	Go to sign up page. Enter details. Press sign up.	Customer have registered. Enters scan page.	**PASS**
3	Client App	Customer Login	Go to login page. Enter details. Press login in.	Customer have logged in. Enters scan page.	**PASS**
4	Client App	Scan restaurant QR	Customer press scan button.	Scanner decodes QR. Enters home page.	**PASS**
5	Client App	Add menu items to cart	Customer add items to basket on button press. Go through menu.	Item added to basket. Enters basket page.	**PASS**
6	Client App	Make order	Enter order details and quantity of each item. Pay order.	Order process success, send order to server app. Return home.	**PASS**
7	Client App	Manage order history	View order history. Delete orders if status is 0.	Enter order history page. Delete unprocessed orders.	**PASS**
8	Server App	Staff Login	Go to business login page. Enter details.	Staff have logged in. Enter home page.	**PASS**
9	Server App	Add menu items	Enter new dishes, name, price, description and image.	Enter details. Enters device gallery to select image. Upload complete.	**PASS**
10	Server App	View menu items	Enter home page to view category. Press each category, view each dish.	Menu items are displayed and categorized.	**PASS**
11	Server App	Delete menu items	Hold on each item to delete.	Item is deleted from database.	**PASS**
12	Server App	Modify menu items	Update each menu item on long press.	Item is updated and viewed in real-time.	**PASS**
13	Server App	Update order status	Go to order history. View orders. Change order status.	Order status is changed, notify client application.	**PASS**

Table 11 Manual Testing Table

6. Project Plan

The progression of this application from the initial design to its final product changed in many ways including user interface and functionality. From the interim stage to the current stage some areas have been altered to produce a more improved application that caters user's needs. The following objectives were set:

1. Ability to Scan QR code to retrieve a restaurant's menu
2. Ability to make in-App payments with PayPal
3. Implement a restaurant management application
4. Improve customers dining experience with the enhancement of technology
5. Ability to perform translation operations and visual accessibility features

From the objectives above all requirements are met to develop an accessible menu application for restaurants. Each objective implemented in this application determines its importance in the role of providing a flexible application. While each development process includes its own complexity, as a consequence the translation property and accessibility features including text enlarge, speech to text activities suffered significantly.

A major significant change of this project is the development of the server application. Initially this application was designed to support a web application using PHP and JavaScript as discussed in section 3.4.1. Using a native application can provide both portability and flexibility.

If this project was to be re-developed, improvement in accessibility features are essential. More research should be conducted to study different accessibility features and diverse ways to integrate it at a timely manner. A lot of the time spent on this project was to research on technologies, implementation methods and approaches which resulted in a lack of accessibility to users that are visually impaired. Although the current application has not reached its full potential, further development and research are conducted to enhance its many features.

7. Conclusion

7.1 Project Summary

To summarize this project, both the client and the server application provides an easy and efficient way to manage a restaurant business. This report shows the gradual development process of this digital menu application system from its initial stage to the current.

Not only does this project show the technical development of this project but an immense amount of research has been conducted to gain more knowledge to develop an application that is usable and is capable to cater customer needs. Research including analysing other author's journals, books and papers that may be of interest have all been conducted thoroughly to ensure that the final project is to be attainable for users of all age group and ethnicity.

It is evident that this project portrays its title, as a digital menu application for accessibility in restaurants. Through the progression of this report and development there are many aspects that are implemented to produce an accessible mobile application. Although the main functionality is smoothly implemented, there are still many areas to improve like translation optimization. An alternative method to maintain a fully functional translation application, Google Translation API must be implemented however by developing an user friendly accessible application a human translator is much preferred. This can take an extensible amount of time to implement as each menu item and its details must be translated one by one by a human translator, due to the time constraint this functionality became a downfall.

Despite a few drawbacks, there were still many areas of success within both applications. These areas included:

1. A successful implementation of quick response code decoder.
2. Has the ability to perform translation properties.
3. A successful implementation of processing payment transactions within the application.

4. A successful restaurant management application which interacts smoothly with the restaurant customers.

The project incorporated many challenging accessibility features that has yet to be implemented correctly. Throughout the development of this project a considerable amount of unfamiliar technical knowledge are studied and researched.

7.2 Future development

This application has shown a lot of potential for further development and capability to increase a more efficient and accessible application. Due to the time constraint when developing this project, there are many areas that this author would like to implement in future works to optimize this applications potential and performance. The following concepts are determined for future development.

7.2.1 Advanced Accessibility

Presently this application possesses accessibility features for visual impairment, colour blindness and minor translation properties. To enhance this accessibility aspect of the system, further research of efficient usability features can allow this application to grow in revenue and performance. Future research includes translating each menu items into a selected language by users. Research consists of different third party APIs, language and culture which can contribute to a fully translated menu application.

Another accessible feature that is for future development is to integrate machine learning into the client application. By learning customers likes and dislikes the application can produce products that cater to a specific customer. For example, promotion deals for returning customers based on their recent orders can provide restaurant owners a sense of customers experience and its popularity in the market. Integrating machine learning can provide a recommendation functionality to users either by their recent orders or searches which can increase revenue and profit both for the restaurant owner and the application's developer.

7.2.2 Ability to support other devices

Currently this application only supports android devices, but after conducting system

testing by test subject A 5.1.1, it was evident that the ability to support a wider range of devices is a definite possibility.

Future development will include the ability to support devices of all platforms which shows that this digital menu application can be accessible to customers with any mobile device. A particularly popular platform includes IOS, as SWIFT is still relatively new to the tech industry, there are many aspects and features to be researched and studied. Without any prior knowledge of Objective-C it can be reasonable to take the time to study this language and develop in the nearing future.

7.2.3 Analytic features

Currently the server application provides the restaurant owner functionalities includes add, update and delete menu items. Viewing incoming orders and changing each orders status. Despite these management functions, analytic features are still areas to be integrated in the server application.

Implementing analytic features to the server application including amount of active customers, orders growths and downfalls, dish ingredient shipping service management all play an enormous role in restaurant businesses. This can allow restaurant owners to manage their restaurant all in one through the server application. Analytic features can show managers monthly gross income or set targets of sales daily.

Bibliography

[1] D. A. Koutroumanis, “Technology’s effect on hotels and restaurants: Building a strategic competitive advantage.,” *Journal of Applied Business and Economics,* vol. 12, no. 1, pp. 72-80, 2011.

[2] Denso Wave, Corporate, “QR code,” [Online]. Available: http://www.qrcode.com/en/history/.

[3] ZXING, “ZXing 3.3.3 API. github,” Google Play Store, [Online]. Available: https://zxing.github.io/zxing/apidocs/.

[4] Square, “A type-safe HTTP client for Android and Java,” Retrofit, Square, 2 April 2019. [Online]. Available: https://square.github.io/retrofit/. [Accessed 2 April 2019].

[5] Google, “Firebase by platform,” Firebase, 1 September 2011. [Online]. Available: https://firebase.google.com/docs/?authuser=0. [Accessed 2 April 2019].

[6] D. Browser, “DB Browser for SQLite,” DB Browser , 7 July 2014. [Online]. Available: https://sqlitebrowser.org/. [Accessed 2 April 2019].

[7] B. Wong, “Points of View, Color blindness,” *Nature methods,* vol. 8, no. 6, p. 441, 2010 .

[8] L. Troiano, C. Birtolo and R. Armenise, A validation study regarding a generative approach in choosing appropriate colors for impaired users., Benevento: SpringerPlus, 2016.

[9] L. Hanks, N. D. Line and A. S. Mattila, “The impact of self-service technology and the presence of others on cause-related marketing programs in restaurants.,” *Journal of Hospitality Marketing & Management,* vol. 25, no. 5, pp. 547-562, 2016.

[10] P. Cullen, “Time, tastes and technology: the economic evolution of eating out.,” *British Food Journal,* Vols. 96,, no. 10, pp. 4-9, 1994.

[11] Z. Yang and R. T. Peterson, “Customer perceived value, satisfaction, and loyalty: The role of switching costs.,” *Psychology & Marketing,* vol. 21, no. 10, p. 799–822, 2004.

[12] J. Wirtz, B. M. Noone and S. E. Kimes, “The Effect of Perceived Control on Consumer Responses to Service Encounter Pace: A Revenue Management Perspective,” SAGE, Ithaca, 2012.

[13] C. I. Ltd, “Customer Experience Lifecycle Audit,” Customer Input, 1 January 2011. [Online]. Available: http://www.customerinput.com/customerexperience/experience-lifecycle-audit/. [Accessed 1 March 2019].

[14] Rangelov, “OrderWizard,” rangelov@orderwizard.mobi, 19 September 2013. [Online]. Available: https://play.google.com/store/apps/details?id=mobi.orderwizard&hl=en_US.

[15] Q. Orders, “Quick Orders your personal waiter,” Quick Order Inc., 26 May 2015. [Online]. Available: http://quickorders.net/#features. [Accessed 29 March 2019].

[16] eHopper, “eHopper POS Feature,” [Online]. Available: https://ehopper.com/product-features/.

[17] M. H. Goadrich and M. P. Rogers, “Smart smartphone development: iOS versus Android.,” *In Proceedings of the 42nd ACM technical symposium on Computer science education,* vol. 42, no. 1, pp. 607-612, 2011.

[18] P. Pearce, A. Felt, G. Nunez and D. Wagner, “Addroid: Privilege separation for applications and advertisers in android,” *In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security,* vol. 7, no. 1, pp. 71-72, 2012.

[19] M. Rouse, “mobile device fragmentation.,” [Online]. Available: https://searchmobilecomputing.techtarget.com/definition/mobile-device-fragmentation.

[20] C. Maia, L. M. Nogueira and L. M. Pinho, “Evaluating android os for embedded real-time systems,” *In 6th international workshop on operating systems platforms for embedded real-time application,* vol. 6, no. 1, pp. 63-70, 2010.

[21] M. H. Goadrich and M. P. Rogers, “Smart smartphone development: iOS versus Android,” *In Proceedings of the 42nd ACM technical symposium on Computer science education,* vol. 42, pp. 607-612, 2011.

[22] G. Wells, “The Future of iOS Development: Evaluating the Swift Programming Language,” CMC Senior Theses, Claremont .

[23] W. Jobe, “Native Apps vs. Mobile Web Apps,” International Journal of Interactive Mobile Technologies, Stockholm, 2013.

[24] R. Meier, Android Application Development, Perth: Framework 318, 2012.

[25] E. Almrot and S. Andersson, “A study of the advantages & disadvantages of mobile cloud computing versus native environment,” Blekinge Institute of Technology , Karlskrona, 2013.

[26] E. Ciurana, “Developing with google app engine,” *Mobile Web Apps,* vol. 30, no. 5, pp. 22 - 27, 2013.

[27] A. Khandeparkar, R. Gupta and B. Sindhya, “An introduction to hybrid platform mobile application development,” *International Journal of Computer Applications,* vol. 118, no. 15, pp. 31-33, 2015.

[28] Google, “Firebase Cloud Messaging,” Google, Firebase, 29 March 2019. [Online]. Available: https://firebase.google.com/docs/cloud-messaging/. [Accessed 2 April 2019].

[29] Google, “ Firebase Authentication,” Google, Firebase, 27 March 2019. [Online]. Available: https://firebase.google.com/docs/auth/. [Accessed 2 April 2019].

[30] Google, “Google Play services,” Google LLC, 29 March 2019. [Online]. Available: https://www.apkmirror.com/apk/google-inc/google-play-services/#variants. [Accessed 2 April 2019].

[31] M. Wolfson, “Converting your Android App to Jetpack,” A Medium Corporation, 27 November 2018. [Online]. Available: https://medium.com/google-developer-experts/converting-your-android-app-to-jetpack-85aecfce34d3. [Accessed 29 March 2019].

[32] OpenJaw, “ The Rise of QR Code in China and its Effect on the Travel Industry,” 10 April 2017. [Online]. Available: http://www.openjawtech.com/qr-code-travel-industry/.

[33] Wechat, Tencent, “Wechat,” [Online]. Available: https://www.wechat.com/en/.

[34] A. Payne and P. Frow, “A strategic framework for customer relationship management,” *Journal of marketing,* vol. 69, no. 4, pp. 167-176, 2005.

[35] T. Bohling, D. Bowman, S. LaValle, V. Mittal, D. Narayandas, G. Ramani and R. Varadarajan, “CRM Implementation: Effectiveness Issues and Insights,” SAGE, Maryland, 2006.

[36] A. Bhargave, N. Jadhav, A. Joshi, P. Oke and S. Lahane, “Digital ordering system for restaurant using Android.,” *International journal of scientific and research publications,* vol. 3, no. 4, pp. 1-7, 2013.

[37] A. De Angeli and L. Kyriakoullis, “Globalisation vs. localisation in e-commerce: cultural-aware interaction design,” *In Proceedings of the working conference on Advanced visual interfaces,* vol. 1, no. 1, pp. 250-253, 2006.

[38] J. Hutchins, “Machine Translation and Human Translation: In Competition or in Complementation?,” *International Journal of Translation,,* vol. 13, no. 1, pp. 6-19, 2001.

[39] PayPal, “PayPal Developer,” [Online]. Available: https://developer.paypal.com/.

[40] A. Rodrigues, K. Montague, H. Nicolau and T. Guerreiro, “Getting smartphones to TalkBack: understanding the smartphone adoption process of blind users.,” *In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility,* vol. 17, no. 1, pp. 23-32, 2015.

[41] R. I. Duke, H. Nicolau and V. L. Hanson, “mBrailler: Multimodal Braille Keyboard for Android,” Rochester Institute of Technology, Rochester, 2016.

[42] J. A. Livermore, “Factors that Significantly Impact the Implementation of an Agile Software Development Methodology.,” *JSW,* vol. 3, no. 4, pp. 31-36, 2008.

[43] D. Hughey, “The Traditional Waterfall Approach,” 2009. [Online]. Available: http://www.umsl.edu/~hugheyd/is6840/waterfall.html.

[44] J. Highsmith and M. Fowler, “The Agile Manifesto,” Software Development, Utah, 2001.

[45] J. GULLIKSEN, B. GORANSSON, I. BOIVIE, S. BLOMKVIST, J. PERSSON and A. CAJANDER, “Key principles for user-centred systems design,” *BEHAVIOUR & INFORMATION TECHNOLOGY,* vol. 22, no. 6, p. 397–409, 2003.

[46] A. E. Design, “Adobe XD,” Adobe Creative, 1 October 2015. [Online]. Available: https://www.adobe.com/ie/products/xd.html. [Accessed 2 April 2019].

[47] Colblindor, “Coblis — Color Blindness Simulator,” Colblindor, 27 October 2016. [Online]. Available: https://www.color-blindness.com/coblis-color-blindness-simulator/. [Accessed 2 April 2019].

[48] SAP, “Model View Controller (MVC),” SAP, 1 January 2019. [Online]. Available: https://help.sap.com/doc/saphelp_uiaddon20/2.05/en-US/12/1b8e6337d147af9819129e428f1f75/content.htm?no_cache=true. [Accessed 2 April 2019].

[49] Facebook, “Facebook for Developers Documentation,” Facebook, [Online]. Available: https://developers.facebook.com/docs/. [Accessed 10 April 2019].

[50] Youtube, “Developer Documentation,” Google, [Online]. Available: https://developers.google.com/youtube/documentation/. [Accessed 10 April 2019].

[51] A. Masny, “Paper DB,” Paper, [Online]. Available: https://github.com/pilgr/Paper. [Accessed 10 April 2019].

[52] S. Gunasekera, “Create apps that are safe from hacking, attacks, and security breaches,” in *Android apps security*, Apress, 2012, pp. 67-72.

[53] P. P.-S. Chen, “The entity-relationship model—toward a unified view of data,” *ACM Transactions on Database Systems (TODS),* vol. 1, no. 1, pp. 9-36, 1976.

[54] B. Stonehem, Google Android Firebase: Learning the Basics., usa: First Rank Publishing, 2016.

[55] R. Jääskeläinen, “Think-aloud protocol.,” in *Handbook of translation studies 1*, John Benjamins B.V., 2010, pp. 371-374.