

Smart Restaurant

T.H.T.D. THRIMANNA
2021



Smart Restaurant

A thesis submitted for the Degree of Master of
Information Technology

T.H.T.D. Thrimanna

University of Colombo School of Computing

2021



Declaration

The dissertation is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: T.H.T.D.Thrimanna

Registration Number: 2018/MIT/079

Index Number: 18550794

Signature:

Date: 22/07/2021

This is to certify that this thesis is based on the work of Mr./Ms. under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: W.V Welgama

Signature:

Date: 22/07/2021

Abstract

Nowadays people are more inclined to use mobile technologies for their systems. Smart Restaurant is a system of integrating the restaurant management system through mobile technology. It works to reduce manual process and increases the accuracy of process in a restaurant. This complete system will provide a pleasant experience for the customer and the restaurant staff. It is increasingly attracting customers and helping to transform the restaurant to grow with a larger customer base.

This consists of four main parts. Android Tab Application, Android Mobile Application, Android TV Application and Android Admin Portal. Smart Restaurant system manages and maintains the orders of customers. Customers can use mobile application and Tab application to directly connect with the reservations and order process. The TV app and Admin portal will make the day-to-day work of restaurant staff easier.

This system completely reduces the unnecessary time spent by customers and restaurant staff. And also, the system is fully functional under a real-time process.

Each order is linked to a single table and is built by one customer per order, but with greater accuracy. Food items can be easily searched, viewed, selected and the bill status can be viewed by customers in real time. Order status and order details are being updated in real time for staff members.

Acknowledgement

I would like to thank my supervisor Mr W.V Welgama from UCSC to his endless support throughout the project. His guidance in narrowing down the thesis project scope, frequent follow up and comments as well as encouragements have an irreplaceable role for the successful completion of the work. I would also like to extend my sincere thanks to entire UCSC lecture panel to giving me this opportunity.

Finally, I am indebted to my beloved parents, brothers, sisters and dearest friends and I would like to express my profound gratitude to them.

Table of Contents

Declaration	II
Abstract	III
Acknowledgement	IV
List of Figures	VIII
List of Tables	XI
List of Acronyms	XII
Chapter 1 Introduction	1
1.1 Project Overview	1
1.2 Motivation	1
1.3 Objectives	2
1.4 Background of study	3
1.5 Scope of the study	4
1.6 Feasibility Study	5
Technical Feasibility	5
Economic Feasibility	5
Operational Feasibility	5
1.7 Structure of the dissertation.....	6
Chapter 2 Background	7
2.1 Introduction	7
2.2 Requirement Analysis	7
Functional requirements	7
Non - Functional requirements	8
2.3 Review of Similar Systems	11
Cash Registry.....	11
POS System	11
Food Delivery	12
2.4 Related Technologies	12
Tech Stack	13
2.5 Related Design Strategies.....	13

Chapter 3 – Design.....	16
3.1 Introduction	16
3.2 System Architecture	16
3.3 Software Architecture.....	17
MVVM Architecture	17
Base Components	18
Cloud Firestore Architecture	20
3.5 UML Diagrams.....	21
Use case diagram	21
Class Diagram	23
Activity Diagram	24
Chapter 4 Implementation.....	26
4.1 Introduction	26
4.2 Implementation environment	26
Used software and tools.....	26
Architecture Components.....	26
UI Design.....	32
Utils	35
Chapter 5 Testing and Evaluation	36
5.1 Introduction	36
5.2 Testing Types	36
Unit Testing	36
Instrumented Testing.....	37
Useability Testing.....	37
5.3 Test Cases.....	37
Customer Tablet Application	37
Admin Tablet Application.....	41
5.4 Functional Evaluation.....	43
User Evaluation	43
Firebase Tools	43
Chapter 6 Conclusion.....	48
7.1 Introduction	48
7.2 Project Limitations	48

7.3 Difficulties encountered	48
7.4 Benefits of Implementing This Project	48
7.5 Potential future work	49
References	50
Appendicix A	52
Appendicix B	54
User Manual for Table Tablet Applications	54
User Manual for Table Admin Applications	60
User Manual for Kitchen Applications	64
Appendicix C	65

List of Figures

Figure 2.1 Reservation through the mobile application	9
Figure 2.2 Order Placement through the tablet application	9
Figure 2.3 Main Flow	10
Figure 2.4 Waterfall Method.....	15
Figure 2.5 Iterative model.....	15
Figure 3.1 System Architecture	16
Figure 3.2 MVVM Architecture	17
Figure 3.3 Activity lifecycle	18
Figure 3.4 Fragment Overview	19
Figure 3.5 Data Organization in Firestore	20
Figure 3.6 Smart Restaurant use case diagram	22
Figure 3.7 Smart Restaurant class diagram.....	23
Figure 3.8 Smart Restaurant dining order placement Activity Diagram	24
Figure 3.9 Smart Restaurant mobile reservation Activity Diagram	25
Figure 5.1 Crash-free statistics	44
Figure 5.2 App Trends	44
Figure 5.3 Crashes	45
Figure 5.4 Crash effected devices	45
Figure 5.5 Screen view	46
Figure 5.6 User Engagement of Screens	46
Figure 5.7 Functional Evaluation.....	47

Figure B.1 Table Register ScreenFigure.....	54
Figure B.2 Login Screen	54
Figure B.3 Customer Registration	55
Figure B.4 Menu	55
Figure B.5 Select menu items	56
Figure B.6 Cart.....	56
Figure B. 7 Pending Status.....	57
Figure B.8 Processing Status	57
Figure B.9 Served Status.....	58
Figure B.10 Confirm the Termination	58
Figure B.11 Bill Settlement Pending Status	59
Figure B.12 Bill Settled Status.....	59
Figure B.13 User Login	60
Figure B.14 Order Status	60
Figure B 15 Food Details	61
Figure B.16 Add Food item	61
Figure B.17 Admin Panel	62
Figure B.18 Category Details	62
Figure B.19 User Details.....	63
Figure B.20 Table Details	63
Figure B.21 Pending and Processing orders	64
Figure B.22 Accept an order	64

Figure C.1 Summary Report	65
Figure C.2 Sales Report	65
Figure C.3 Performance Report	66
Figure C.4 Category Report	66

List of Tables

Table 1.1 System functionalities.....	4
Table 2.1 Phase wise description of the waterfall model.....	14
Table 5.1 User tablet application test cases	41
Table 5.2 Admin application test cases.....	43

List of Acronyms

IDE Integrated Development Environment

MVVM Model View ViewModel

OS Operating System.

POS Point of sale

SDK Software Development Kit

UML Unified Modeling Language

UI User Interface

Chapter 1 Introduction

1.1 Project Overview

In today's fast-paced world, when everyone is squeezing in on time. So people always try to use the easiest way to complete their tasks without taking much time and energy. Food is a basic human need. So, the food industry has a good market all over the world. There are restaurants, shops and many more ways to get food. Somehow today people are busy with their own work.

They don't have much time to waste on simple things. Because of that people love to have their food at the restaurants.

Saving time is not the only reason restaurants are popular. Most people behave subtly when placing a food order. Dining at restaurants is supposed to be one of the most loved activities worldwide. For many reasons people like to go to restaurants and spend their time. To celebrate life events, formal and informal events, avoid cooking, the reputation of specialized chefs, and gaining a variety of experiences are the few of them.

Because of that, the rush of restaurants can be increased anytime. Then the customers who come to dine may have to face a lot of trouble at restaurants. SMART RESTAURANT is a complete system designed primarily for use in the restaurant industry. This system allows to increase the scope of the business by reducing the labor costs associated with hotels and restaurants. The system also allows users to quickly and easily manage a menu which customers can browse and use to place orders with just a few clicks. Restaurant employees are able to process these orders through the system. It can be avoided by the time wastage of customers and staff members.

Using this type of system will not only save your money and time, it will also be an environmental point of view advantage. Mainly reducing paper usage is going to be a massive plus point to the world.

1.2 Motivation

The motivation for planning this system is largely because I have faced this kind of situation a lot and according to my personal experiences, I do not like to waiting for a long in the restaurants or to give a call to the store to place an order, especially during the peak lunch or

dinner hours. This problem currently exists in many restaurants in Sri Lanka. Many of them have trouble managing the peak hours. Many restaurant staff have been reduced due to the uncertainty caused by the covid situation. Labor costs are also high these days. To run the highest profitable business in this kind of domain management should try to run the restaurant with the minimum number of staff. For these reasons the automated restaurant system will be more affordable for restaurants.

1.3 Objectives

Objective of this project is to give a smart solution to the restaurant process.

- Minimize human interaction with the restaurant process.
 - Check the availability of the tables
 - Reserve tables.
 - Order placement.
 - Bill request.
 - Order passing to kitchen.
 - Updating current order status.
 - Billing processes

are key features of the restaurant process and all the components will be automated.

- Check the availability of the tables, Reserve tables and order placement that are going to be automated to be done remotely. Customers can also place orders once they visit the restaurant.
- Avoid paper bills and give a digitization bill to Table and send a copy of the bill to the customer's email.
- Avoid sending orders to the kitchen via written paper notes, as placed orders will appear on the TV screen in the kitchen.
- Avoid paper menus, where customers can see a digitized menu with pictures of the foods they are going to choose.
- Minimize the time wastage at the restaurant.
 - Avoid the customer waiting time for the waiters.
 - Avoid waiting for the bill to come
 - Avoid waiting for waiters until the customer places an order.
- Giving facilities to customers to get real time information.

- Updated bill.
- Menu information.
- Order status.

will be informed to the customer in real time.

- Increase the quality of the service. Real time, efficient and automated systems can increase the attraction of the customers and also it can improve the accuracy of the service.

1.4 Background of study

Many people in urban areas try to get pre prepared meals rather than cooking. That's why the rush of restaurants has increased. As a solution to this, food delivery systems have been introduced by several companies. But the majority of the public is looking for fresh food. And also, for special occasions people trying to have the dining experience at the restaurants.

Online ordering systems help people to order the food for where they are. But it will not resolve the issues with the restaurant's dining. Even the online table booking system can cause the below hassles if the restaurant runs manually.

- Tables may not be available when entering the restaurant.
- It Takes a long time to contact a waiter with them.
- When they are given only a printed menu, it is difficult to get an idea about the foods by seeing only the name of the food.
- They may need staff assistance to find out more details about a food.
- Cannot get a clear idea about the final amount until the bill recive to the table.
- Could not find a way to re-order the same order which was previously ordered.

And also, there are number of annoying troubles that restaurant staff may have to face,

- Waiters have to wait until the customer selects their food items from the menu.
- Customers call the waiters several times when they need something.
- Cashier has to enter all the bill items one by one.

Hence, to resolve these issues, what I propose SMART RESTAURANT system. The main advantages of this system is, it can be greatly simplify the ordering process for the customer and the restaurant, and since the entire process of ordering is automated.

1.5 Scope of the study

The proposed system will be developed to manage customer interaction and restaurant staff interaction in the restaurant process. The main structure of the system is divided into 4 main components.

1. Android Tab Application.
2. Android Mobile Application.
3. Android TV Application.
4. Admin Portal.

Component	Description	Features
Android Tablet Application	The restaurant has an Android tablet which is attached to every table and the app installed on the tablet can be accessed by any customer who goes to dining at that table.	<ul style="list-style-type: none">● Login/Register using mobile number.● Promotion/Notices.● Check the menu and view details of the items.● Order placement.● Check the history of the orders and reorder.● Check the bill, Request the bill and select the payment method.● Rate the service.
Android Mobile Application	The mobile app is a public Android app that can be accessed by any customer.	<ul style="list-style-type: none">● Reserve a table for a specific time slot.● Place an order for the reserved table.
Android TV Application	This is the application that is going to be installed on the TV installed in the kitchen.	<ul style="list-style-type: none">● Real-time update the pending orders.● Change the status of the order.
Android admin portal	Cashiers and restaurant managers can access the portal.	<ul style="list-style-type: none">● Order Process.● Bill process.● Check the status of the restaurant tables.● Reports.

TABLE 1.1 SYSTEM FUNCTIONALITIES

Referring to Table 1.1, Each part of the system has described including their usage and functionalities.

1.6 Feasibility Study

Technical Feasibility

User interfaces of Admin Portal, Mobile Application, Tab Application and TV Application will be deployed in a user-friendly manner. Therefore, staff members and the customers can handle the applications easily.

Usage of the mobile devices are getting increase day by day due to the portable features, effective and attractive applications. Mobile Application, Tab Application and TV Application will be developed in Android platform. Mobile Application is going to be a public application and Tab and TV Applications are going to be a part of the internal system. In Sri Lanka majority of the people are using Android phones. Therefore, it may not need much technical knowledge to use the Mobile/Tab/TV applications.

Economic Feasibility

This means the benefits we get from the product relative to the total cost It would be cost effective to use Android as an open-source platform. We can easily find Tab and TV Android devices for a reasonable cost.

Using this system can reduce the labor cost. And also, easily can monitor the sales and run a quick server to find out what food is selling highly and what food is not selling well.

Therefore, by increasing the efficiency of the service can be developed the business of the restaurant.

Operational Feasibility

When it comes to using the system, it should be usable by anyone, restaurant customers can be anyone. There can be any level of education. So, the system will be suitable for any person and at any age.

When it comes to staff members, they may need a little training. Specially to handle the TV application in the kitchen and to handle the admin portal at the cashier.

1.7 Structure of the dissertation

- Chapter 1. Introduction

In this chapter gives a brief introduction about the background & motivation of the proposed system. Even it described the aim & objectives of the project.

- Chapter 2. Background

This Chapter describes a review of the work related to the project, the analysis of the requirements updates and a comprehensive review of the relevant literature about the similar systems and their technologies.

- Chapter 3. Design

The idea about design of the system is given in this chapter. After reading this chapter the structure of the system will be clear to the reader.

- Chapter 4. Evaluation

This chapter gives an evaluation of the system. It will be discussed whether the project objectives were satisfied or not. If not reasons for them.

- Chapter 5. Conclusion

This chapter includes the work indicating a summary of the results of the project.

Chapter 2 Background

2.1 Introduction

In this chapter we discuss the traditional methods, the difference between the proposed project and the similar systems that exist.

Today the internet is widely used everywhere. The Internet is very convenient for people because almost everything can be done online. And everyone is using a smartphone these days. Because of that technology, it would be more useful to provide a mobile based restaurant system.

In the early days many restaurants used the manual methodology to serve customers. For the billing part also, they were using written bill methods. Customers have a lot of trouble because of the manual method.

With the advancement of technology, a number of new approaches are being introduced to the food industry. Various techniques have been used to overcome the difficulties faced by the customers during the service period. Not only the food industry but many industries have improved their services by acknowledging the intervention of technology.

2.2 Requirement Analysis

Functional requirements

⇒ The following are functional requirements for **Tablet Application**.

- In the Tablet application it should be able to register for a table.
- Customer should be registered using their mobile number.
- This must grant the user validity. (Customer validation). You can test whether the customer is in the system or available from a new customer.
- If there is a promotion or notification, it should show up when the user logs into the application.
- From the tablet application users can go through the menu and see the details of the food items.
- The user should be able to select items, add them to the cart and place an order.
- The user should be able to check his/her old orders and reorder them.
- The user must be able to get the current size of the bill, requesting termination of the bill select the payment method.

- The user should be able to rate the service.
- ⇒ The following are functional requirements for **Mobile Application**.
- Mobile Application users can access the mobile application anywhere.
 - The uses can be able to enter the arriving time.
 - They can be able to select the number of seats.
 - The user can be able to place an order.
- ⇒ The following are functional requirements for **TV Application**.
- Kitchen staff should be able to learn about new orders.
 - Kitchen staff should be able to change the status of the orders.
 - Kitchen staff can start to process the order.
- ⇒ The following are functional requirements for **Admin portal**.
- Admin users can manage Tables, Categories, Menu, Staff, Offers and Reports.
 - Cashier can see the orders current status in real time.
 - Cashier should be able to process the bill. After settling the bill table should be changed to the available status.

Non - Functional requirements

- ⇒ Performance - The application is able to retrieve real data in real time.
- ⇒ Availability - The service should be available 24 * 7 as the customer can reserve a table at any time.
- ⇒ Security - Security of the Customer details should be provided.
- ⇒ Setup/ installation - Public Mobile Application should be able to deploy to Google play store. The tablet application and TV application for internal use and must have a way to install them easily. The apks of the internal applications can be fetch from a public URL.

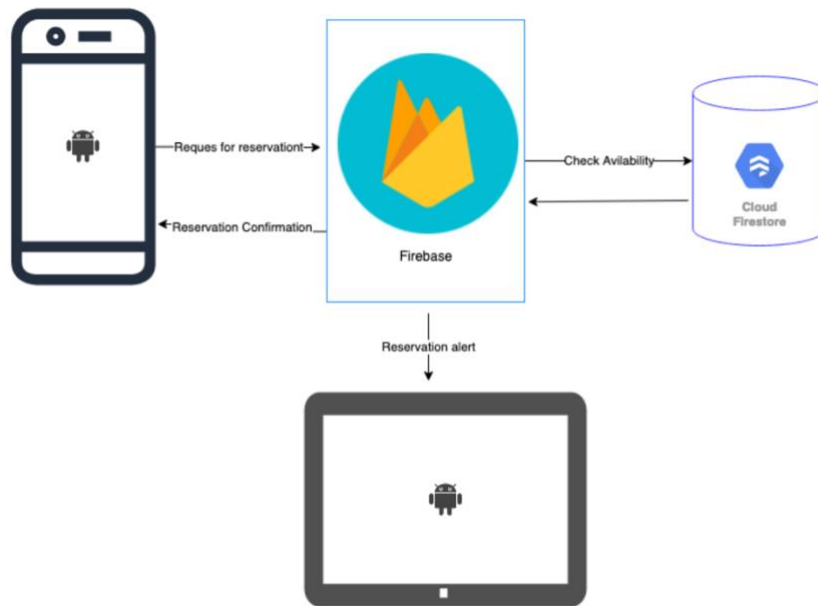


FIGURE 2.1 RESERVATION THROUGH THE MOBILE APPLICATION

Referring to Figure 2.1, Mobile sends the request with reservation details to the firebase, and it checks the table availability in the Firestore database according to the details. In a success request, it stores the reservation record in Cloud Firestore database.

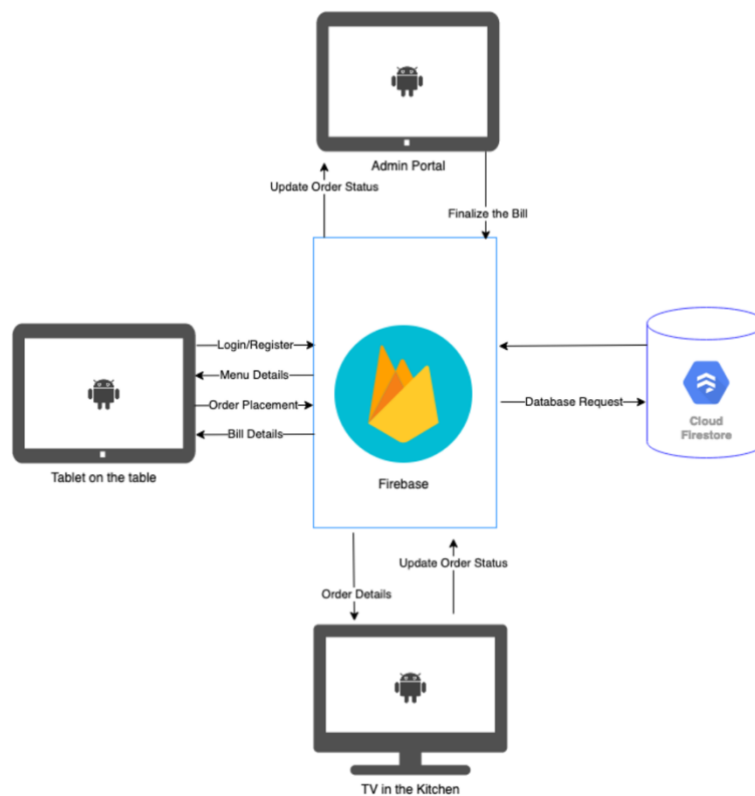


FIGURE 2.2 ORDER PLACEMENT THROUGH THE TABLET APPLICATION

An overview of the order placement through the tablet application depicted in Figure 2.2. According to that server all the components of the system have connected to the server.

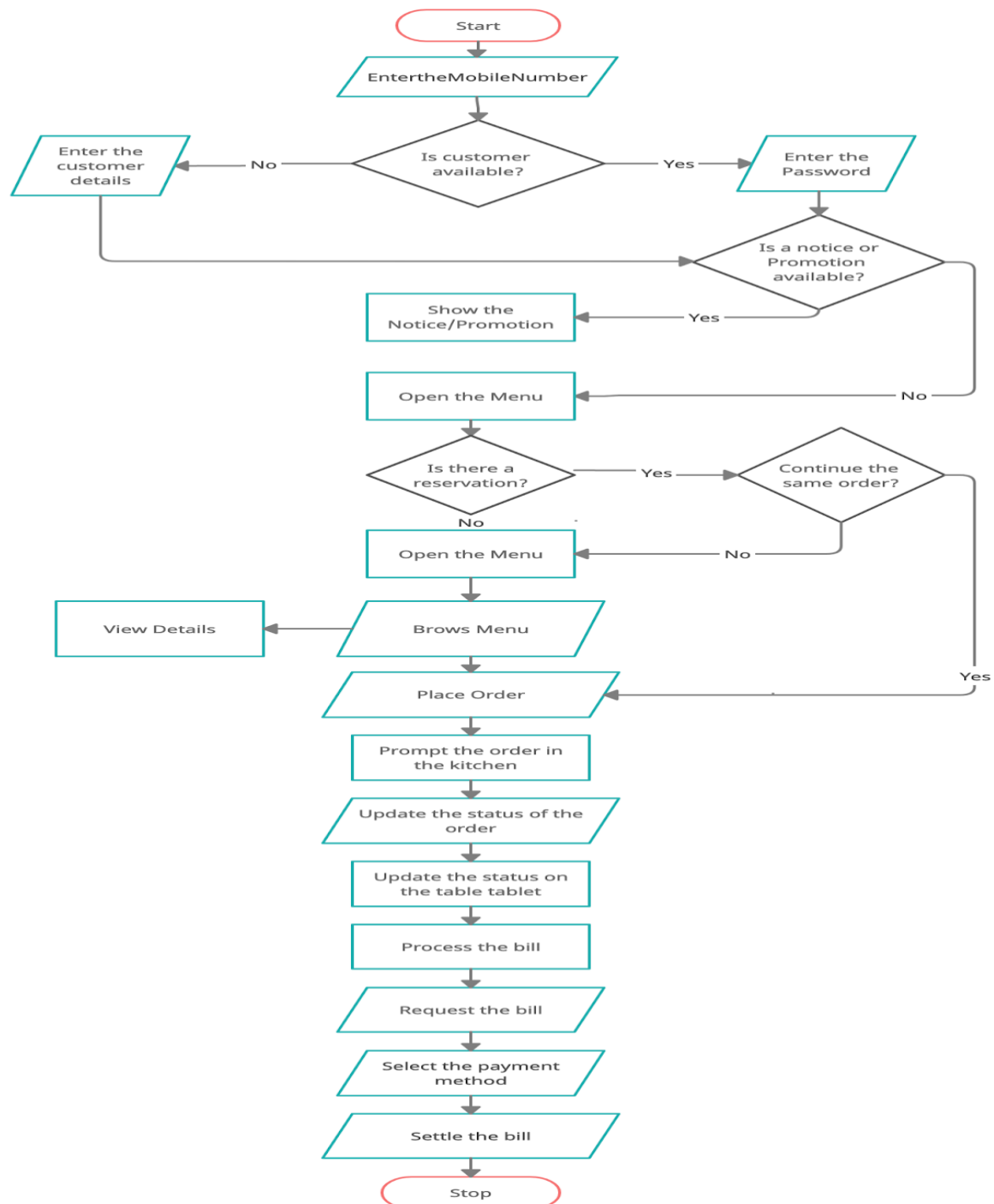


FIGURE 2.3 MAIN FLOW

Referring to Figure 2.3, The entire flow of the system is shown

2.3 Review of Similar Systems

Cash Registry

Restaurants and other outlets began to use cash registers to avoid difficulties with the written bill system. This method is faster and accurate than the handwritten system.

A restaurant cash register is a mechanical or electronic device for registering and calculating transactions. It has a physical drawer for storing cash. Nowadays modern electronic restaurant cash registers are usually attached to a printer that can print out receipts for record-keeping purposes. When a transaction is to be recorded, the operator punches in the bill amount and proceeds to record the sale. The cash drawer opens only when a sale is recorded. The operator collects the cash and stores it in the drawer, in something called the till. Once the operator closes the drawer, the transaction is complete.

POS System

A POS system can do everything that the restaurant cash register does, and much more. It has improved features such as advanced data management, integrations, and the freedom to install them on a variety of devices and form factors. A POS system is central to the technology requirements of the modern restaurant. And restaurant cash registers just don't cut it anymore. (Datta.S, 2021)

Odoo Point of Sale and restaurant management system. (Odoo, 2021)

Odoo Restaurant allows to have an overview of your restaurant quickly, move from one floor to another and visualize the orders of each table. Every component must be purchased by the client as a plugin.

- Available Features in the system
- Restaurant Features
- Manage tables
- Offer a bill splitting option
- Bill Printing
- Orders can be print at the kitchen or bar
- Include a tip option into payment
- Pricing Features
- Apply Discounts

- Consider the discount tags with a barcode scanner
- Manage a loyalty program
- Get the Pricelists in Point of Sale.

Food Delivery

Nowadays food delivery is famous. Some people like to order food and get it where they are.

In Sri Lanka these days Uber and Pick me are the most famous food delivery applications. Uber Eats and Pick me food changed the ordinary food ordering process to a smart food ordering process over a mobile application. (Uber, 2021)

But the majority of the public is looking for fresh food. So they like to go to a restaurant and have a dining experience. The following section illustrates a brief review of an existing system which has similar features.

Foodie365cloud

The Foodie365cloud is a restaurant software for fine dining restaurants, take away and quick service restaurants, bar, clubs and disco. Serve more customers and keep them happy too with Foodie365cloud most innovative single order on multiple KOT(Kitchen order taking) and BOT(Bar Order Taking) print channel functionality. With visual layout of floor plan, the manager can check and update the current status of every table and other activities performed in house. Easily manage weekend rush using powerful Reservation system. (foodie365cloud, 2021)

- Available features.
- Manage guest seating.
- Customer offers and loyalty management.
- Menu management.
- Kitchen management.
- Bar management.

2.4 Related Technologies

The proposed system is highly based on mobile technology. And all the mobile parts are going to be completed with the Android platform. All the scenarios are updating in real time.

Tech Stack

Proposed Mobile Application/ Tab Application and TV Application are developing using Native Android and following technologies are using there. Mobile, Tablet and TV Solutions will be applied to Model, View, ViewModel (MVVM) architecture.

- Kotlin / Java
- Firebase
- Cloude Firestore
- Firebase Storage
- Firebase Performance Indicator
- Android gitpack
- Crashlytics
- Room Database
- Material Design
- Espresso
- Junit
- Robolectric
- Google Play Services

2.5 Related Design Strategies

Iterative Waterfall software development life cycle will be applied. Iterative Waterfall methodology is a linear project management process, which the stakeholder and customer requirements are gathered at the beginning of the project. Feedback pathways allow these to be corrected at a later stage as well as at a later stage. According to the Iterative waterfall methodology, the following steps in sequence will be the procedure in Table 2.1.

Sequential phase	Description
Feasibility Study	Understanding the problem and then determine the various possible strategies to solve the problem

Requirements Analysis and Specification	Gather comprehensive information about the project. End of this phase the project requirements will be finalized and clear.
Design.	System design provides some help in specifying hardware, clear the system requirements and defining the overall system architecture. And also end of this phase all the UI/UX designs will be finalized.
Implementation and Unit testing.	Coding and Implementation takes place in this phase. Get the clear requirements from the previous phase and create the complete functional product. Implement the code in small pieces, which are integrated at the end of this phase. Same time creating Unit test cases to cover all the code segments.
Integration and System testing.	Once the implementation and unit testing is completed integration testing can be started to test the flows and scenarios. Then deploy the system to QA server and evaluate the compatibility of the system.
Perform user acceptance testing (UAT)	Release the beta version of the system for User Acceptance Testing.
Maintenance	After the successful UAT cycle deploy the system to the Live Server and do the Maintenance.

TABLE 2.1 PHASE WISE DESCRIPTION OF THE WATERFALL MODEL

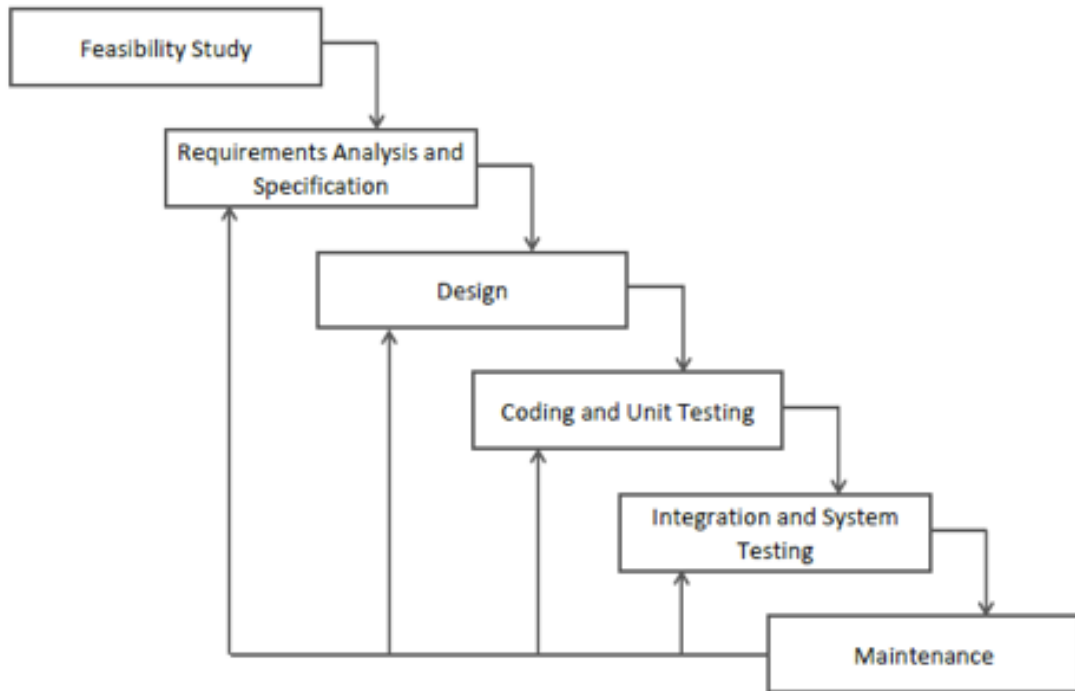


FIGURE 2.4 WATERFALL METHOD

The graphical overview of the Iterative waterfall methodology shown in Figure 2.4. According to it, from Requirement Analysis and Specification to Maintenance, all the steps can be re-considered according to the iterative model during project on going.

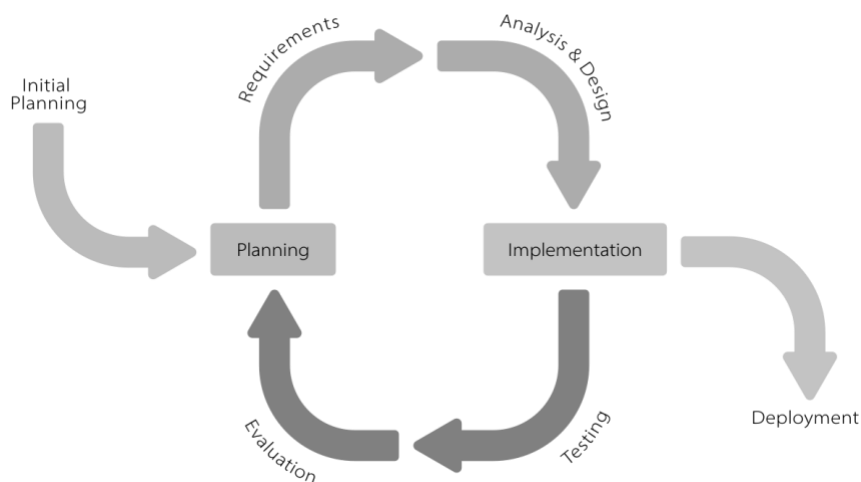


FIGURE 2.5 ITERATIVE MODEL

The Iterative model shown in the Figure 2.5. The iterative model begins with a small part of the requirement and expands the scope step by step with analysis, testing and evaluation steps. Deploy when satisfied with implementation.

Chapter 3 – Design

3.1 Introduction

The previous chapter describes the difference between the proposed method and the traditional method and the existing similar methods. This chapter describes how and when the technologies and technology to be used to solve the above problems are appropriate.

3.2 System Architecture

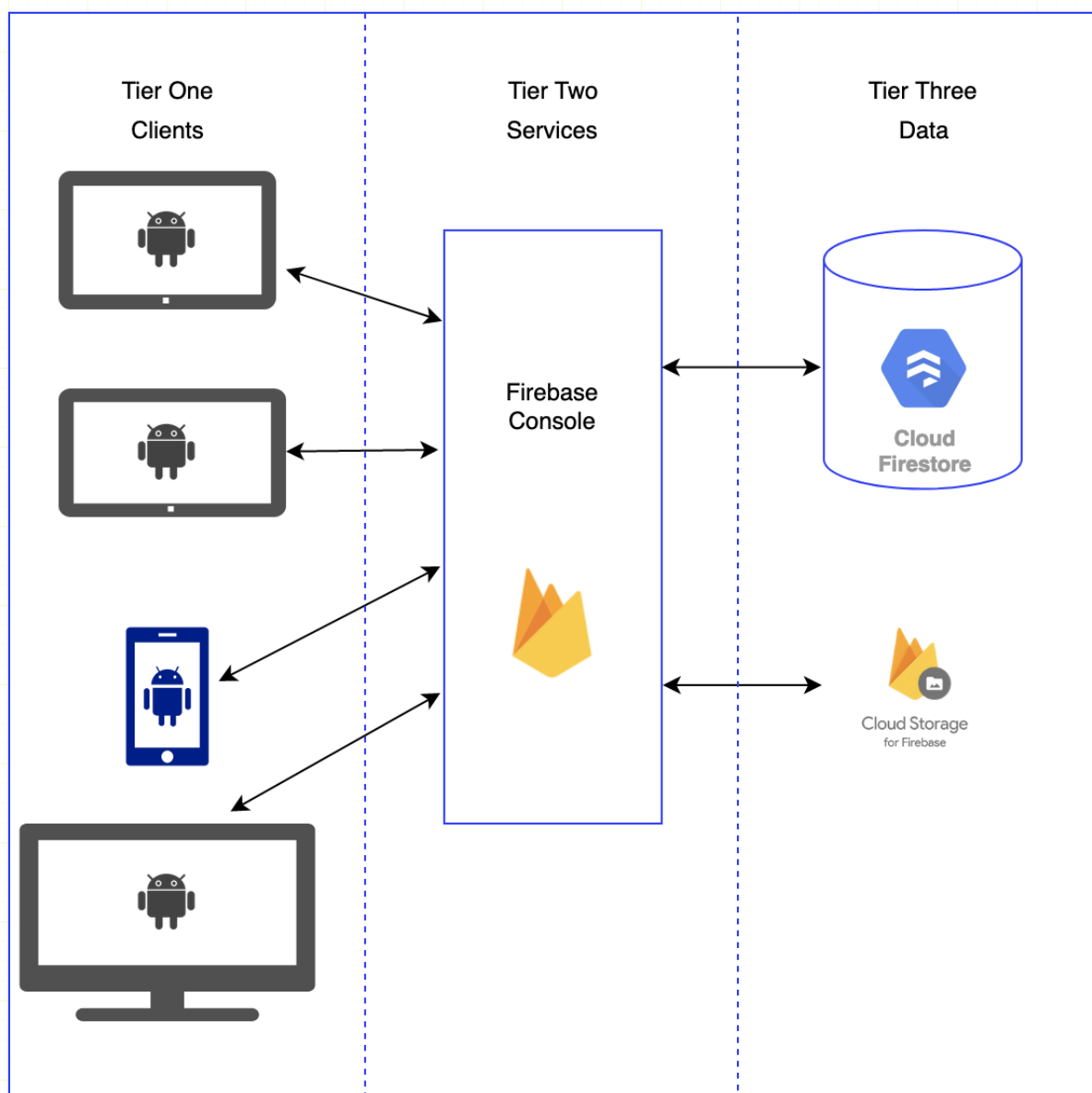


FIGURE 3.1 SYSTEM ARCHITECTURE

It consists of three layers, the client layer, the service layer and the data access layer, respectively. The client layer is intimidating for the system and user. Users can interact with the system through a mobile app, tablet app or TV app. All system processors and logics consist of client layers such as calculations, validations etc. All the services are based on the service layer. Moves the database to the data access layer, which include the firestore database and cloud storage from the firebase. That is used to store information according to the NoSQL database which based on document based database. As mentioned above, the interface / client layer consists of three components that build the system based on the three-tier software architecture.

3.3 Software Architecture

Backend of the system is based on Firebase console with the real time concept, and Mobile Application, TV Application and Tablet Application are based on MVVM architecture.

MVVM Architecture

The **Model, View, ViewModel (MVVM pattern)** is all about guiding you in how to organize and structure your code to write maintainable, testable and extensible applications.

Model – It simply holds the data and has nothing to do with any of the business logic.

ViewModel – It acts as the link/connection between the Model and View and makes stuff look pretty.

View – It simply holds the formatted data and essentially delegates everything to the Model. holds the formatted data and essentially delegates everything to the Model. (tutorialspoint, 2021)

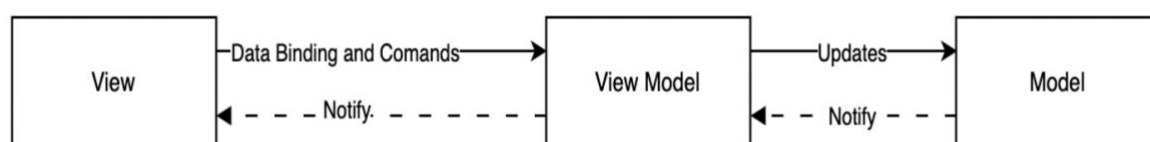


FIGURE 3.2 MVVM ARCHITECTURE

Diagram which is shown in Figure 3.2 shows the MVVM architecture. According to that View, View Model and Model Communicating sequence and Model, View Model and View notify sequence.

Base Components

There are two main components using in Android Applications. The Activities and Fragments are the main two principles using in application model.

Manifest file – Manifest.xml file is working as the root file of an Android project. All the application-level permissions, Activities, Services, Receivers, Providers are defined here.

Activity - An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window in an Android Application. (Android, 2021)

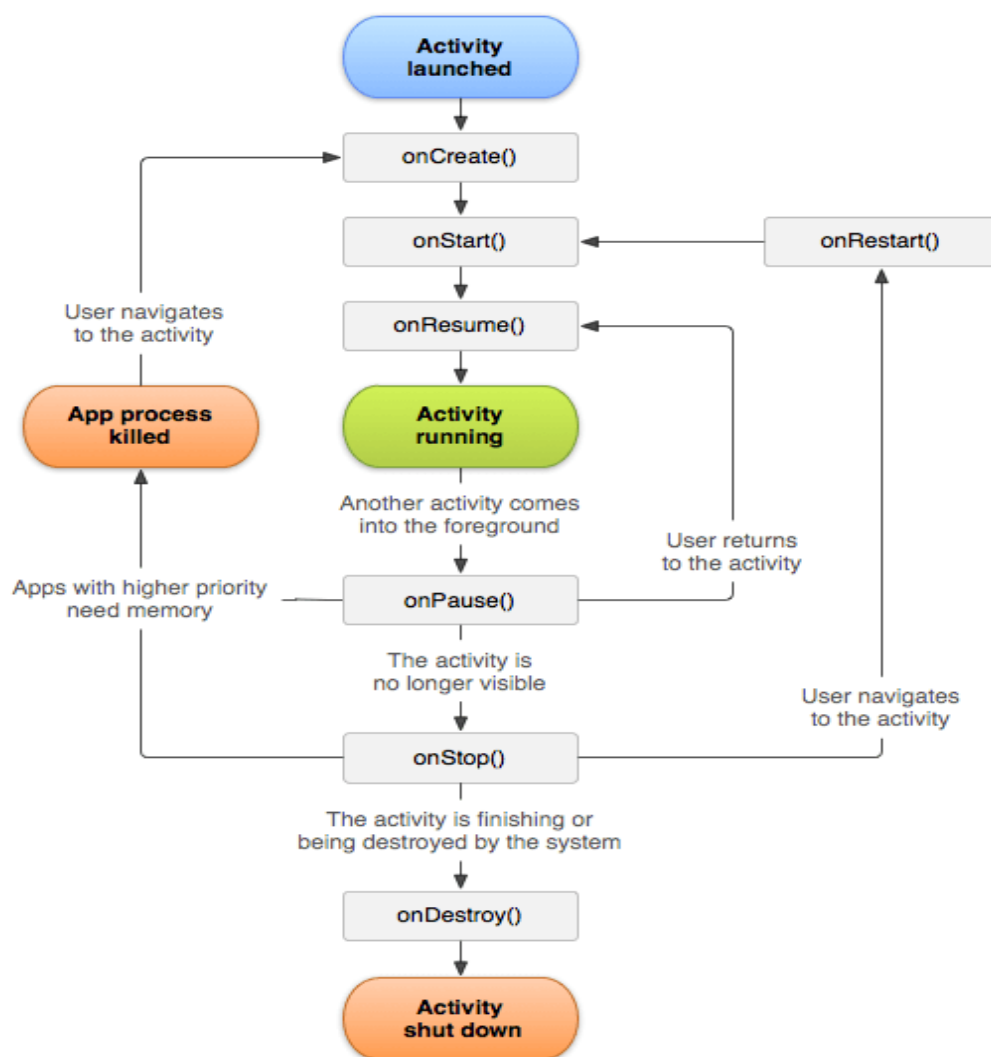


FIGURE 3.3 ACTIVITY LIFECYCLE

The complete lifecycle is shown in Figure 3.3. There should be a launch activity defined in the Manifest file and launch activity appear first when the app open. According to the graph each method is override in certain situations. As an example, In the activity initiation it calls onCreate() method, All the components should be defined inside the onCreate() method. When activity come back from background it calls onResume() method. When the app goes background it calls onPause() method. When activity closed it calls onDestroy() method.

Fragment – Each Fragment instance has its own lifecycle. When a user navigates and interacts with the app, your fragments transition through various states in their lifecycle as they are added, removed, and enter or exit the screen. (Android, 2021)

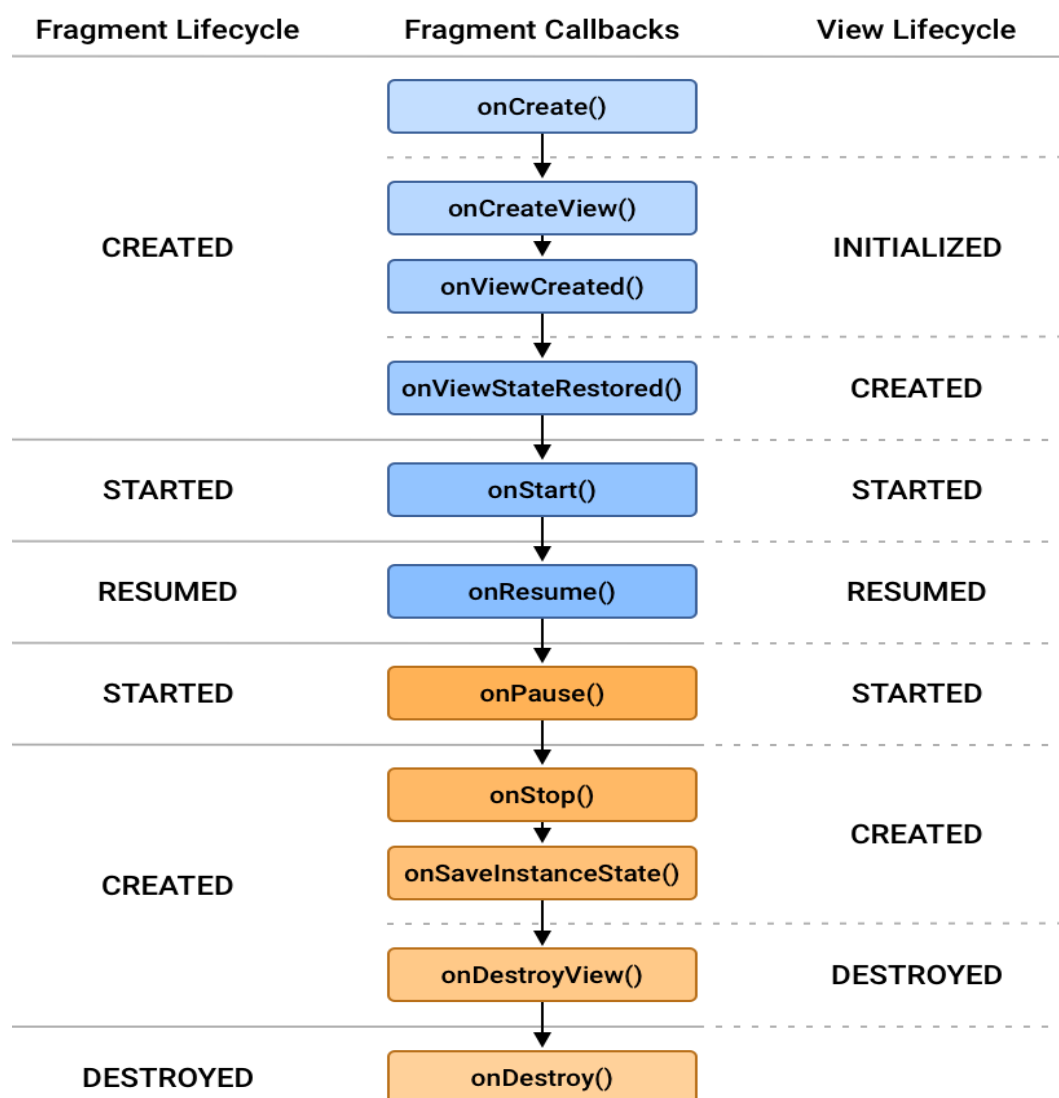


FIGURE 3.4 FRAGMENT OVERVIEW

The complete lifecycle of the fragment is shown in Figure 3.4. According to that Fragment also has a set of override methods which calls in a certain situation.

Cloud Firestore Architecture

Cloud Firestore is based on NoSQL, Document oriented database. Unlike an SQL database, there are no tables or rows. Instead, gather data in documents. And the documents are organized in the collections. Each document contains a pair of key values. Cloud Firestore is optimized for storing large collections of small documents. All documents should be stored in collections. Documents can contain subsets and nested objects, both of which can include complex objects such as threads, primary fields, or lists. (Firebase, 2021)

- Collection

In Cloud Firestore collection is working as the container for the documents.

- Document

In Cloud Firestore one record calls as a document. Data storing inside the document as a key-value pairs. Document has a unique key to identify the document. And the key should be unique to the document and value can any data with any data type. Document can be have sub-collections.

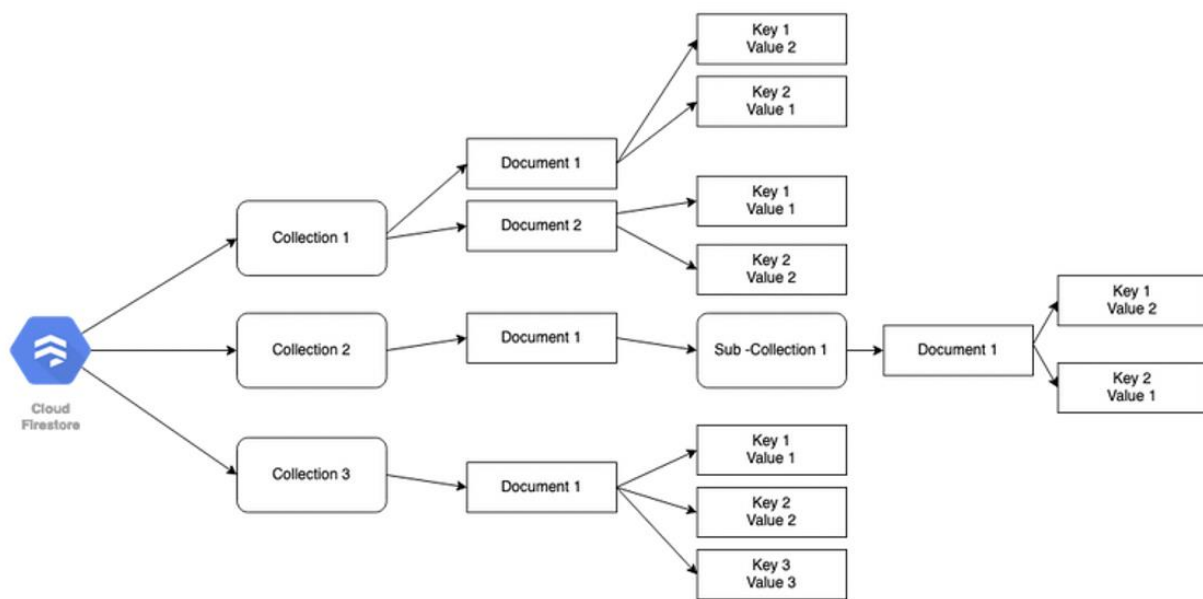


FIGURE 3.5 DATA ORGANIZATION IN FIRESTORE

Diagram which shown in Figure 3.3 shows the data organization structure of the firestore. Top level of the structure is collection. Collection can have multiple documents with a unique document id for the collection. Document id should not be unique for the different collections.

Document can be having a sub-collection. Under each document data should be store with the key-value pairs. These keys should be unique for each document.

3.5 UML Diagrams

UML stands for Integrated Format Language. Simply UML is an innovative approach to software modeling and documentation. In fact, it is one of the most popular business process modeling technology. It is based on diagrams of software components. As the old proverb says: "A picture is worth a thousand words". Using visual representations allows us to better understand potential flaws or errors in software or business processes.

There are several types of UML diagrams, each of which serves a different purpose, either before or after implementation (as part of the documentation). The two most broad categories that encompass all other types are Behavioral UML diagram and Structural UML diagram. As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas other describe the behavior of the system, its actors, and its building components. The different types are broken down as follows. (Ceta, 2020)

Use case diagram

Use case diagrams are usually developed in the early stages of development and people often use templates for the following purposes. (visual-paradigm, 2021)

1. Specify the context of a system
2. Capture the requirements of a system
3. Validate a systems architecture
4. Drive implementation and generate test cases
5. Developed by analysts together with domain experts

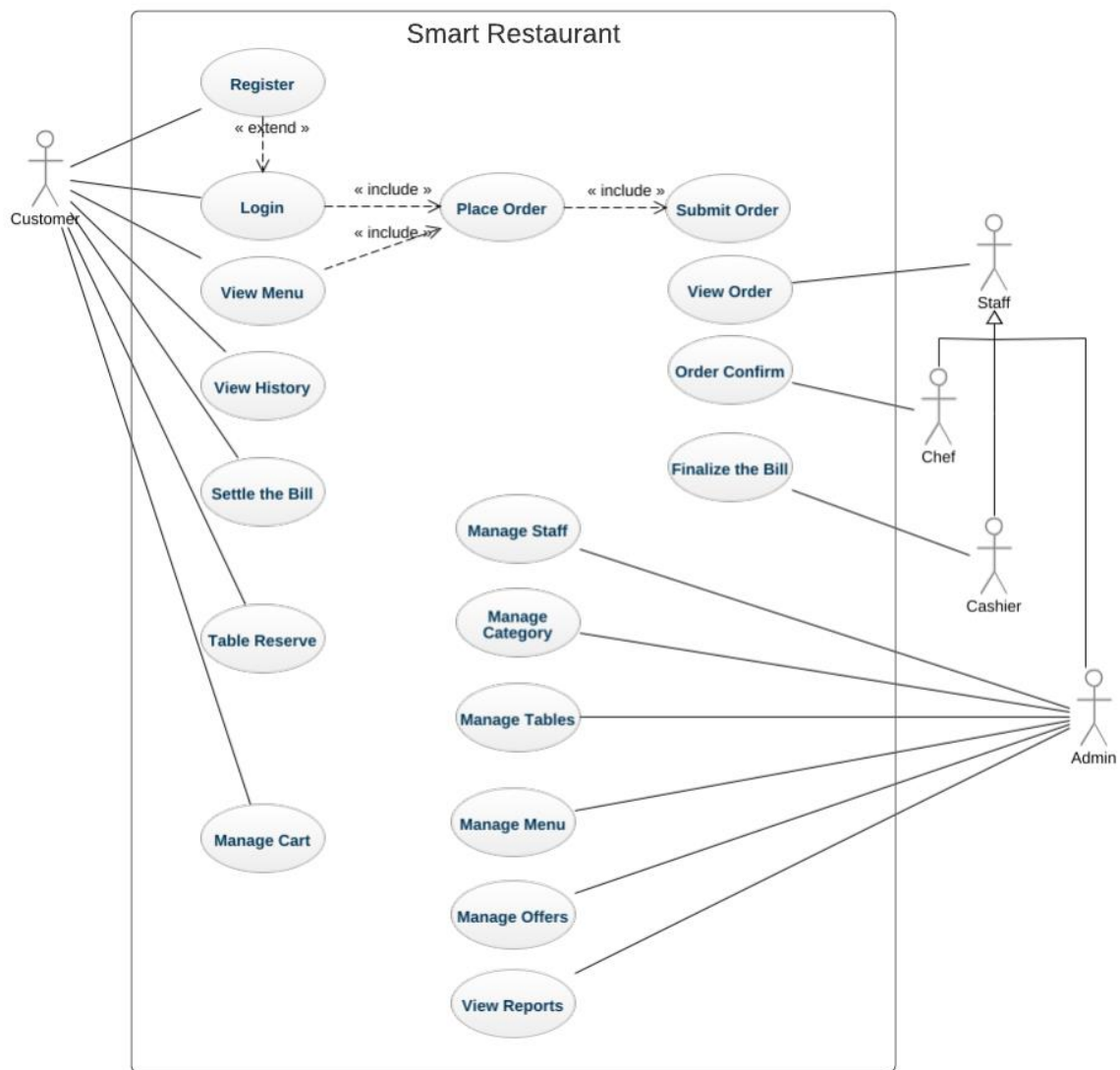


FIGURE 3.6 SMART RESTAURANT USE CASE DIAGRAM

According to the Use case diagram which showed in Figure 3.4,

- Customer should login to the Application to check the menu. If it is a new customer should register.
- Customers are able to place an order and submit the order.
- Customers can view the history of the orders and re-order an old order.
- Customers can settle the bill and terminate the session.
- Staff members have three main categories. Admin, Cashier and Chef.
- Manager is the only authorized person to create/update/delete the Menu items, Staff, Category, Offers and Tables.
- Chef is the only person who can accept order.

- Any staff member can view the details of an order.

Class Diagram

The main building block of object-oriented modeling is class diagrams. They are used to show the various objects in a system, their properties, their functionality, and the relationships between them.

The classes are interconnected in a specific way. Relationships, especially in class diagrams, involve a variety of logical relationships. The following are possible logical relationships in UML.

- Association
- Directed Association
- Reflexive Association
- Multiplicity
- Composition
- Aggregation
- Inheritance/Generalization
- Realization

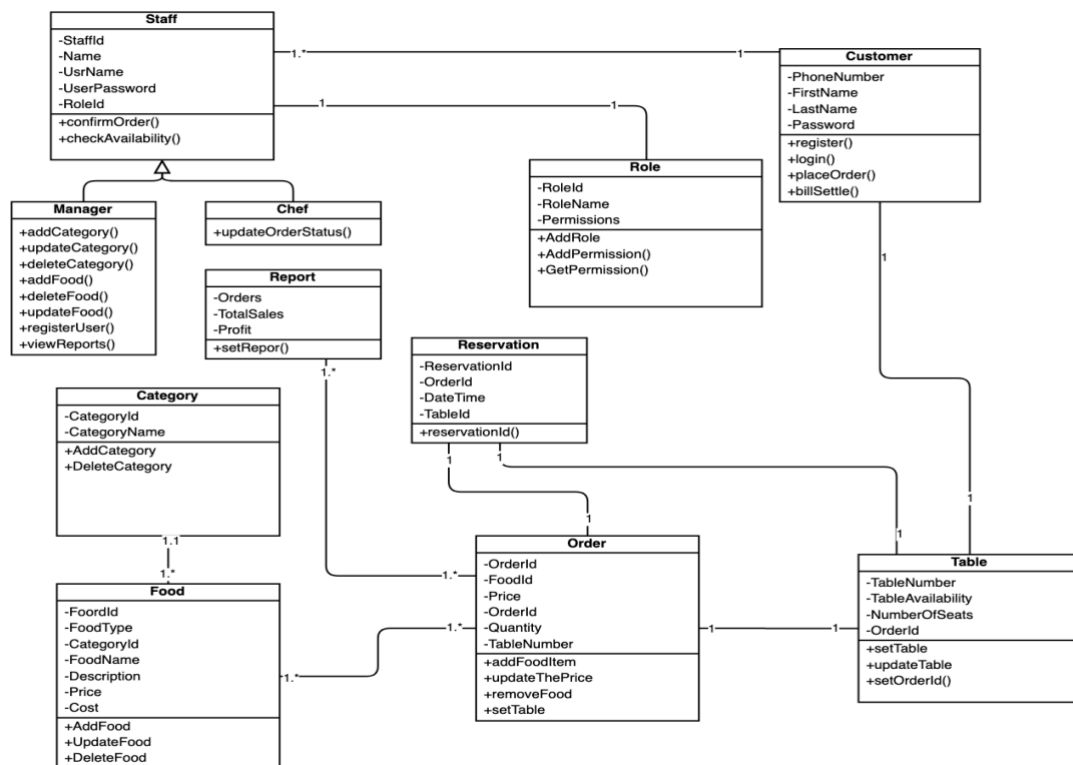


FIGURE 3.7 SMART RESTAURANT CLASS DIAGRAM

According to the class diagram which showed in the Figure 3.5,

- One order can be assigned to a table one time. When the customer settles the bill order session will be terminated. If the customer needed another item should be processed as a new order.
- For a one reservation one order should be assigned.
- Each food item has one category.
- Orders should be directly converted into the reports.
- Each staff member has one role which has its own permissions.

Activity Diagram

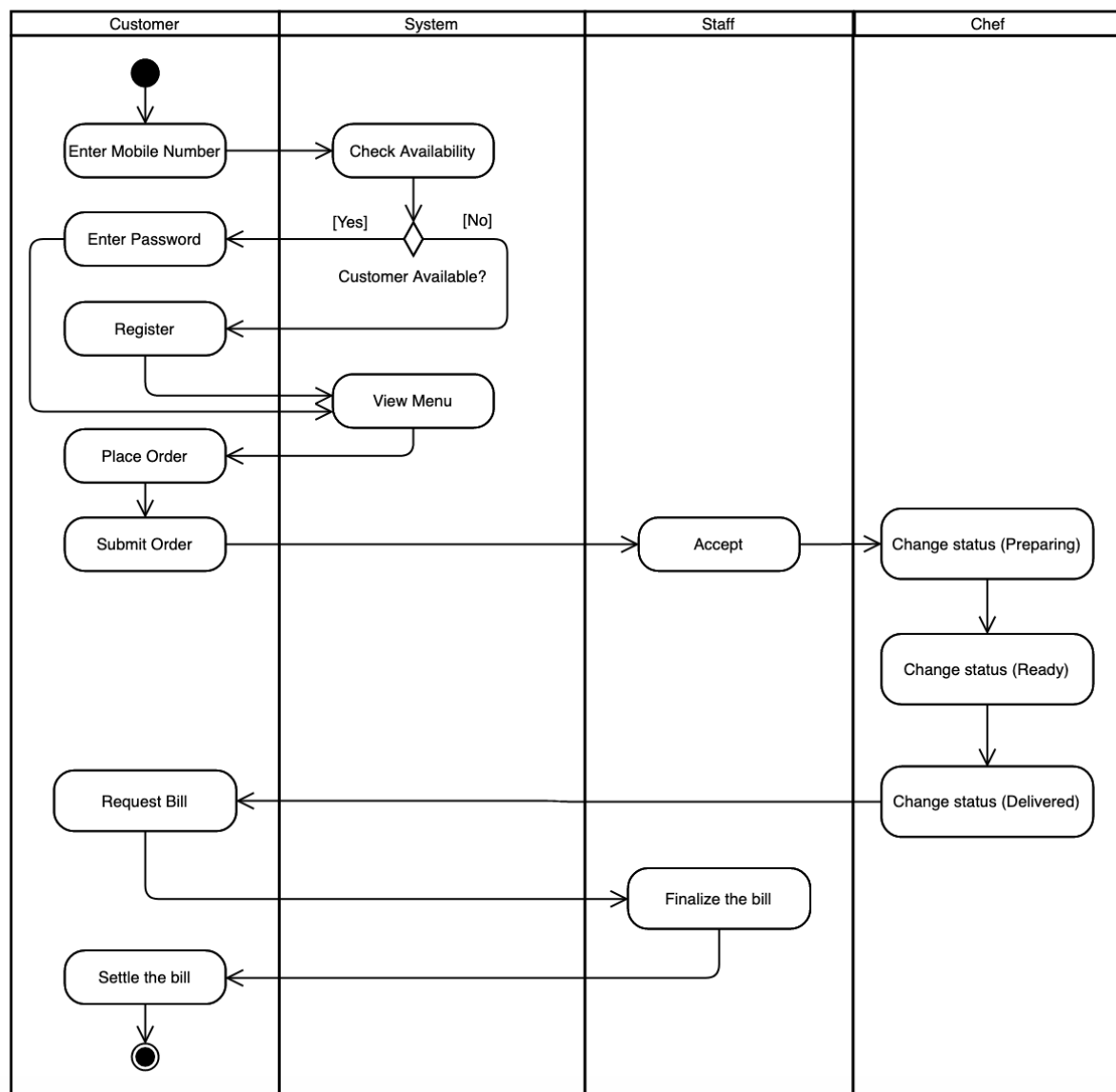


FIGURE 3.8 SMART RESTAURANT DINING ORDER PLACEMENT ACTIVITY DIAGRAM

Activity Diagram which is shown in Figure 3.6 explains the food ordering flow when customers come to the dining. When the customer arrives at the restaurant, he / she can place an order on the tablet attached to the table.

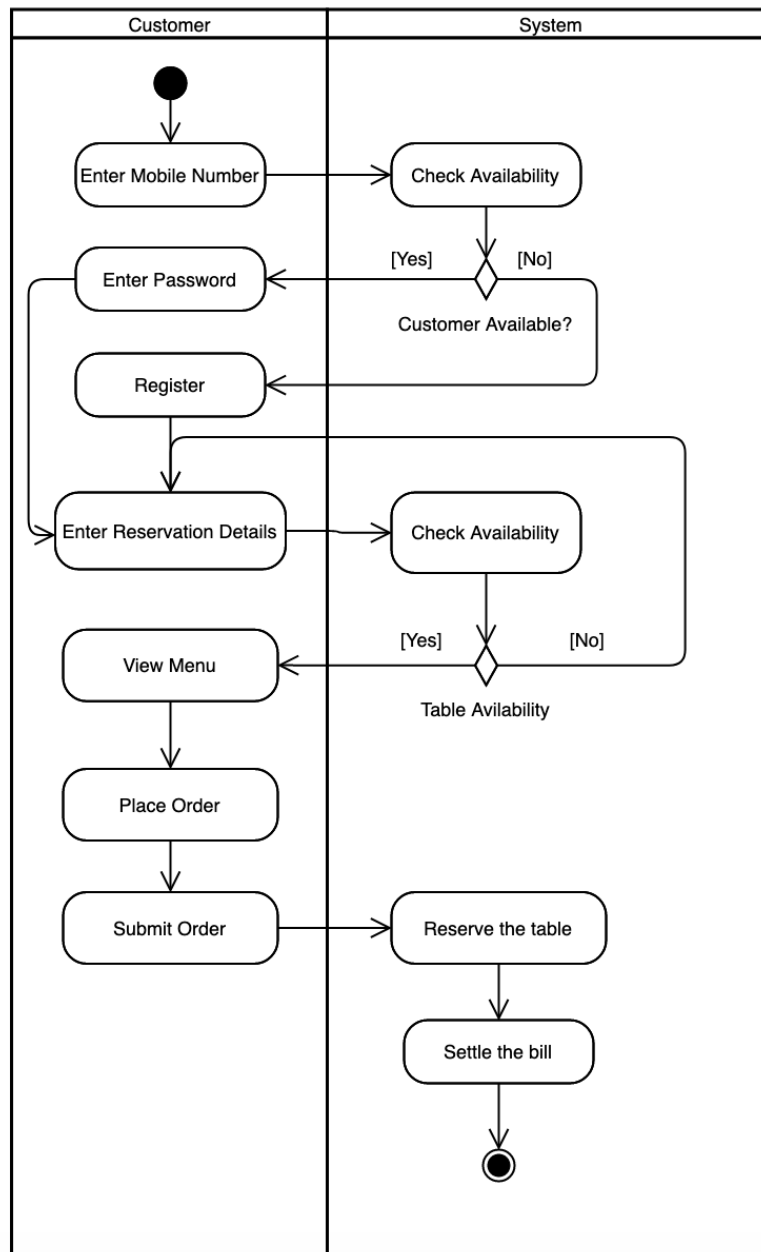


FIGURE 3.9 SMART RESTAURANT MOBILE RESERVATION ACTIVITY DIAGRAM

Activity Diagram which is shown in Figure 3.7 explains the table reservation flow through the mobile application. It checks the availability of the dining table according to the time slot entered. When a dining table is available, the customer can place an order. The table is reserved for the time slot which entered.

Chapter 4 Implementation

4.1 Introduction

This chapter contains a description of the development practices used to implement the project. The code module and database implementation are the most critical aspects of this chapter. And also, this chapter discusses the implementation environment selection process and resource requirements.

4.2 Implementation environment

In this section describes the resources which used during the implementation.

Used software and tools

Android Studio 4.1.1 is the main IDE used to do the development.

MacOS Catalina – It is giving smooth development experience in MacOS while developing Android developments and Android Studio is a heavy IDE. It needs a better processor with at least 8GB ram. Here in Smart Restaurant system included 4 android applications. Therefore, it was necessary to develop in parallel when each application was connected.

Android development environment used and the min SDK version is 22 (Android Lollipop) and the compile SDK version is 30 (Android Red Velvet Cake).

Kotlin, Java used as the programming languages and XML used to create the UIs.

Cloud Firestore used as the cloud database and Firebase cloud storage used to store the files.

Firebase Crashlytics to monitor the app crashes.

Architecture Components

MVVM is the software architecture which has used for all the four applications. It has main three layers. Model, View and ViewModel.

Model

Model has included the standard Kotlin data classes which can be convert the firestore document in to an Object and also this data class can be directly sync as the key-value pairs to the Firestore.

```
@Parcelize
data class Category(
    var catID: String? = "",
    var name: String? = "",
    var image: String? = ""
) : Parcelable
```

```

@Parcelize
data class Type(
    var typeId: String? = "",
    var name: String? = ""
) : Parcelable

```

```

@Parcelize
data class Food(
    var cartItemId: String? = "",
    var foodId: String? = "",
    var cat: String? = "",
    var cost: Double? = 0.0,
    var desc: String? = "",
    var image: String? = "",
    var name: String? = "",
    var price: Double? = 0.0,
    var type: String? = "",
    var isAdded: Boolean = false,
    var count: Int = 1,
    var cartId: String = ""
) : Parcelable

```

Kotlin data classes under the menu defined like above. All the field has a default value to avoid the null point exceptions.

View

View is working as the presentation layer. This includes activities, fragments and adapters. Each Activity has extended to a BaseActivity class which included the common methods which are using in the activities.

```

fun showToast(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}

```

showToast() is the method which calls when needs to show a Toast message.

```

private fun setProgress() {
    dialog = AppUtils.showProgress(this)
}

```

```

fun showProgress() {
    if (!dialog.isShowing) {
        dialog.show()
    }
}

```

```

fun hideProgress() {
    try {
        dialog.dismiss()
    } catch (e: Exception) {
        println(e)
    }
}

```


setProgress() , showProgress() and hideProgress() functions have used to define the progress bar, show progress bar and hide progress bar. In the activities and fragments commonly defines the viewmodel.

And the activity/fragment should be observed to the state which defined in the viewmodel and other variables.

ViewModel

ViewModel is included the business logics of each Activity and Fragment. ViewModel can be reusable with multiple Activities and Fragments. BaseViewModel has extended to each ViewModel and it is included some common functions and instances.

```
enum class Status {  
    LOADING,  
    SUCCESS,  
    SUB_SUCCESS1,  
    SUB_SUCCESS2,  
    SUB_SUCCESS3,  
    SUB_SUCCESS4,  
    COMPLETE,  
    CART_NOT_EMPTY,  
    CART_EMPTY,  
    ERROR,  
    VALIDATION_ERROR,  
    RESPONSE_ERROR,  
    LIST_EMPTY,  
    LOGOUT,  
    SPLASH_TO_LOGIN,  
    SPLASH_TO_HOME,  
    WRONG_VERSION,  
    CORRECT_VERSION,  
    UNAUTHORIZED,  
    VERSION_ERROR,  
    TIMEOUT,  
    RESET,  
    APPBLOCK,  
    APPBLOCK_WARNING,  
    APPUNBLOCK,  
    PHONE_NUMBER_EXISTS,  
    ITEM_EXISTS  
}
```

Status enum has defined the all-possible status can be taken during the business process.

ViewModelState class has defined all the states callbacks can be taken during the business process inside the ViewModel class. This returns the Status value to the view from ViewModel. Few of the functions are shown below.

```

fun loading(): ViewModelState {
    return ViewModelState(Status.LOADING)
}
fun success(): ViewModelState {
    return ViewModelState(Status.SUCCESS)
}
fun sub_success1(): ViewModelState {
    return ViewModelState(Status.SUB_SUCCESS1)
}
fun cartEmpty(): ViewModelState {
    return ViewModelState(Status.CART_EMPTY)
}

var state: MutableLiveData<ViewModelState>? = null

var database: FirebaseFirestore? = null
var storage: StorageReference? = null

private fun defineFirestore() {
    database = FirebaseFirestore.getInstance()
    storage = FirebaseStorage.getInstance().reference
}

```

Here in BaseViewModel class defined the state as a MutableLiveData. This helps to separate the ViewModel and View (Business layer and Presentation layer) without sharing any instance between them. That avoids the keeping unnecessary instance and reduce the memory leaks.

FirebaseFirestore instance and Firebase StorageReference are defined inside the BaseViewModel and it can be access from any viewmodel class. Using the FirebaseFirestore instance can be save, fetch, update and delete data from Firestore database.

Here some main functionalities which needed to communicate with Firebase Firestore and Firebase Storage.

Get data from Firestore Collection

```

fun getCategories() {
    state!!.postValue(ViewModelState.loading())
    database!!.collection(TABLE_FOOD_CATEGORY)
        .get().addOnSuccessListener { documents ->
            var catList = ArrayList<Category>()
            var docs = documents.documents
            if (docs.isNotEmpty()) {
                for (d in docs) {
                    val category =
                        Category(d.getString("id"), d.getString("name"), d.getString("image"))
                    catList.add(category)
                }

                if (currentSelectedCategoryIndex == -1) currentSelectedCategoryIndex = 0
                if (currentSelectedCategory.isNullOrEmpty()) currentSelectedCategory =
                    catList!![0].catID!!
            }
            displayCategoryData(catList)
            allCats.clear()
        }
}

```

```

        allCats.addAll(catList)
        getItems()
    }
    .addOnFailureListener { exception ->
        Log.w("TAG", "Error getting documents: ", exception)
        state!!.postValue(ViewModelState.error())
    }
}

```

This is a sample code for fetch the Category data set from Firestore. This shows how to fetch the all the data from the collection Category.

Insert data to Firestore Collection

```

private fun addAllData(
    url: String,
    name: String
){
    val item: MutableMap<String, Any> = HashMap()
    item[COLUMN_NAME] = name.toUpperCase()
    item[COLUMN_IMAGE] = url
    item[COLUMN_ID] = System.currentTimeMillis().toString() + name

    database!!.collection(TABLE_MENU)
        .whereEqualTo(COLUMN_NAME, name)
        .get()
        .addOnSuccessListener {
            if (it.documents.isEmpty()) {
                database!!.collection(TABLE_FOOD_CATEGORY)
                    .document(System.currentTimeMillis().toString() + name)
                    .set(item)
                    .addOnSuccessListener { documentReference ->
                        Log.d(
                            ContentValues.TAG,
                            "DocumentSnapshot added with ID: $documentReference"
                        )
                        state!!.postValue(ViewModelState.success())
                    }.addOnFailureListener {
                        println(it)
                        state!!.postValue(ViewModelState.error())
                    }
            } else {
                state!!.postValue(ViewModelState.item_exists())
            }
        }
}

```

This is a sample code for add Menu item to the Firestore menu collection. Firstly, it is checking the availability of the Menu item in the Menu collection. To do that fetch the item list which has the same name to the Menu item which going to be added to the collection. If the fetched list is empty, that menu item is not available.

Delete a document from Firestore Collection

```
database!!.collection(TABLE_CART_ITEMS).document(doc).delete()
```

This is a sample code for delete a Cart Item from cart_items collection. To delete a record from a collection we should delete the document. To do that firstly it should find the document id. Then using that selected document id, it can be deleted the record permanently.

Update a field in a document

```
database!!.collection(TABLE_CART)
    .document(docId)
    .update(
        mapOf(
            COLUMN_AMOUNT to currentAmount
        )
    )
```

This is a sample code to update an amount in the cart collection. It should get the document id and can be updated the record.

File upload to Firebase Storage

```
fun uploadFile(
    imageUri: Uri,
    name: String
){
    state!!.postValue(ViewModelState.loading())
    if (imageUri != null) {
        val fileReference =
            storage?.child("${System.currentTimeMillis()}")
        fileReference?.putFile(imageUri)?.addOnSuccessListener { task ->
            fileReference.downloadUrl.addOnSuccessListener {
                addAllData(it.toString(), name)
            }
        }
        ?.addOnFailureListener {
            state!!.postValue(ViewModelState.loading())
        }
        ?.addOnProgressListener { takeSnapshot ->
            val progress =
                (100.0 * takeSnapshot.bytesTransferred / takeSnapshot.totalByteCount)
        }
    }
    } else {
        state!!.postValue(ViewModelState.error())
    }
}
```

This is a sample code for upload the files to the firebase storage and get the URL of the image.

UI Design

There are two types of users going to be involved with this system.

1. Customers.
2. Staff.

Customer based applications are targeting any type of users. And the staff-based applications are targeting the trained employees. To satisfy the both of the user types, UI output should be more flexible and user friendly. The applications are provided a minimal number of informative illustrations and avoided unnecessary distractions.

Material design is a popular UI design standard for Android development which drives to support best practices. In the all-applications UI designs are completely based on the Material design concepts. Table tablet application, Admin application and Customer mobile application included the basic Splash screen, Login screen and Registration screens. And the main screens of these application are included a bottom navigation view as a main navigation component. And the Kitchen TV application is a single screen application. It contains only a one screen which has ability to control using the touch screen or TV remote.

UI Components

In android there are many UI components which can use to make the user experience easy. Some of the critical components are described below.

ViewGroup

In Android most popular three ViewGroups are LinearLayout, RelativeLayout and ConstraintLayout. To organize the view Vertically and Horizontally it mostly used LinearLayout. To create a simple, inter relation between view component it used RelativeLayout. To create complex layout, it used ConstraintLayout. ConstraintLayout is the most popular ViewGroup in Android which can be created advance constraint between view components.

RecyclerView

There are several ways to implement a scrolling list, grid views in Android. The most effective way to archive this kind of a task is using a RecyclerView. The base concept of the RecyclerView is reuse the same view item of the list while scrolling the list. RecyclerView adapter is doing this programmatically.

```

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvCategory"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"/>

```

This is a sample XML code for a RecyclerView.

```

class MenuCatAdapter(private var list: List<Category>) :
    RecyclerView.Adapter<MenuCatAdapter.ViewHolder>() {
    var selectedItem = 0
    override fun onCreateViewHolder(parent: ViewGroup, p1: Int): ViewHolder {
        return ViewHolder(
            LayoutInflater.from(parent.context).inflate(
                R.layout.layout_category_item,
                parent,
                false
            )
        )
    }

    override fun getItemCount(): Int {
        return list.size
    }

    @SuppressWarnings("ResourceAsColor")
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.name.text = getItem(position).name
        AppUtils.loadImagePicaso(getItem(position).image!!, holder.image)

        if (selectedItem == position) {
            holder.image.setBackgroundResource(R.drawable.drawable_selected_circle)
        } else {
            holder.image.setBackgroundResource(R.drawable.drawable_not_selected_circle)
        }
    }

    fun setItems(list: List<Category>) {
        this.list = list
    }

    fun setCurrentSelectedItem(pos: Int) {
        this.selectedItem = pos
    }

    fun getItem(pos: Int): Category {
        return list.get(pos)
    }

    class ViewHolder(
        itemView: View
    ) : RecyclerView.ViewHolder(itemView) {
        var name = itemView.tvName
        var image = itemView.ivImage
    }
}

```

This is a sample code for the RecyclerView adapter. It includes a ViewHolder which defines the view components of the itemview and the actions of the itemview components. Inside the

onCreateViewHolder() creates the itemview using the layout file. Inside the onBindViewHolder() bind the data to the view components. All the time it is getting updated while the RecyclerView scrolls.

BottomNavigationView

BottomNavigationView has used as the main navigation component. Except TV application other three apps main screen based on this view. To populate the navigation buttons bottom_navigation_menu.xml has load to the BottomNavigationView. This view is defined inside the MainActivity and to create the different layers of the screens, it has added different Fragments for each screen.

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="5dp"
    app:itemIconSize="25dp"
    app:labelVisibilityMode="labeled"
    app:menu="@menu/bottom_navigation_menu"
    android:background="@color/transparent_black"/>
```

Sample xml code to the BottomNaviationView.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/bottom_navigation_home"
        android:icon="@drawable/ic_menu"
        android:title="@string/menu" />

    <item
        android:id="@+id/bottom_navigation_orders"
        android:icon="@drawable/ic_orders"
        android:title="@string/orders" />

    <item
        android:id="@+id/bottom_navigation_cart"
        android:icon="@drawable/ic_cart"
        android:title="@string/cart"
        app:actionLayout="@layout/layout_cart_count"/>

    <item
        android:id="@+id/bottom_navigation_offers"
        android:icon="@drawable/ic_offeres"
        android:title="@string/offers" />

    <item
        android:id="@+id/bottom_navigation_account"
        android:icon="@drawable/ic_account"
        android:title="@string/account" />
</menu>
```

Sample code to bottom_navigation_menu.xml

```
private fun loadFragment(fragment: Fragment) {
    val transaction: FragmentTransaction = supportFragmentManager.beginTransaction()
    transaction.replace(R.id.flContent, fragment)
    transaction.addToBackStack(null)
    transaction.commit()
}
```

Sample code to load a fragment to the content view inside the main_activity.

Utils

AppUtils

Under the helper package AppUtils class has all the commonly used functions.

Ex – startActivity(), closeActivity() function including the Animations.

```
fun startActivityRightToLeftForResult(
    activity: Activity,
    aClass: Class<out Activity>,
    requestCode: Int
){
    val intent = Intent(activity, aClass)
    activity.startActivityForResult(intent, requestCode)
    activity.overridePendingTransition(R.anim.pull_in_right, R.anim.push_out_left)
}

fun closeActivityLeftToRight(activity: Activity) {
    activity.overridePendingTransition(R.anim.left_in, R.anim.slide_to_right)
}
```

SharedPref

This is where the SharedPreferences defines. All the types saving to the SharedPreferences and retrieving from SharedPreferences has defined here.

```
fun saveString(key: String, value: String) {
    val editor = Restaurant.instance.getSharedPreferences(
        SharedPref.FILE_KEY, Context.MODE_PRIVATE).edit()
    editor.putString(key, value)
    editor.commit()
}

fun getString(key: String, defaultval: String): String? {
    val prefs = Restaurant.instance.getSharedPreferences(SharedPref.FILE_KEY,
        Context.MODE_PRIVATE)
    return prefs.getString(key, defaultval)
}
```

Constants

All the constants are defined inside this class.

Chapter 5 Testing and Evaluation

5.1 Introduction

This chapter describes the approach to taken to test the system and evaluated with a set of test cases that has been identified in the system. In addition, essential descriptions and definitions related to software testing, third-party libraries, functional behaviors, software testing rules. There is also Firebase platform, by integrating Firebase to the Android Studio it can be test the behavior of the applications through the test tools of Firebase, such as Analytics, Crashlytics, and Performance Monitoring.

5.2 Testing Types

Throughout the testing process it has used three main types of testing.

1. Unit Testing
2. Instrumented Testing
3. Useability Testing.

Unit Testing

It does not need an actual device to run the unit test, it must be set up on a Java virtual machine and unit testing should be performed on it.

In Smart Restaurant Unit tests are written with JUnit, a standard unit testing framework for Java. In these unit tests checking the specific function or check the particular code part working properly as desired.

Here is an example Unit test which created to test the total value of the cart.

```
@RunWith(AndroidJUnit4::class)
class CartTest {
    @Test
    fun cartSum(){
        var currentSum = 0.0
        val cartList = ArrayList<Food>()
        currentSum = 0.0

        val food1 = Food(price = 100.0, count = 10)
        val food2 = Food(price = 200.0, count = 20)
        val food3 = Food(price = 300.0, count = 30)
        cartList.add(food1)
        cartList.add(food2)
        cartList.add(food3)

        for (cl in cartList) {
```

```

        currentSum += (cl.price!! * cl.count)
        println(currentSum)
    }
    Assert.assertEquals("14000.0", currentSum.toString())
}
}

```

Instrumented Testing

Test the application using a real device or emulator called instrumented testing. These test result can be view on a screen. In most cases, they are written to test the integrity of the application and the accuracy of the UI Automation of functionality and user interaction. The majority of the instrumentation tests are written using Espresso library.

Useability Testing

Two small groups of people were chosen to test the 4 applications separately. To test the customer applications joined the people who does not have much technical experience. Because the Customer mobile application and Customer Table Tablet applications are created to the usage of the any person who are coming to the restaurant. The admin application and the Kitchen TV application were tested by people who have some administration knowledge. To distribute the application used the Firebase App distribution feature. By using the emails of the users can send the apks of the applications to the users.

5.3 Test Cases

Customer Tablet Application

The following test cases were identified to verify the customer tablet application functionality.

Test Id	Test Case	Pre-Condition	Steps	Input data	Expected output
TC1	Register a Table	<ul style="list-style-type: none"> Should be available a table 	<ol style="list-style-type: none"> Open the table register screen. Select a table number. Click Add button. 	<ul style="list-style-type: none"> Table number 	Go to the Login Screen.

TC2	Valid User Registration -	<ul style="list-style-type: none"> • A user not logged in. • Use an unregistered phone number. 	<ol style="list-style-type: none"> 1. Go to the Login screen. 2. Click Register button. 3. Open the Register screen. 4. Enter the register details. 5. Click the Register button. 	<ul style="list-style-type: none"> • Name. • Phone Number. • Password. • Confirm Password. 	Go to the Main Screen
TC3	Invalid User Registration —	<ul style="list-style-type: none"> • A user not logged in. • Use a registered phone number. 	<ol style="list-style-type: none"> 1. Go to the Login screen. 2. Click Register button. 3. Open the Register screen. 4. Enter the register details. 5. Click the Register button. 	<ul style="list-style-type: none"> • Name. • Phone Number. • Password. • Confirm Password. 	Show the message “Phone number already exists”
TC4	Login with valid credentials	<ul style="list-style-type: none"> • A user not logged in. • Use a registered phone number. • User the correct password. 	<ol style="list-style-type: none"> 1. Go to the Login Screen 2. Enter the login details 3. Click Login button 	<ul style="list-style-type: none"> • Phone Number • Password 	Go to the Main Screen
TC5	Login with unregistered phone number	<ul style="list-style-type: none"> • A user not logged in. • Use an unregistered phone number 	<ol style="list-style-type: none"> 1. Go to the Login Screen 2. Enter the login details 3. Click Login button 	<ul style="list-style-type: none"> • Phone Number • Password 	Show the message “Login Failed”

TC6	Login with wrong password	<ul style="list-style-type: none"> • A user not logged in • Use a registered phone number. • Use a wrong password 	<ol style="list-style-type: none"> 1. Go to the Login Screen 2. Enter the login details 3. Click Login button 	<ul style="list-style-type: none"> • Phone Number • Password 	Show the message "Login Failed"
TC7	Place order	<ul style="list-style-type: none"> • User should be logged in. • Status should be Place an Order. 	<ol style="list-style-type: none"> 1. Open the Main Screen. 2. Add Food Items to the Cart 3. Click Next button. 4. Open the Cart Screen. 5. Click the Submit button. 	<ul style="list-style-type: none"> • Food Items. 	Status should change to "Pending"
TC8	No old orders	<ul style="list-style-type: none"> • User should be logged in. • There should not be previous orders to the logged user. 	<ol style="list-style-type: none"> 1. Open the Main Screen. 2. Open the Orders Tab. 		Show message "No orders to show"
TC9	Get old order	<ul style="list-style-type: none"> • User should be logged in. • Should have a previous order. 	<ol style="list-style-type: none"> 1. Open the Main Screen. 2. Open the Orders Tab. 3. Select an order. 4. Click the Apply button. 	<ul style="list-style-type: none"> • Food Items 	Add the previous order items to the current cart.
TC10	Add Offer	<ul style="list-style-type: none"> • User should be logged in. 	<ol style="list-style-type: none"> 1. Open the Main Screen 2. Open the offer Tab. 	<ul style="list-style-type: none"> • Offer details. 	Offer value should be added to the

		<ul style="list-style-type: none"> Offer should be available. 	<ol style="list-style-type: none"> Select an offer. Click Apply button. 		cart screen offer text area.
TC11	No Offers	<ul style="list-style-type: none"> User should be logged in. Offer should not be available. 	<ol style="list-style-type: none"> Open the Main Screen Open the offer Tab. 		Show message "No offers to show"
TC12	Order Processing	<ul style="list-style-type: none"> User should be logged in. There should be a submitted order. Order status should be "Pending" 	<ol style="list-style-type: none"> Open the Main Screen Open Cart tab. 		Status should be changed to "Processing"
TC13	Order Serve	<ul style="list-style-type: none"> User should be logged in. There should be a submitted order. Order status should be "Processing" 	<ol style="list-style-type: none"> Open the Main Screen Open Cart tab. 		Status should be changed to "Served"
TC14	Bill settlement pending	<ul style="list-style-type: none"> User should be logged in. There should be a submitted order. Order status should be "Served" 	<ol style="list-style-type: none"> Open the Main Screen Open Cart tab. Click Ready to Pay Confirm the action. 		Open the Bill Settle screen. And the correct payable amount should be show with the status

					“Bill Settlement Pending”
TC15	Bill Settle	<ul style="list-style-type: none"> • User should be logged in • There should be a submitted order. • Order status should be” Bill Settlement Pending” • Bill Settle screen should be open. 	1. Settle the Bill.		Status should be changed to “Bill Settled” and “Thank you message show” and within 5 seconds should be logout and go to the login screen.

TABLE 5.1 USER TABLET APPLICATION TEST CASES

Admin Tablet Application

The following test cases were identified to verify the admin tablet application functionality.

Test Id	Test Case	Pre-Condition	Steps	Input data	Expected output
TC16	Login with valid credentials	<ul style="list-style-type: none"> • A user not logged in. • Use a registered username. • User the correct password. 	<ol style="list-style-type: none"> 1. Go to the Login Screen 2. Enter the login details 3. Click Login button 	<ul style="list-style-type: none"> • Username • Password 	Go to the Main Screen

TC17	Login with unregistered user	<ul style="list-style-type: none"> • A user not logged in. • Use an unregistered username. 	<ol style="list-style-type: none"> 1. Go to the Login Screen 2. Enter the login details 3. Click Login button 	<ul style="list-style-type: none"> • Username • Password 	Show the message “Login Failed”
TC18	Add a food	<ul style="list-style-type: none"> • Admin User should be logged in. 	<ol style="list-style-type: none"> 1. Go to the Food Tab. 2. Click add button. 3. Open new food screen. 4. Enter the food details. 5. Click Save button. 	<ul style="list-style-type: none"> • Name • Description • Category • Type • Cost • Price. • Image 	Go back to food screen and show the success message.
TC19	Add a Category	<ul style="list-style-type: none"> • Admin User should be logged in. 	<ol style="list-style-type: none"> 1. Go to the Category Tab. 2. Click add button 3. Open New category screen. 4. Enter the food details. 5. Click Save button. 	<ul style="list-style-type: none"> • Name • Image 	Go back to the category screen with success message.
TC20	Add a Table	<ul style="list-style-type: none"> • Admin User should be logged in. 	<ol style="list-style-type: none"> 1. Go to the Table Tab. 2. Click add button 3. Open New table screen. 	<ul style="list-style-type: none"> • Table Number • Number of seats. 	Go back to the table screen with

			4. Enter the table details. 5. Click Save button.		success message.
TC21	Add a Offer	<ul style="list-style-type: none"> Admin User should be logged in. 	6. Go to the Offer Tab. 7. Click add button 8. Open New offer screen. 9. Enter the offer details. 10. Click Save button.	<ul style="list-style-type: none"> Name Value Date 	Go back to the offer screen with success message.

TABLE 5.2 ADMIN APPLICATION TEST CASES

5.4 Functional Evaluation

The prototype is assessed by examining whether a prototype achieving the objectives stated in the project proposal. For the evaluations I have used two methods. Get the feedback from the users and use the Firebase tools.

User Evaluation

- Questionnaire / demonstration-based interviews targeting 15 people who are generally using restaurants for dining.
- Questionnaire followed by application description targeting 15 potential people who can use the admin application as a restaurant owner.

The questionnaire was prepared as below in Appendix A.

Firebase Tools

Firebase has provided number of useful tools to maintain the quality of the android apps. Three tools have used here in Smart Restaurant system.

- Crashlytics.

2. Analytics.
3. Performance.

Crashlytics

With Crashlytics it is easy to track the crashes of the applications while using them. This tool can be used with the development environment, UAT environment and also in Production environment.



FIGURE 5.1 CRASH-FREE STATISTICS

Referring the Figure 5.1 This is the graph which can get a general idea about the app application stability.

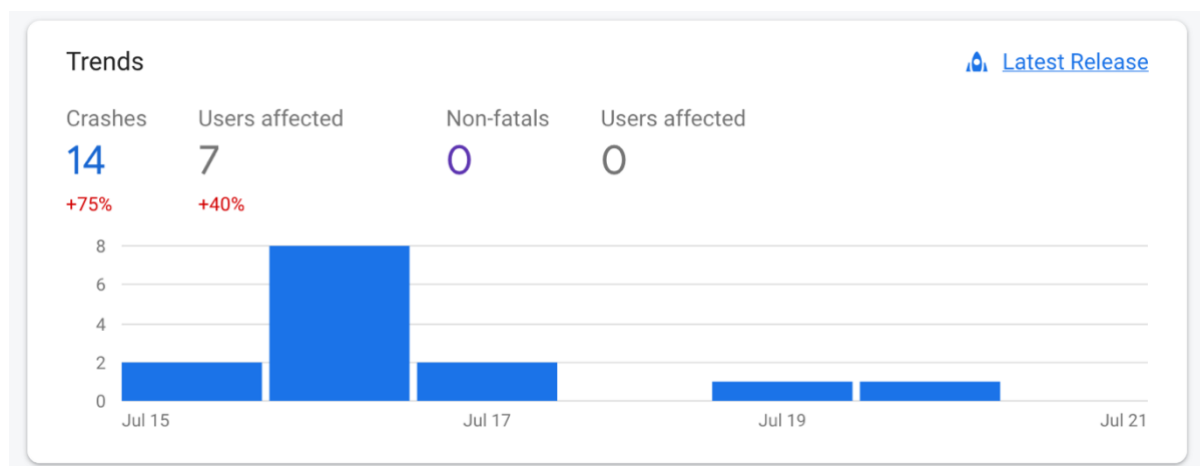


FIGURE 5.2 APP TRENDS

Referring Figure 5.2, the app trends it shows the number of crashes and the number of users affected. This can be filter with for different period of time

Issues

Search by user ID

Filter issues

Issue state = "Open"

Search issue title, subtitle or keys

Issues	Event type	Versions	Events	Users
<div>HomeViewModel.kt – line 114</div> <div>com.thejan.proj.restaurant.tablet.android.viewmodel.HomeViewModel\$getCurre...</div>	Crash	1.0 – 1.0	3	2
<div>BaseActivity.kt – line 45</div> <div>com.thejan.proj.restaurant.tablet.android.view.activity.BaseActivity.hideProgress</div>	Crash	1.0 – 1.0	2	1
<div>HomeViewModel.kt – line 47</div> <div>com.thejan.proj.restaurant.tablet.android.viewmodel.HomeViewModel.getCateg...</div>	Crash	1.0 – 1.0	2	1
<div>LoginViewModel.kt – line 22</div> <div>com.thejan.proj.restaurant.tablet.android.viewmodel.LoginViewModel.login</div>	Crash	1.0 – 1.0	2	2
<div>TableRegisterViewModel.kt – line 62</div> <div>com.thejan.proj.restaurant.tablet.android.viewmodel.TableRegisterViewModel\$g...</div>	Crash	1.0 – 1.0	1	1
<div>LoginActivity.kt – line 105</div> <div>com.thejan.proj.restaurant.tablet.android.view.activity.LoginActivity.update</div>	Crash	1.0 – 1.0	1	1

FIGURE 5.3 CRASHES

Referring Figure 5.3, all the crashes of the applications show including the file and code line details. And also it shows the number of time and number of users that effected.

From each crash it can be find the complete details about the devises wich these crashes effected. Memory details, Ram details, Device model, OS version, Application state (Background or Forgorund) are included there. Refer the Figure 5.4

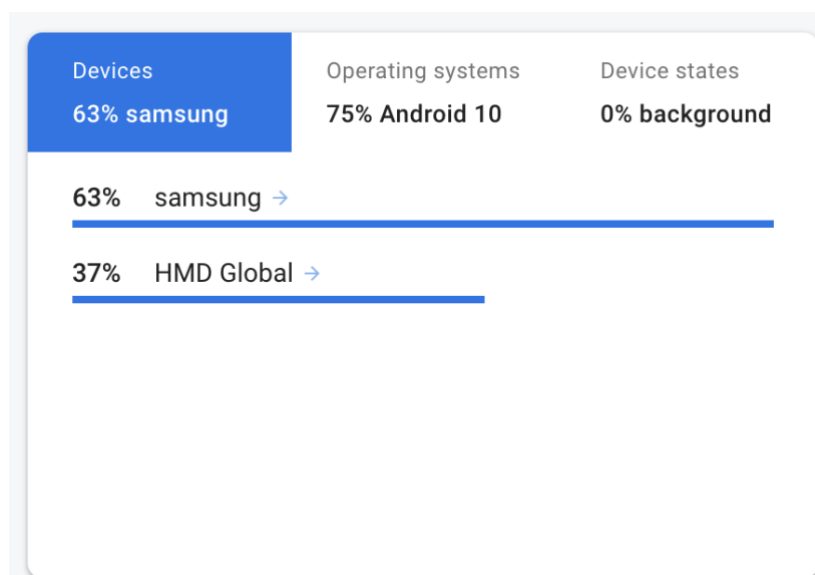


FIGURE 5.4 CRASH EFFECTED DEVICES

Analytics

Google Analytics provide the complete analysis about the application usage. Through this tool can be identify the application use time, Event count, Number of users and specially the usage of each screen of the application.

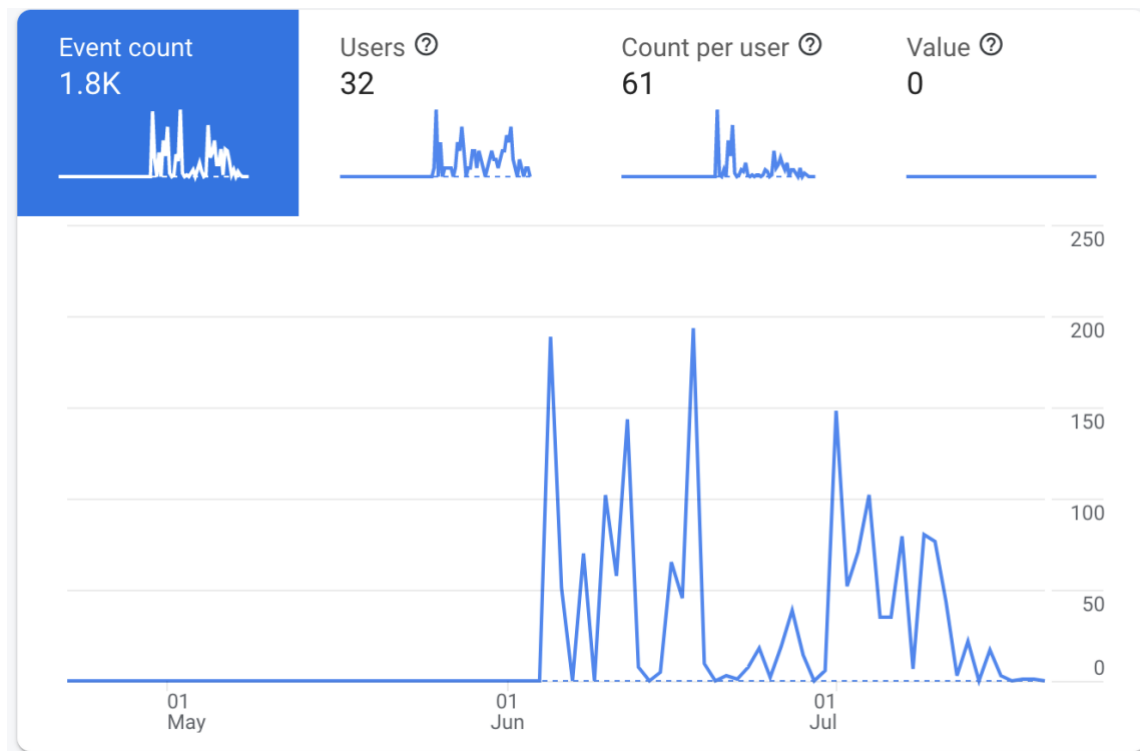


FIGURE 5.5 SCREEN VIEW

User engagement > Screen class				
Screen class	% total		Avg. time	
MainActivity	57.7%	-	1m 45s	-
LoginActivity	25.68%	-	4m 50s	-
CartActivity	10.19%	-	1m 23s	-
AddUpdat...Activity	2.25%	-	0m 40s	-
SalesReportActivity	1.46%	-	2m 48s	-
RegisterActivity	0.64%	-	0m 33s	-
SplashActivity	0.57%	-	0m 02s	-
SettleActivity	0.53%	-	0m 45s	-

FIGURE 5.6 USER ENGAGEMENT OF SCREENS

Referring Figure 5.4, graph gives the details about the event counts, user count and count per users based on the screen view. And in Figure 5.5 give the details about the duration of usage of each screen.

Performance

Provides an understanding of how long it takes to monitor firebase performance **an** application for completing a specific task. By default, it is measured automatically **application** attributes of the life cycle, such as the time interval When the application is opened and when it responds

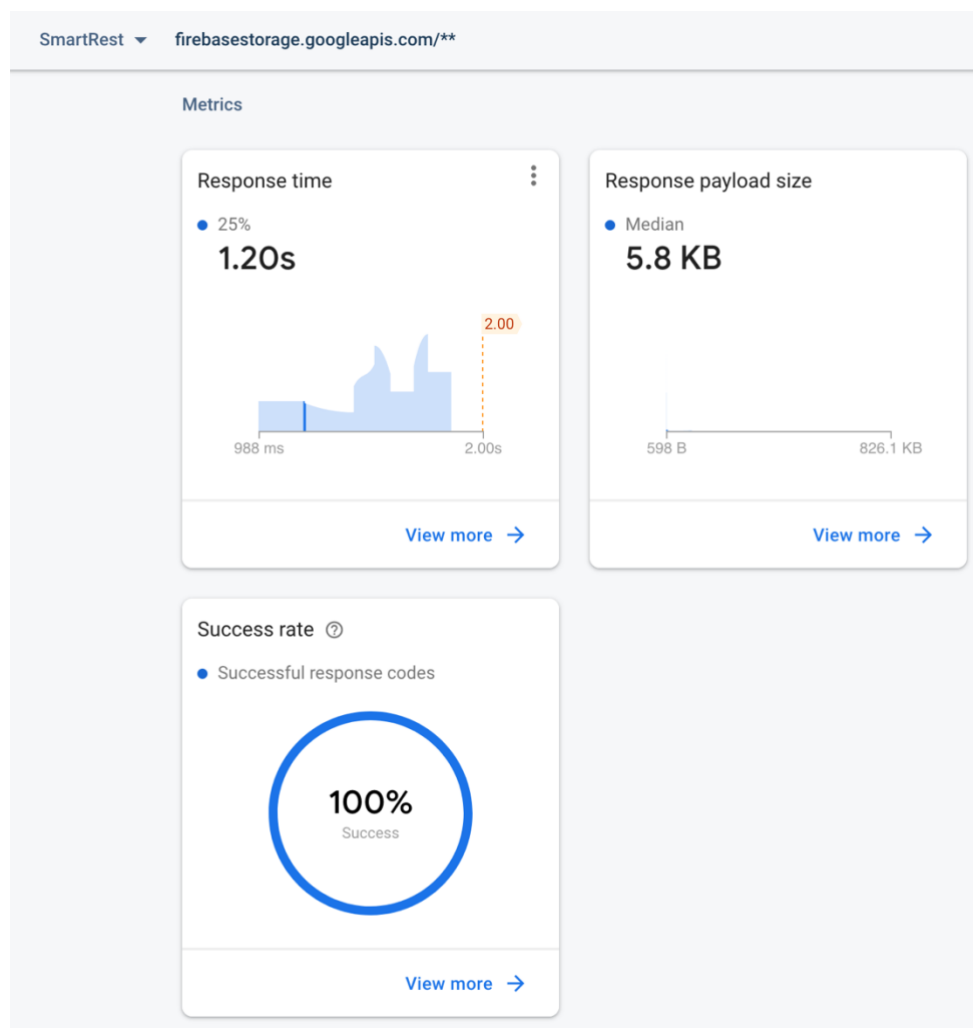


FIGURE 5.7 FUNCTIONAL EVALUATION

Referring Figure 5.6, show the summary about the response time, Response payload size and summary of success rate.

Chapter 6 Conclusion

7.1 Introduction

The previous chapter explained the overall evaluation process of the system, and this chapter describes the conclusions drawn from those evolutions and future new developments and system enhancements.

7.2 Project Limitations

The system has several shortcomings. Complete system based on Android, if someone not prefer to use Android, they are having a trouble with this system. Specially people who does not have Android device, they cannot use the Mobile application. When it comes to the Android device specification there are large number of different Android device brands and models. Under the Smart Restaurant project, it has tested only for few device brands and models. As an example, Tablet Applications have tested for Samsung Galaxy Tab J, Dialog TabPro and Pixel C tabs only. However, it currently needs to be tested with other brands as well to make sure the system is compatible with the majority of the devices.

7.3 Difficulties encountered

The main difficulty faced during the project was setup developer environment. Android Studio heavy IDE and it needs much better processor and ram to run smoothly. And also 4 Android application develop parallelly meant it need much better environment. Used MacBook Pro laptop with 16 GB ram and the development environment could prepared on those resources.

7.4 Benefits of Implementing This Project

The main purpose of this project was to make easy the lives of the participants who gather around the restaurant domain. It was primarily aimed at a dining inside the restaurant option which not clearly addressed in the existing systems. One of the other objectives was reduce the interconnection of each person during restaurant process. Given the current status of the world, this would be the perfect solution to slow down the COVID 19 virus and continue the dining process in restaurants. And also, this kind of a solution can be reducing the usage of the papers. Menu, invoices can be removing by using this solution. And the other benefits are going to be small restaurants can reduce the employee count, Customers can have a real time experience and the managers can view the progress in real time.

7.5 Potential future work

1. Firebase can be use with any developer platform. Do the same development for the iOS platform is going to be wide the range of users.
2. Currently the system has only the cash payment feature. In future paling to integrate the online payment methods. Then it can be more reduced the human interaction.
3. In the current system user cannot order something after terminate the order process without getting login back and creating a new order. In future planning to give the opportunity to the customers to resume the same order even it get terminated.

References

Datta.S, 2021. [Online]

Available at: <https://limetray.com/blog/restaurant-cash-register>

[Accessed 14 February 2021].

Uber, E., 2021. *Uber Eats: Food Delivery and Takeout: Order Online from Restaurants Near You*. [Online]

Available at: <https://www.ubereats.com/lk>

[Accessed 29 August 2021].

foodie365cloud, 2021. *India's Best Restaurant management software / POS Software / GST Ready*. [Online]

Available at: <https://foodie365cloud.com/>

[Accessed 29 August 2021].

tutorialspoint, 2021. *MVVM “ Introduction*. [Online]

Available at: https://www.tutorialspoint.com/mvvm/mvvm_introduction.htm

[Accessed 29 August 2021].

Android, 2021. *Activity : Android developers*. [Online]

Available at: <https://developer.android.com/reference/android/app/Activity>

[Accessed 29 August 2021].

Android, 2021. *Fragment lifecycle : Android Developers*. [Online]

Available at: <https://developer.android.com/guide/fragments/lifecycle>

[Accessed 29 August 2021].

Firebase, 2021. *Cloud Firestore Data model / Firebase Documentation*. [Online]

Available at: <https://firebase.google.com/docs/firestore/data-model>

[Accessed 29 August 2021].

Ceta, N., 2020. *Tallyfy*. [Online]

Available at: <https://tallyfy.com/uml-diagram/>

[Accessed 29 August 2021].

visual-paradigm, 2021. *What is Use Case Diagram?*. [Online]

Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what->

is-use-case-diagram

[Accessed 29 August 2021].

Datta.S, 2021. *Limetray*. [Online]

Available at: <https://limetray.com/blog/restaurant-cash-register/>

[Accessed 29 August 2021].

Odoo, S., 2021. *Odoo S.A.*. [Online]

Available at: <https://www.odoo.com/app/point-of-sale-restaurant>

[Accessed 29 August 2021].

Appendix A

EVALUATION QUESTIONNAIRE

Smart Restaurant

EVALUATION QUESTIONNAIRE

The objective of the questionnaire is to gather the information to build a cost-effective mobile based smart restaurant system. The information gathered through this questionnaire assesses the relevance and usefulness of the proposed system. So I welcome your feedback

Mark the appropriate answers with a 'X'.

1. Age group

16-20	
21-30	
31-50	
Above 50	

2. How do you rate your knowledge of mobile literacy?

An advanced user	
An average user	
Basic user	
Non-literate	

3. Have you used any mobile based restaurant system?

Yes	
No	

4. Prefer mobile based restaurant system compared to manual method?

Yes	
No	

5. Was the graphical user interface easy to use?

Yes	
No	

6. Level of satisfaction?

Very satisfied	
Satisfied	
Un-satisfied	
Very un-satisfied	

7. Any suggestions.

--

Appendicix B

User Manual for Table Tablet Applications

The user manual is designed to provide user guidance on use the table tablet application. This Application can use to any person who is coming to dine at the restaurant. Initially the restaurant staff should be able to register the tablet to the table by adding a table number. This feature should be hidden to the restaurant customers.

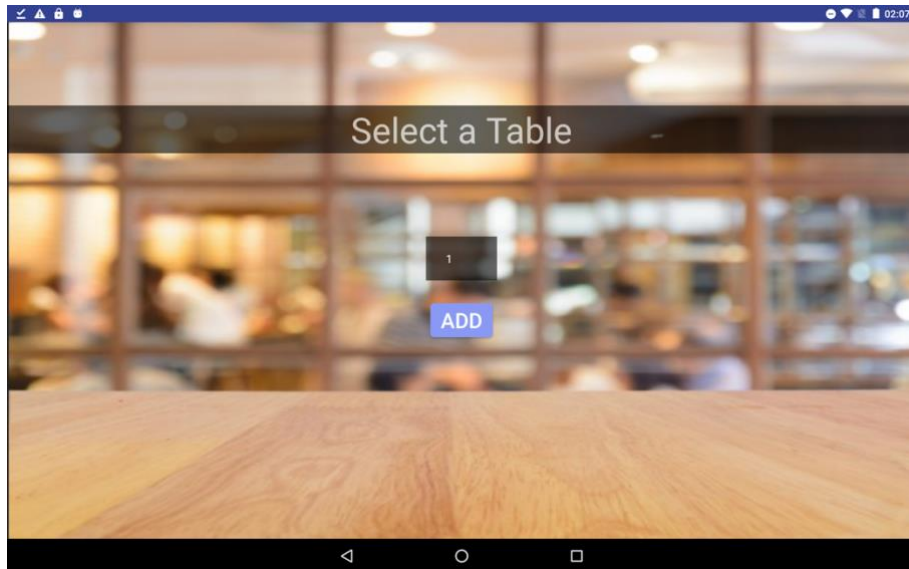


FIGURE B.1 TABLE REGISTER SCREENFIGURE

Referring Figure B.1, Staff member should be selected the table number and add it to the table registration.

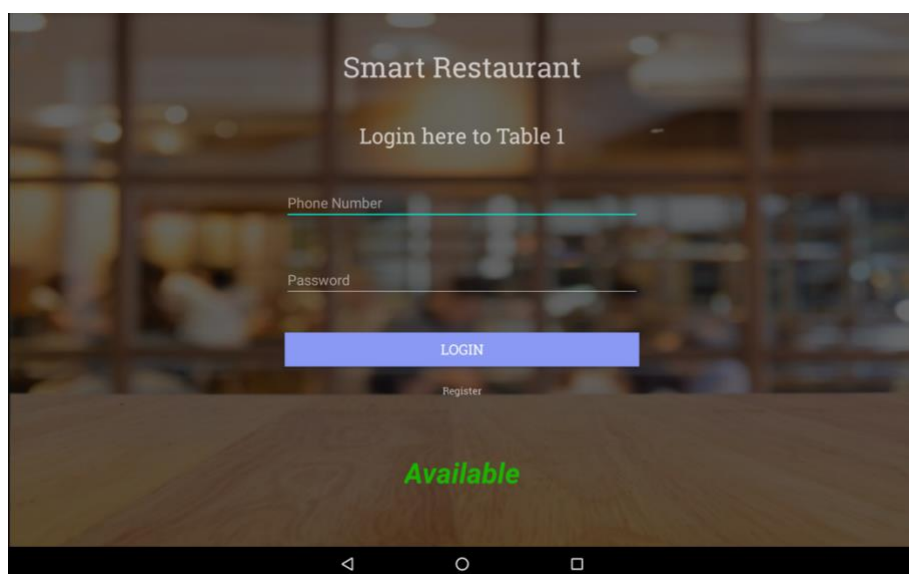


FIGURE B.2 LOGIN SCREEN

Referring Figure B.2, Show the login screen to the Table applications. If the table is available is shows Available. Otherwise, it shows Reserved.

If the customer is already registered customer, can directly login to the system. Otherwise, customer should go to the register screen for the self-registration.

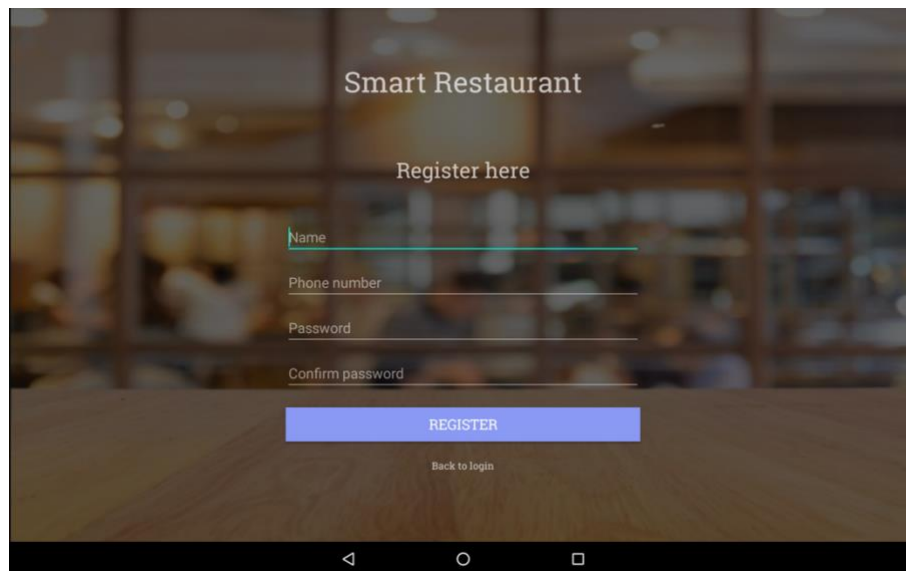


FIGURE B.3 CUSTOMER REGISTRATION

Referring Figure B.3, If the customer is new to the restaurant, should give these details and register first. It will be redirected to the main screen.

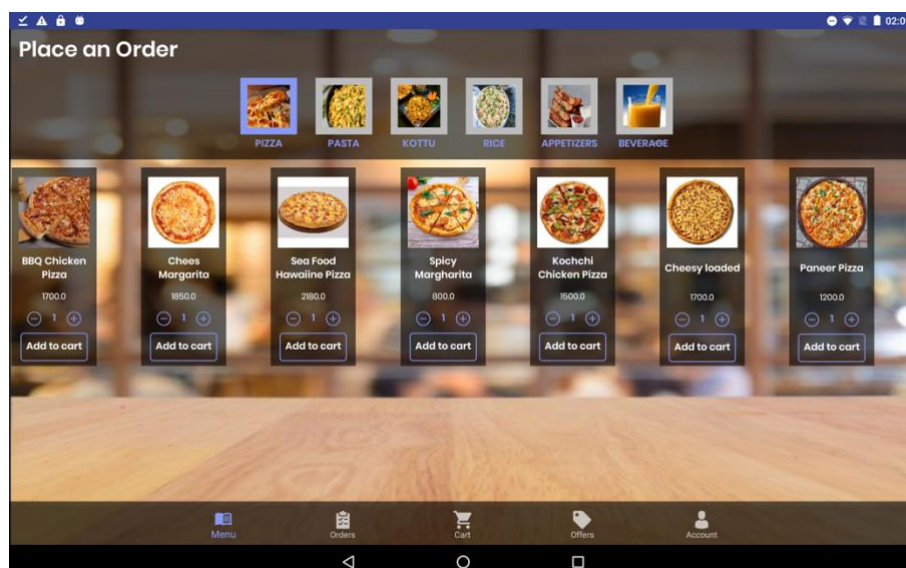


FIGURE B.4 MENU

Referring the Figure B.4, This screen shows the complete menu. User can select each category to filter out the food items by category. The status Place an Order meant user have to place a

order here. By clicking Add to cart button user can select the items to the cart. And also, user can change the count of the selected item by clicking plus and minus buttons of each food item.

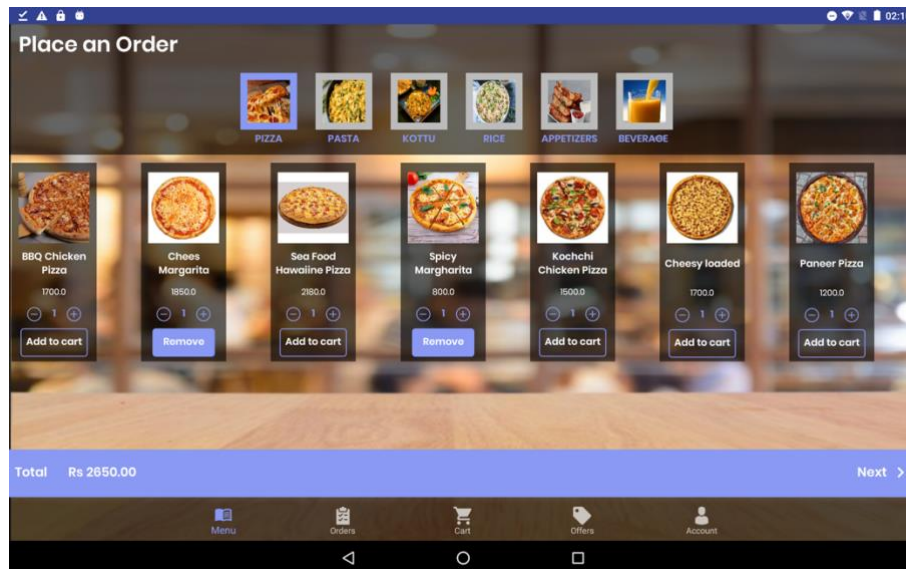


FIGURE B.5 SELECT MENU ITEMS

Referring Figure B.5, When the user selects at least one item to the cart it shows the next button with the current amount.

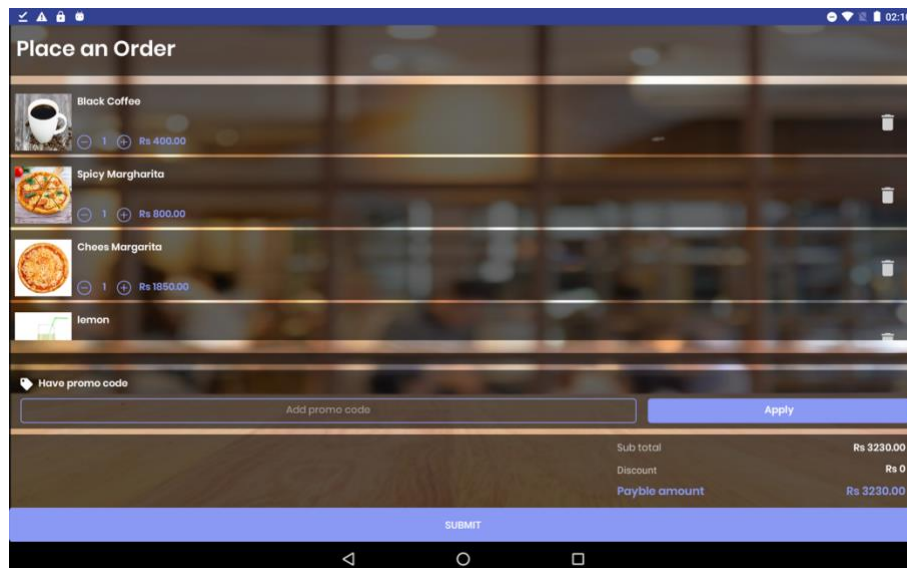


FIGURE B.6 CART

Referring Figure B.6, After click the next at Figure B.5 user redirect to this screen. This is the Cart screen. User can modify the order on here as well. When the order selection finalize user should select the Submit button.

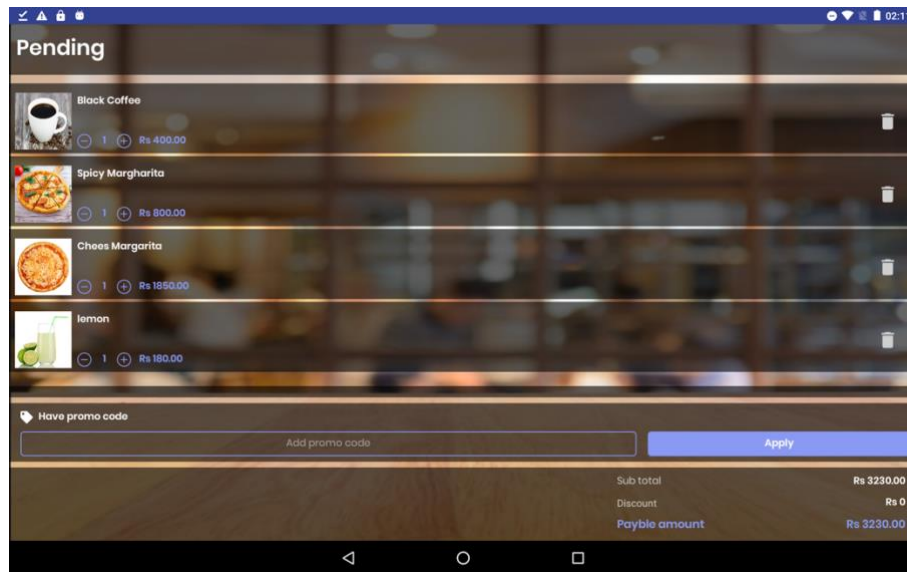


FIGURE B. 7 PENDING STATUS

Referring Figure B.7, after user submit the order status changes to Pending and button getting hide.

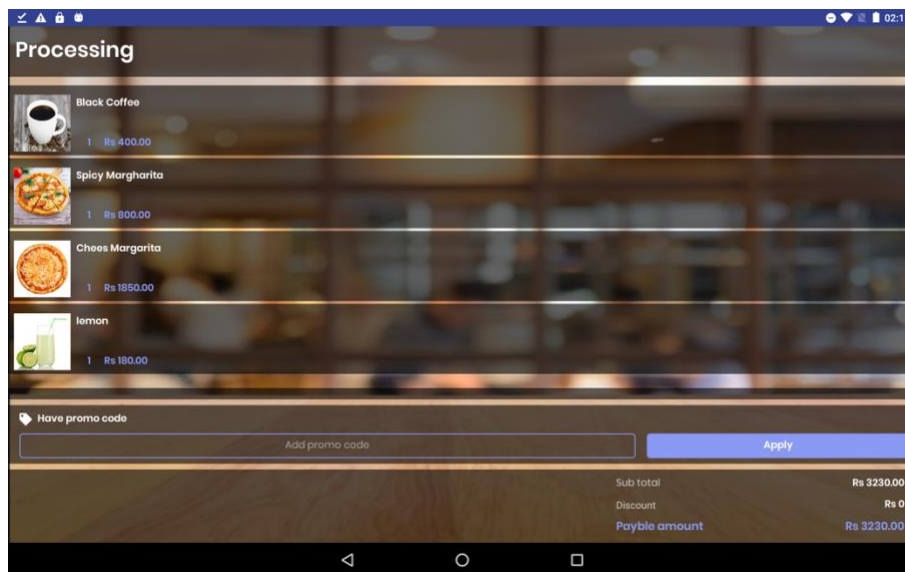


FIGURE B.8 PROCESSING STATUS

Referring Figure B.8, When the chef accepts the order, Order status changing to the Processing. That means the order has successfully submitted and the accepted by someone inside the kitchen.

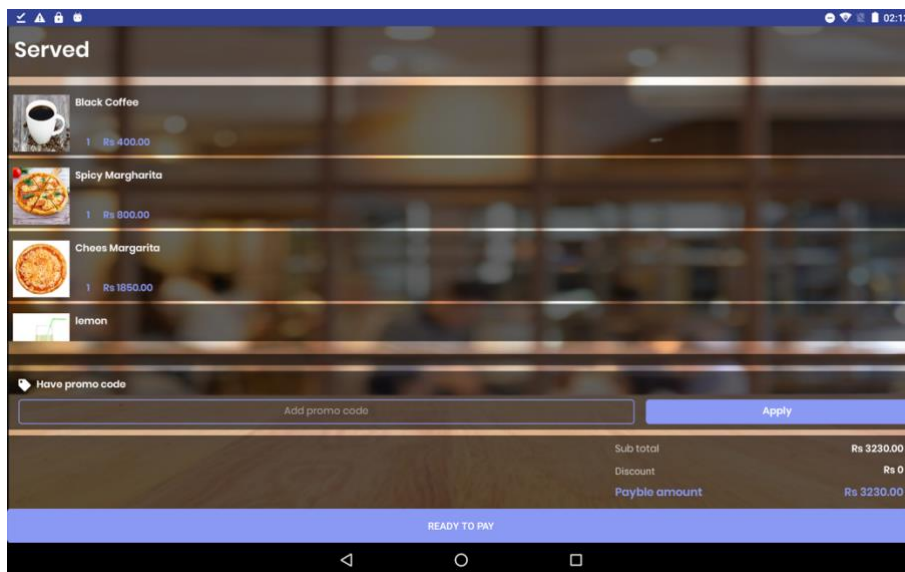


FIGURE B.9 SERVED STATUS

Referring Figure B.9, Kitchen has completed the order and it is serving to the table. After customer complete the dining, customer should click Ready to pay button.

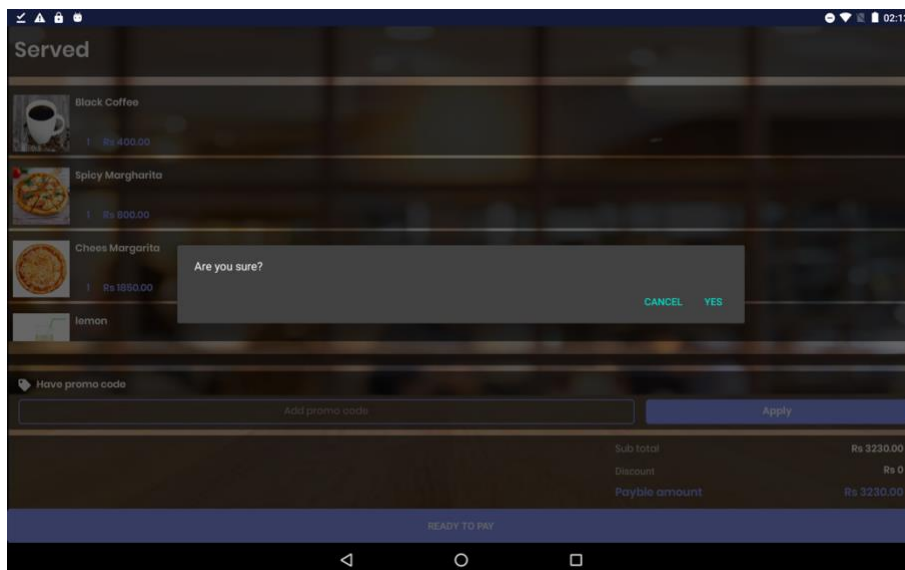


FIGURE B.10 CONFIRM THE TERMINATION

Referring Figure B.10, It is asking for the confirmation to terminate the order and request to bill payment.

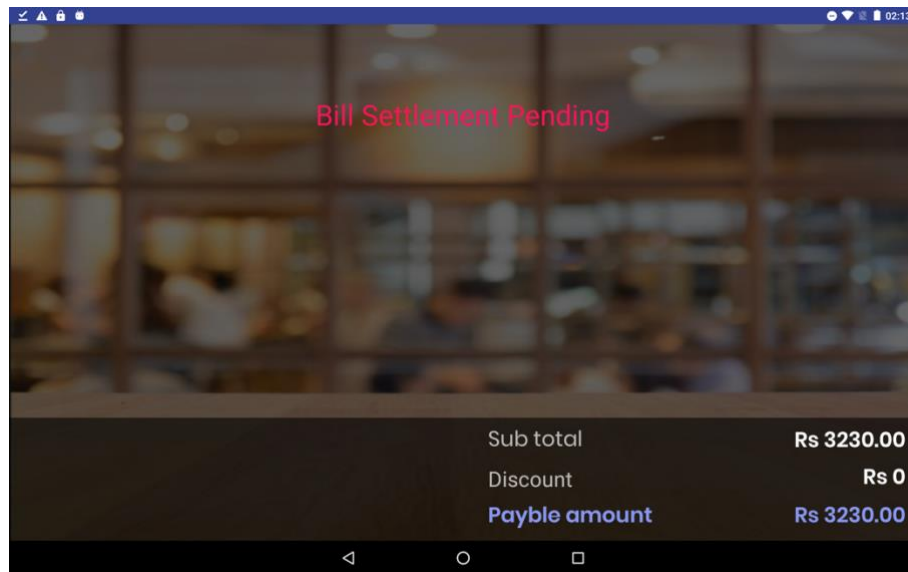


FIGURE B.11 BILL SETTLEMENT PENDING STATUS

Referring Figure B.10, Customer has requested to pay the bill. Until the bill getting settled this screen shows in the tablet.

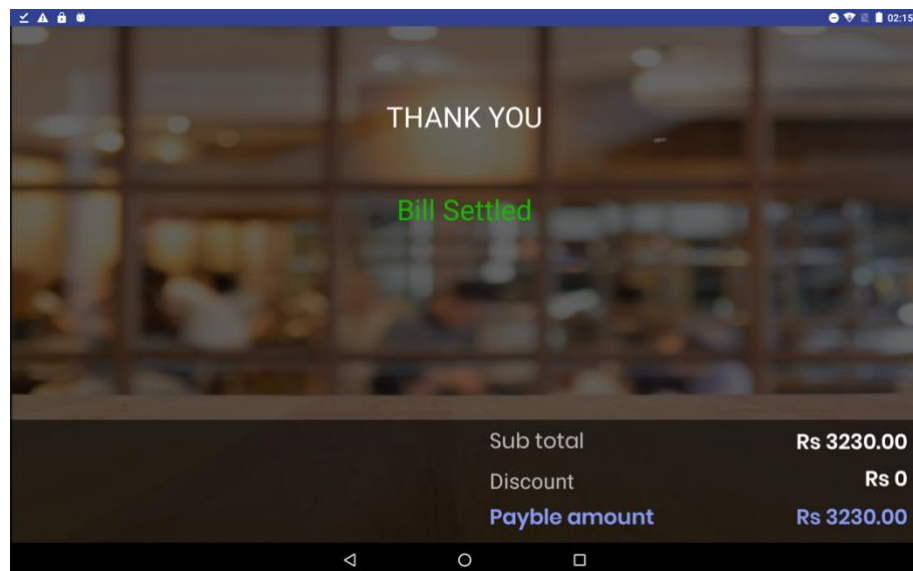


FIGURE B.12 BILL SETTLED STATUS

Referring Figure B.11, User has settled the bill and it shows this screen for 10 seconds. After 10 seconds this redirects to the login screen.

User Manual for Table Admin Applications

This user manual is designed to provide user guidance on use the admin application. This Application can be used by the super admin users and cashier staff. Depending on the user role, features may be enabled / disabled.

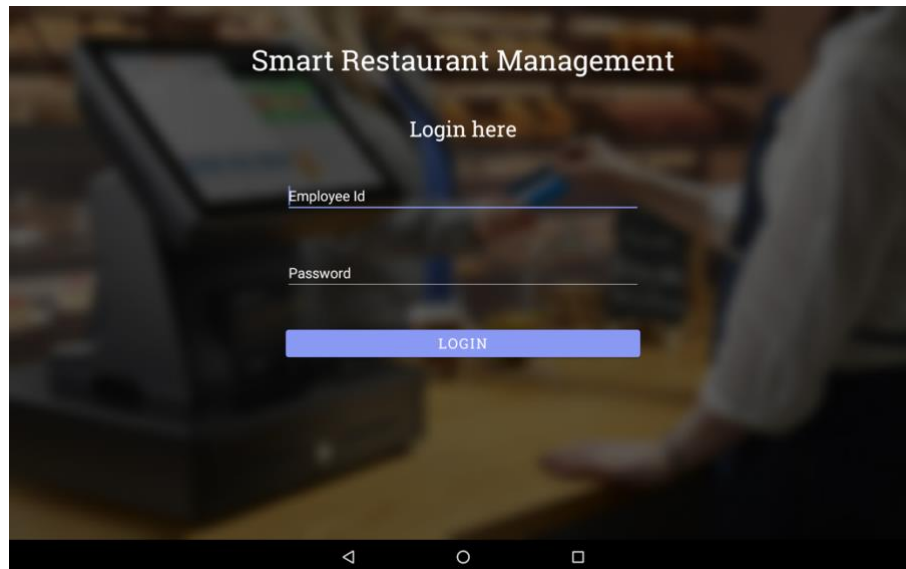


FIGURE B.13 USER LOGIN

Referring Figure B.13, User registered employees can login to the application by using the employee id and the given password.

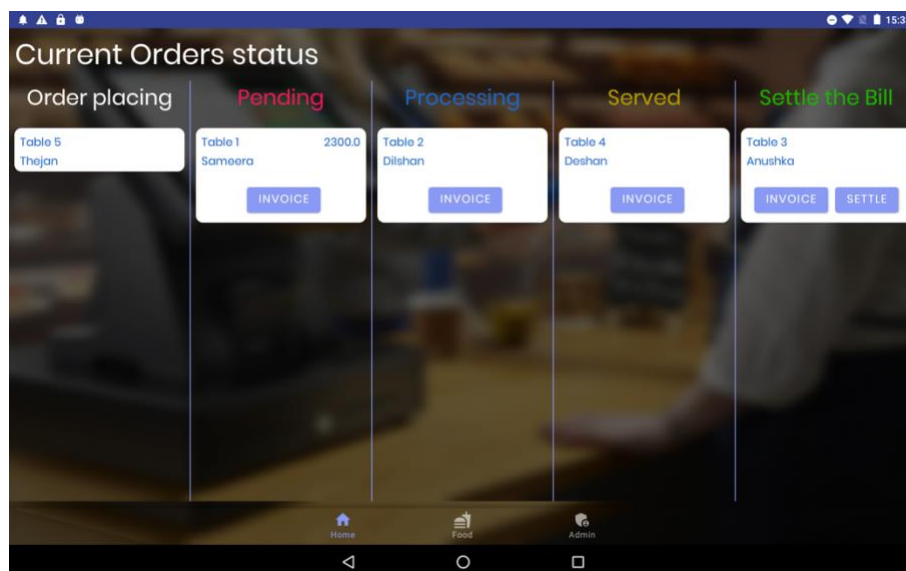


FIGURE B.14 ORDER STATUS

Referring Figure B.14, Cashier and the admin user can see the order status of each table. When the customer started to place an order in a table it appears in the order placing column. After customer submit the order, it moves to the Pending column. From it onwards user can view the customers invoice by clicking the invoice button in each order. In the last stage when the

customer settles the bill user can terminate the process of the order by clicking the SETTLE button.

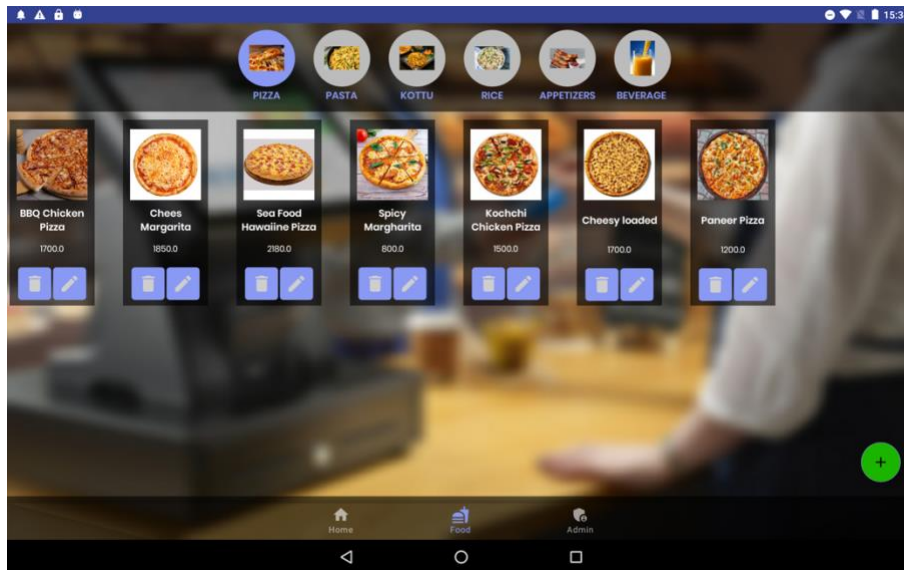


FIGURE B.15 FOOD DETAILS

Referring Figure B.15, Available food items can be access here. Only admin user able to add, delete and edit the items. Cashier can only view the items.

FIGURE B.16 ADD FOOD ITEM

Referring Figure B.16, Admin users can add a new food item using this form. Adding a image is mandatory.

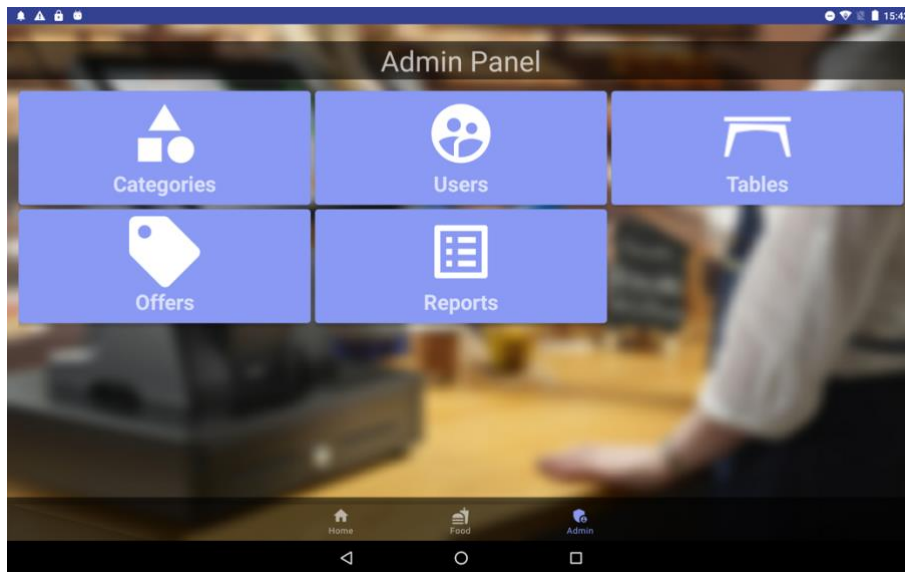


FIGURE B.17 ADMIN PANEL

Referring Figure B.17, Only these features can be accessed by the admin user. This tab is not visible to other users.

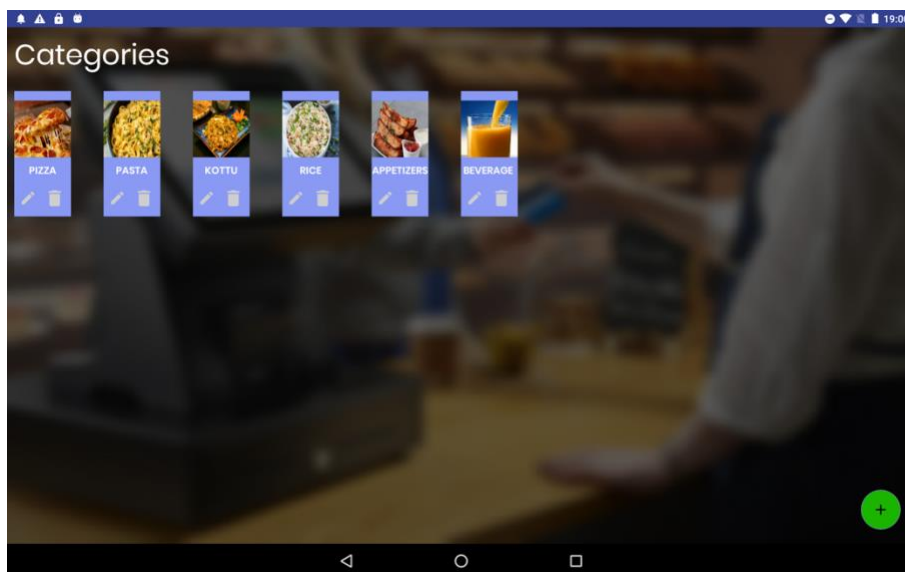


FIGURE B.18 CATEGORY DETAILS

Referring Figure B.18, Admin user can add, delete and edit the category details. Category name should be unique and adding a image to the category is mandatory.

Name	Emp Id	Role	Pin	Password
Thejan	1	Admin	N/A	123
Dhanushka	2	Chef	1122	123456
Kasun	3	Cashier	N/A	123

FIGURE B.19 USER DETAILS

Referring Figure B.19, Only admin users can be creating, delete and edit the user details. Specially the passwords and pin codes for the users are generating by the admin user. Pin codes are only available to the chef role. Pin code needs to access the kitchen application.

Table Number	Device id	Number of seats	Booking
1	N/A	4	N/A
2	N/A	2	N/A
3	190fe8e0b41914be	4	N/A
4	73cca70543a26df1	4	N/A
5	N/A	4	N/A
6	N/A	4	N/A
7	N/A	4	N/A

FIGURE B.20 TABLE DETAILS

Referring Figure B.20, Admin user can create, delete and edit the table details. Device which attached to the table should be register with a table number. When the device registering with the table number it is getting map with the table number and the device unique id.

User Manual for Kitchen Applications

The user manual is designed to provide user guidance on use the kitchen application. This application use by the users who has the chef role.

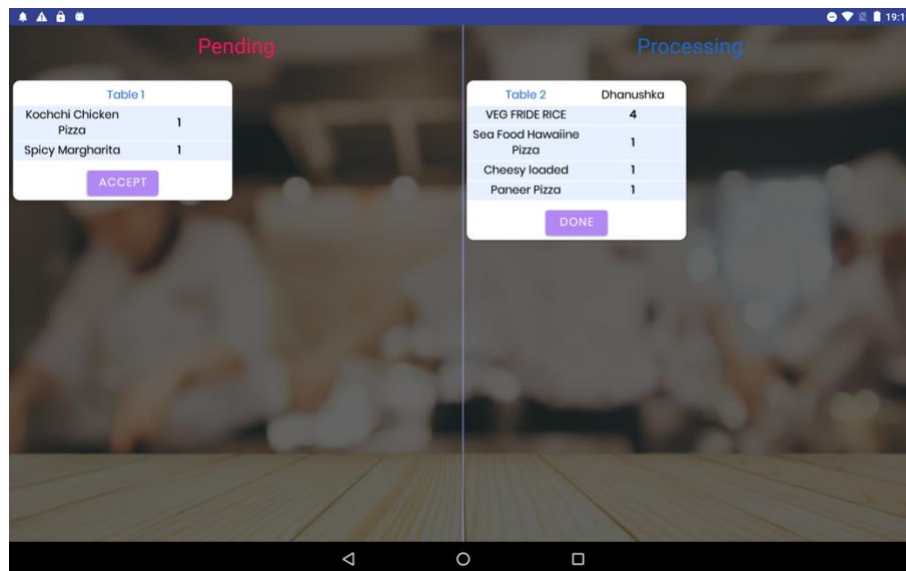


FIGURE B.21 PENDING AND PROCESSING ORDERS

Referring Figure B.21, When the customer place and order it comes to the pending column and one of the available chefs can be accept the order and it moves to the Processing column. Once the chef completed the order should click the DONE button to serve the order to the table.

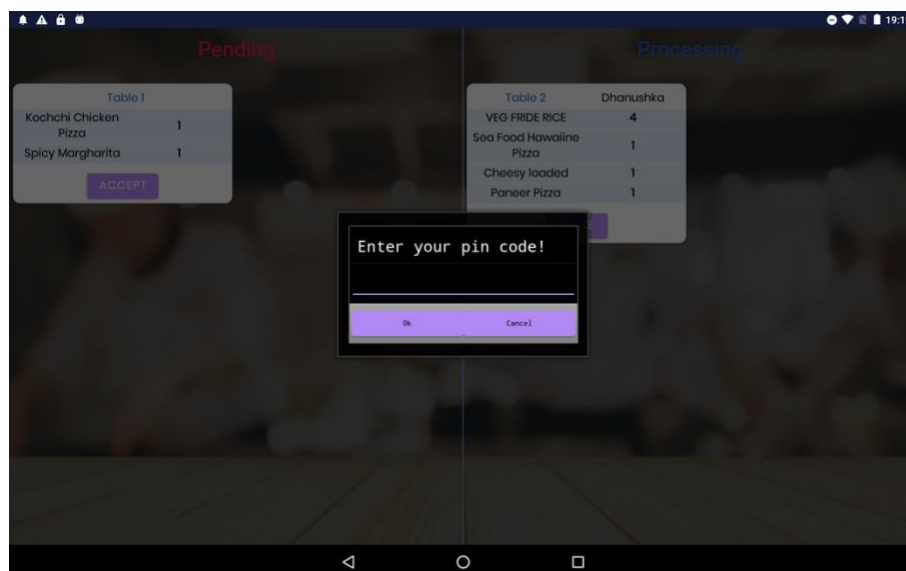


FIGURE B.22 ACCEPT AN ORDER

Referring Figure B.22, When the chef accepting the order should enter the pin number.

Appendicix C

Management Information Systems reports

Summary Report

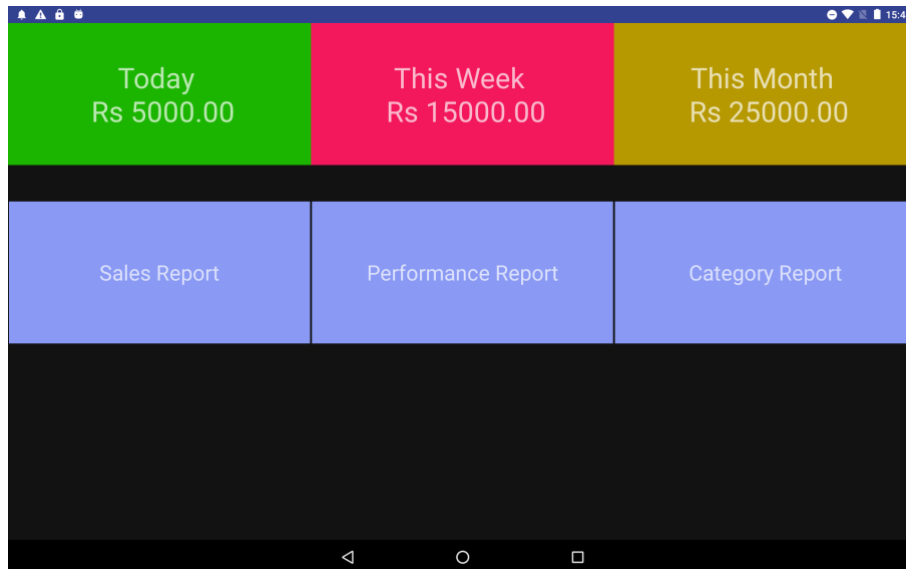


FIGURE C.1 SUMMARY REPORT

Referring Figure C.1, Report screen shows in this screen. It shows today's sale, this week sale and this month sale. Admin user only can access this screen and admin user has access to all the reports.

Sales Report

Today

This Week

This Month

This Year

Custom

Order Id	Food Id	Food Name	Cost	Price	Quantity
1	2	Mix Fride Rice	1000	1500	1
1	5	Orange Juice	500	750	1
1	3	Ice Cream	400	800	1
2	1	Vegitable Rice	800	1000	1
2	18	Orange Juice	600	1200	1

Revenue Rs 5250.00

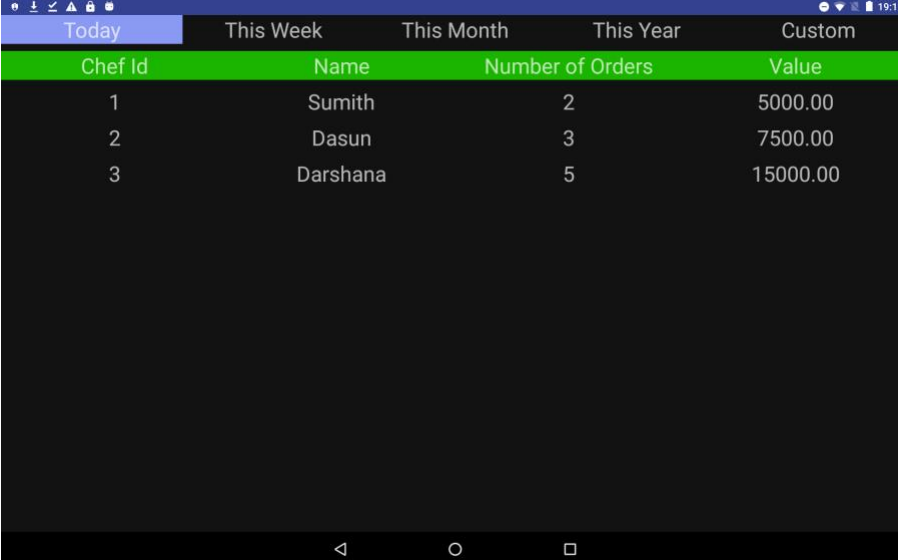
Cost Rs 3300.00

Profit Rs 1950.00

FIGURE C.2 SALES REPORT

Referring Figure C.2, Admin user can access the sales reports. It can be filter by today sales, this week sales, this month sale, this year's sales and custom date interval. And also, from this report user can see the revenue, cost and the profit under those filters.

Performance Report



Today This Week This Month This Year Custom			
Chef Id	Name	Number of Orders	Value
1	Sumith	2	5000.00
2	Dasun	3	7500.00
3	Darshana	5	15000.00

FIGURE C.3 PERFORMANCE REPORT

Referring Figure C.3, Admin user can access the performance report. It can be filter by today performance, this week performance, this month performance, this year's performance of each chef and custom date interval. Admin user can track the performance of each chef through this report.

Category Report



Today This Week This Month This Year Custom			
Category Id	Name	Number of Orders	Value
Cat 1	Rice	5	1400.00
Cat 2	Appetizer	2	800.00
Cat 3	Pizza	3	3200.00
Cat 4	Bevarages	10	7200.00
Cat 5	Kottu	4	2200.00

FIGURE C.4 CATEGORY REPORT

Referring Figure C.4, Admin user can access the category report. It can be filter by today, this week, this month, this year of each category and custom date interval. Admin user can get an idea about each category and identify the fast-moving categories.