RESTAURANT MANAGEMENT SYSTEM

SamihaTahsin
2015-2-50-008

Taslima Yasmin Tarin
2015-2-50-022

A project is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Information & Communications Engineering

Supervisor
Dr. Mohammad Arifuzzaman
Assistant Professor
Department of Electronics & Communications Engineering
East West University

Department of Electronics & Communications Engineering
EAST WEST UNIVERSITY
Dhaka-1212, Bangladesh
September, 2019

Declaration

We, SamihaTahsin and Taslima Yasmin Tarin, hereby, declare that the work presented in this project is the outcome of the investigation perform by us under the supervision of **Dr. Mohammad Arifuzzaman**, Assistant Professor, Dept. of Electronics and Communication Engineering, East West University. We also declare that no part of this project has been or is being submitted elsewhere for the award of any degree ordiploma.

Countersigned

…….

Dr.MohammadArifuzzaman

Supervisor

Signature ofStudent

…….

(SamihaTahsin)

ID:2015-2-50-008

…….

(Taslima Yasmin Tarin)

ID:2015-2-50-022

Letter of Acceptance

This project entitled "**RESTAURANT MANAGEMENT SYSTEM: A E-commerce Site For Fast-food Products Where Customer Can Connect to a Restaurant With Its Menu Through Mobile"** submitted by SamihaTahsin (ID:2015-2-50-008), Taslima Yasmin Tarin (ID:2015-2-50-022) to the Department of Electronics and Communications Engineering , East West University, Dhaka, Bangladesh is accepted by the department impartial fulfillment of requirements for the Award of the Degree of Bachelor of Science in Information and Communication Engineering on Summer,2019.

Supervisor

……..
Dr. Mohammad Arifuzzaman
Assistant Professor
Department of Electronics and Communication Engineering
East West University

Chairperson

……..
Dr. Mohammad Moseeur Rahman
Associate Professor and Chairperson
Department of Electronics and Communication Engineering
East West University

Abstract

Portable based exchange are among a few of the quickest developing zones in Data Innovation nowadays. The development presents assorted opening more especially in supply and dissemination of products and administration beneath the more extensive umbrella of m-commerce since numerous buyers would advantage by having implies of making orders utilizing their portable gadgets. A major common problem facing typical ordering systems in restaurants is that customers have to depend on waiters for getting the menu, order to them which makes things a little bit of harassment as they had to wait for waiter's longtime sometimes. In order to solve this problem we have developed a mobile application for restaurant management and user both where no waiter service exists in between customer and restaurant while ordering food.The app will offe the ability to manage accounts,orders and activitie related to it. It should offer good GUI for easy use and a faster way to complete user requirements. The app will communicate with restaurant web applications.

Acknowledgement

As it is true for everyone, we have also arrived at this point of achieving a goal in our life through various interactionswithandhelpfromotherpeople.Wewouldnotliketomakeefforttofindbestwordstoexpress our thankfulness other than simply listing those people who have contributed to this project itself in an essential way. This work has been carried out in the Department of Electronics & Communications Engineering at East WestUniversity.

First of all we would like to express our deepest gratitude to the almighty Allah for his blessings on us. Next, Our special thanks go to our supervisor Dr. Mohammad Arifuzzaman, Assistant Professor,DepartmentofElectronicsandCommunicationEngineering,EastWestUniversitywho give us this opportunity, initiated us into the field of " Restaurant Management System" , and without whom this work would not have been possible and their encouragements, visionaries and thoughtful comments and suggestions and unforgettable support. We would like to thank all friendswhogiveexcellentcollaborationduringperformanceevaluationstudiesforoverallsupport and helpful suggestions in solving tricky technical problems. Last but not the least, we would like to thank our parents for their unending support, encouragement andprayers.

There are numerous other people who have shown me their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our heart and hope to find a more appropriate place to acknowledge them in the future.

SamihaTahsin

Taslima Yasmin Tarin

August,2019

List of Figures

Table of Contents

Chapter 1

Introduction

We dwell interior the age of information advances. It's energetic our lives and making it less demanding. Since we've got found the adaptability of innovation and grasped it as a serving to device, we've got a bowed to do to create it encourage valuable. Innovation isn't remaining the same; it's energetic and up about day by day.We are preoccupied with the wish to make new technology better. There was innovative intrigued in versatile gadgets and tablets after the trend of PCs & laptops.In 2014, the amount of mobile users reached the same amount as desktop users(almost 1,700 milion users) and that number is still increasing[1].Keep with a figure of the data reason entry Statist, since 2015, the amount of dispatched tablets are going to to be bigger than the sum of transported tablets and portable workstations on [2]. Companies are realizing this modern drift and beginning to consider the versatile exchange over ever some time recently

1.1 Motivation

Within the past five a long time, there are huge deluges of customers/guests derogative fast-food eateries around the world altogether interior the created nations. On the elective hand, clients are tired of holding up on the line for long hours before being served [1.On the advertise, there are many devices with utterly totally diverse coding system bundle and equipment, making the duty of the code bundle engineers loads of unpropitious. One among these steps was to trade desktop application. An advantage of internet applications is that they're doing not place confidence in the platform where they run.[2] There's to boot a simple because of update applications for all users. Another advantage is that everybody computations are resolved on a server; therefore it'll run on devices with outdated hardware nonetheless. Keep with associate increasing kind of mobile and pill users, mobile applications are convenient.

1.2 Scope of the project

Due to the focuses in time of this extend we tend to tend to need to slim the scope. The scope of this extend is to create Relate in Nursing exemplification of application that's prepared to attach with the take separated server and manipulate with information.[3] The exemplification have to be have input interface that's prepared to post to the server any entered data and posting interface where certain information are advancing to be recover from server and shown to the client. Collectively at interims the scope of this venture to speak to nuts and bolts of robot advancement and one of the internet administrations like take separated.

1.3Project Overview Statement:

The aim of this project is to make an android food app as a mobile application. So we have made a food app which we have named it "SPICE JAR". It is to be used by the common people as customers who wants to order in restaurant with no delay and offers them the ability to manage accounts and order foods by sitting in the restaurant with the help of scan a particular qr code which every

restaurant will have their own. There will be no waiters to serve restaurant menu or taking orders from the customers. It will be very convenient for the customers and will be beneficial for the restaurant management system.

The idea of our app “SPICE JAR” comes from one of a coffee shops in Dhanmondi, Dhaka city. That shop use their own code by which customer can see their menu by using the given code but cannot order from it. So we give a thought and think that if we can make a mobile app by which people can order any type of food directly and It will be worked on devices with Android and Apple operation systems .and offered the same possibility as a web application, which is the aim of this project. It will be logically designed and offer the ability to accomplish any task quickly and properly. It will also offer understandable GUI for any user. Our appwillbeofferedonlyforAndroiddevices, specificallyfocusedtotablets.Theapp will communicate with restaurant web application over REST API. The result of this project is the app, which would be deployed and available for use.Ourapplication and customer boththese wouldbe:

- Create an account of their own
- Make an order for own account
- By scanning QR code user can get the restaurant menu
- Order directly
- Restaurant will confirm the order
- Both will get notification

Chapter 2

Literature Rereview

2.1 Mobile Technology Usage in theWorld

The increment among the inconstancy of versatile cell phones inside the world has been fabulous. The chart underneath appears supporter development between 2005 and early 2013, in line with ITU figures. The 6.8 billion endorsers are drawing closer the seven.1 billion world populace. the standard world infiltration stands at ninety six.2%, in line with the ITU [4].

2.2 Mobile App Usage in the world

With over a combine of 7 billion smartphone clients over the world, there is no astonish that the versatile app exchange is thriving. App utilization and smartphone infilteration are still developing at a delicate rate, with none signs of speed down inside the unsurprising future.

2.3 Restaurant/Food Apps usage have increased

As of late, nourishment and eating house apps have seen marvelous development among a center group of onlookers, joined togather Countries office decide as nourishment darlings. Indeed, the utilization of nourishment and eating house apps have hyperbolic by seventieth since 2014, to its current standing of thirty fourth.

Figure 1: Mobile Food App Usage

2.4 Existing Solutions for Ordering Commodities

Paper based completely requesting frameworks are wide utilized in eateries as of now. Papers are utilized in eateries for showing the quality nourishment menus, taking down the customer's arrange and putting away the customer's orders. The impediments of paper essentially based framework are that papers will get basically broken by recolor marks, they will be crushed all through fires or mishaps, are awkward to handle and upgrade changes or will ordinarily float. Consequently, time and cash are squandered. As antiquated menu cards are paper essentially based, any changes ought} to be made inside the menu require distribution of the total menu card, coming about in wastage of paper, time and cash [6]. On tall of that, reason of Deals frameworks are utilized in eateries wherever a organize of cashiers and server terminals ordinarily handle nourishment orders, transmission of orders to the room and intuitively charge posting to visitor profiles.Reason of Deals information are regularly outside to bookkeeping and nourishmen t cost/inventory program bundle bundles and thus the frameworks too can create numerous administration reports. The biggest impediment of this method is that after there's a web blackout a private can't get to the framework making them result to the manual framework [7]. Customers take note it wearing and time strongly to line in eateries for long hours as they expect their orders to be prepared. It's furthermore a genuine issue for benefactors Joined together Countries office arrange on-line to look for out confusing interfaces that they will scarcely utilize or perceive. Finally, openness to boot a challenge confronting a few eateries since with the sharp increment of buyers requesting on-line having Associate in nursing requesting framework that's not versatile neighborly might have a negative affect on the ultimate sales. In arrange to defeat these challenges, the implies forward is to create a versatile application that empowers clients to put orders from the consolation of their possess homes, set the time they require to select the arrange and for those that need to eat from the structure, it gives them a chance to arrange a table.

2.5 Related Work

We inspired to make this mobile food application by researching some of these food application.

Food Panda:

Lunched on Nov nineteenth, 2013 in People's Republic of Bangladesh, Food Panda is a superb on-line food delivery system. individuals will order differing types of food from their partnering restaurants, and that they are going to be delivered at your home for free! individuals pays as presently as they get the food delivered. No MasterCard is needed to order food, no advance payment.

HungryNaki:

Hungrynaki.com could be a Bangladeshi on-line Food Ordering and Delivery Service launched in 2013 to deliver your cravings at people's doorsteps.

Uber Eats:

Uber Eats, the nourishment conveyance app by ride-hailing company Uber, is prepared to dispatch in Bangladesh in April 2019, with Dhaka as its 1st town. the same as the Uber rides app, Uber chow may be a nourishment conveyance app that makes a difference bring nourishment to customers.

Pathao Food:

Pathao Food may be a new service from ride-sharing company Pathao. Anyone will order foods from any close restaurants through Pathao food service.

All these food apps are for home delivery. Though they are very popular in our country but sometimes we suffered a little bit for traffic jam, not well-behaved service man or sometimes connectivity delay while ordering home delivery. Among all these food delivery apps The app that we have created is restaurant and user based where user can get the menu by scanning a specific qr code and can also directly order from that app sitting in the restaurant. That will make more time consuming for both user and restaurant as here no service man is needed.

Chapter 3

Techknowlogical Background & Analyze

Foundation hypothesis in this venture work serves as prophase for creating an application. That permits us to get it more compatibly the principles and innovations of Android advancement and can allow us thought almost assist structure of model extend.

3.1 Technological Background

3.1.1 Android Studio

Android Studio is the official integrated developmenmt environment (IDE) for Google's Android operating system,built on JetrBrains' IntelliJ IDEA software and designed specefically for android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

3.1.2 Flutter

Flutter could be a free, ASCII text file mobile SDK that may be want to produce native-looking android and iOS apps from identical code base. Being in beta for a jiffy, Flutter 1.0 was formally launched in December 2018. The main plan of Flutter revolves around widgets. In terms of recognition, Flutter is making good progress. Whereas Flutter had created it to the very best 100 package repos supported GitHub stars by the time unleash preview one was declared in June 2018, it's up among the ranks and is presently among the very best thirty. This, whereas not a doubt, can be a promising trend. Thousands of Flutter apps have created its because of app stores, among these the Alibaba app with fifty million users. Browse heaps of regarding what the Flutter team has to mention concerning their initial stable unleash and what's on their product roadmap for 2019.

3.1.3 Adobe XD

Adobe XD might be a vector-based apparatus created and uncovered by Adobe Iraqi National Congress. For coming up with and prototyping user expertise for web and versatile apps. The software package is on the showcase for macOS, Windows, iOS and android.it's primarily associate degree UI/UX coming up with application like Sketch in my opinion, it's bunch of tools like pen tool , choice tool etc. coming up with in XD is easy. It's supported the Adobe conventions. If somebody accustomed add Adobe Software's like Photoshop &Illustrator; they'll work with XD terribly simply and expeditiously. Similar to in creative person of Sketch you'll have multiple artboards. An enormous disadvantage is that It doesn't have layers (yet).

3.1.4 Firebase

Firebase may be a Backend-as-a-Service — BaaS — that begun as a startup and developed up into a next-generation app-development stage on Google Cloud Stage. Advance given engineers Relate in Nursing API that permits the blend of on-line chat utility into their websites.
Designers were design Advance to control application data like diversion state
in genuine time over their clients. Templin and Lee set to partitioned the chat framework and conjointly the elemental amount fashion that fueled it.[8] Firebase's to begin with item was the foot of operations essential amount information, Relate in Nursing API that synchronizes application data over iOS, Android, and web gadgets, and stores it on Firebase's cloud.

3.2 Analysis

3.2.1 Android Studio

For our project, we tend to are reaching to use android Studio as a result of the official IDE for mechanical man Application development [13]. Mechanical man studio offers an honest setting with logical structuring. Interface of applications are usually written as a XML file, by victimization graphic interface. The appliance would be running over degree mechanical man person or tested on a tool with android connected to our laptop. Mechanical man Studio offers many choices and makes the event of applications loads of intuitive.

Chapter 4

System Development Methodology

4.1 Development methods used

Developing applications for associate android device are often drained many various ways that. One possibility is to use the android software Development Kit (SDK) and make a native application. an alternative choice is to utilize mobile unifying technologies like Phone Gap or Xamarin which might create building applications for multiple in operation systems faster and easier to keep up.

There are somerenowned methods that are very popular to worldwide for software system development purpose. Those are described briefly below:

- Agile:

Agile computer code improvement might be an abstract framework for endeavor computer program designing comes.

- Waterfall:

The water demonstrate may be a sequent improvement approach, inside which improvement is seen as streaming unfaltering down (like a waterfall) through numerous stages.

- Scrum:

Scrum characterizes a flexible, all-encompassing improvement technique wherever a improvement group works as a unit to realize a standard goal. Extraordinary level of being programmed: Extreme Programming approach (XP) refers to relate spry computer code building technique. It had been made to dodge the occasion of capacities that aren't by and by required. It adapted toward the creation of a top-notch last item with no respect for visit changes in needs.

- Spiral:

The Winding technique amplifies the Waterfall show by including quick prototyping in an exertion to combine points of interest of top-down and bottom-up concepts.

- Extreme Programming

Extreme Programming approach (XP) alludes to an spry computer program building strategy. It was made to dodge the improvement of capacities that are not as of now required. It pointed at the creation of a top-notch last item with no respect for visit changes in necessities.

4.2 Rapid application development method

Rapid Application Development (RAD) Methodology is made to require the foremost advantage of the occasion code . It pointed to scale back the number of development required to form a item. RAD may be a condensed advancement strategy that produces a high-quality framework with moo speculation costs. It's potential since of the adaptability to rapidly adjust required things. It adapted toward giving quick results. RAD (quick application advancement) proposes that item will be created faster and of upper quality by:

- Using workshops or center groups to gather necessities
- Prototyping and client testing of styles
- Re-using computer program bundle parts
- Following a plan that concedes fashion improvements to future item version
- Keeping survey conferences and elective group communication informal

Chapter 5

System Design and Architecture

5.1 Analysis of System Design Requirements

A request or any kind of prerequisite is any work, limitation, or property that the framework ought to offer, meet, or fulfill so as to meet its reason. The objective of inquire about is to supply fundamental needs.

There are some requirements that are considered as functional requirements, as follows:

- The handle ought to have body capacities where the pc client is ready to update, erase and include modern things into the electronic nourishment menu.
- The handle need to have authorization capacities where the framework is ready to draw a line between clients and conjointly the director of the framework by creating user profiles with passwords.
- The process need to have associate external interface for supporters to position their orders and at steady time book a table.
- The prepare need to have a wicker container which can hold all nourishment things chosen by the customer for purchase.
- The handle ought to naturally calculate the common for arrange created.

There are some requirements that are considered as non- functional requirements, as follows:

- Prioritizes the essential capacities of the framework backed the method usage patterns.
- The method must be compelled to be able to hold information made inside the framework for academic degree extended time while not the data being adjusted by the system.
- Representation: The method needs to be compelled to have such reaction times as scholastic degree illustration the arrange set need to be sent right away to the house and a reaction have to be be sent to the supporter all through a} awfully most of 3minutes.
- The prepare needs to be compelled to be fetched- successful once it includes maintenance.

Chapter 6

System Implementation

6.1 Tools for System Development

This framework plan is part into 2 fundamental class's particulerly the shopper or client viewpoiunt, diverse one is that the structure administration viewpoint. The reply was implemented on a area premise giving a one of a kind common sense in each stage as stipulated inside the venture set up. The extend was upheld use Windows 10, Firebase, Adobe XD, Android Studio. By the time all the stages were total, all the functionalities required inside the reply were met.

6.2 System Implementation

6.2.1 Main Starting Design

Figure 2: Android Application Starting Interface

As this android app has two sides user & restaurant management, so at first we will described about the user interface side.

6.2.2 User Registration & Login

Figure3: Get started as user or restaurant

After started the app, one will select themselves from here as a user or restaurant admin If he or she is a user, then he or she will enter the user button. And then will go to the next page that is:

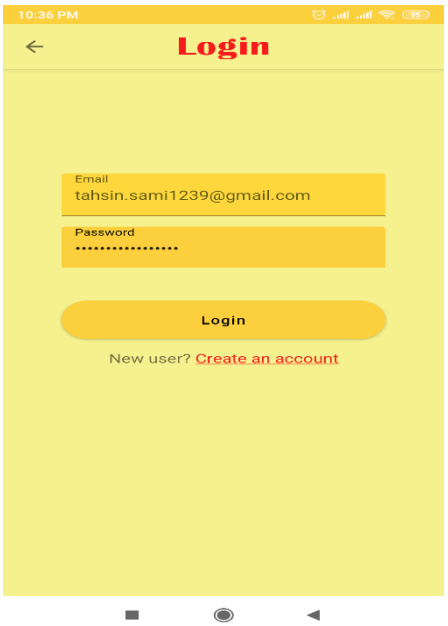


Figure 4: Login page

If user does not have profile, then they will to the option “create an account” and will get this page:

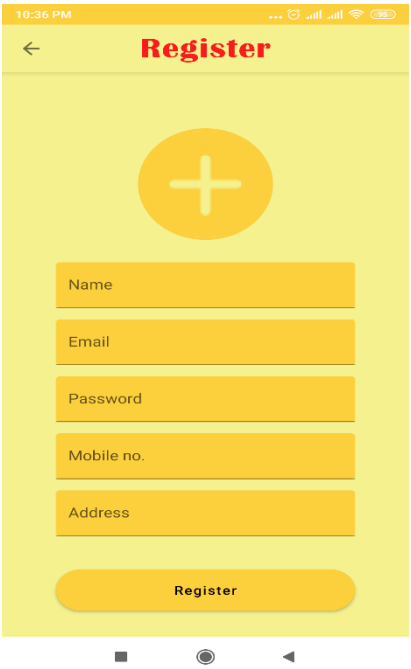


Figure 5: Registration page

6.2.3 After Login Process

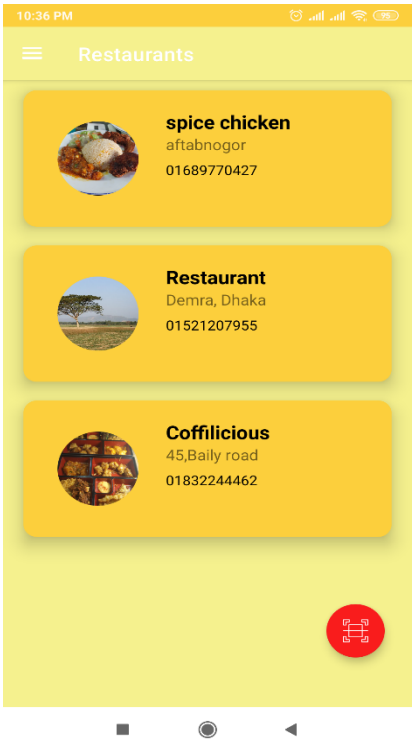


Figure 6: Restaurent list

After login, user can see the restaurent list from their account. There is a red button on the bottom of right side which one is for scan the QR code of any particuler restaurent.

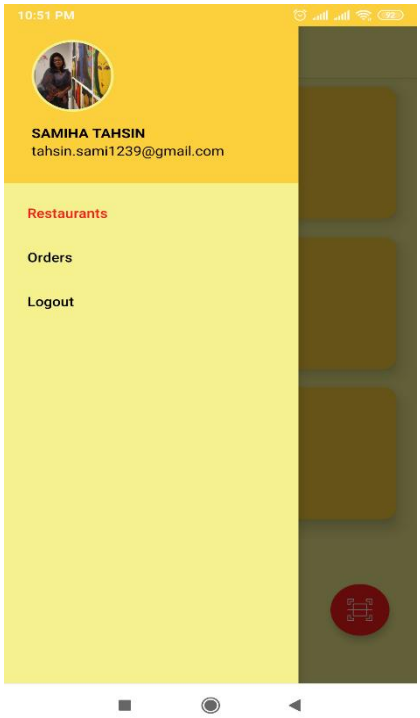


Figure 7: User menu interface

They can logout after complete their orders. It will seems look like:

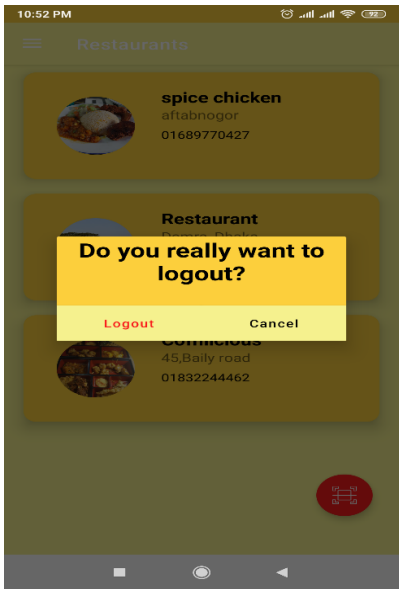


Figure 8: logout page

6.2.4 Restaurent Login & Registration

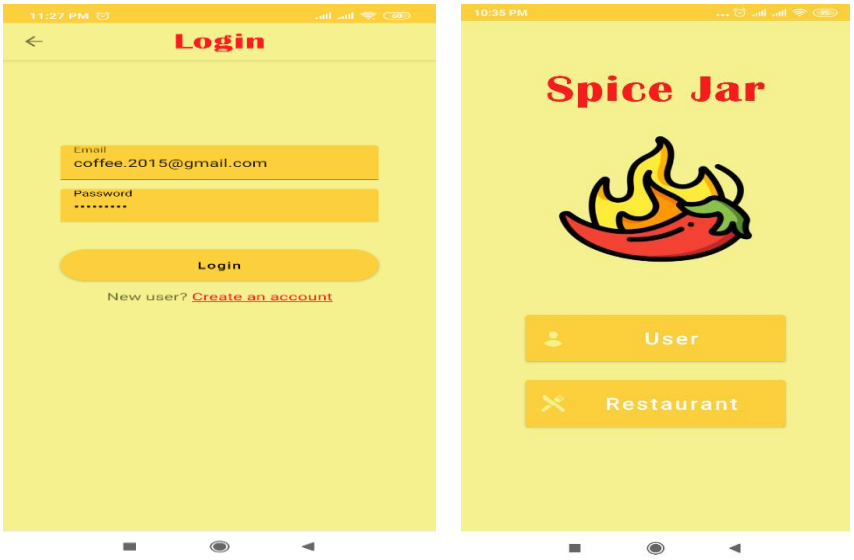


Figure 9: Restaurent Login Page

If any restaurent wanted to open any account for their restaurent they need to go the same option " Create an account" and will get the same registration page as previously given.

6.2.5 Restaurent Menu create,Add Items, Edit Items

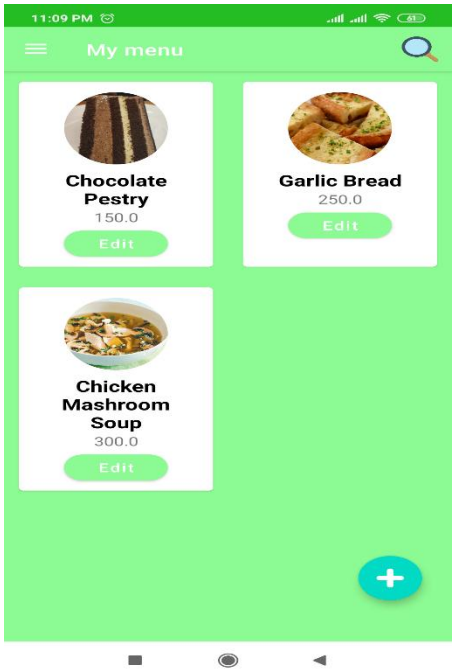


Figure 10: Restaurent's "My menu"

If restaurent want to add any item to their menu, they will use the plus button to add.

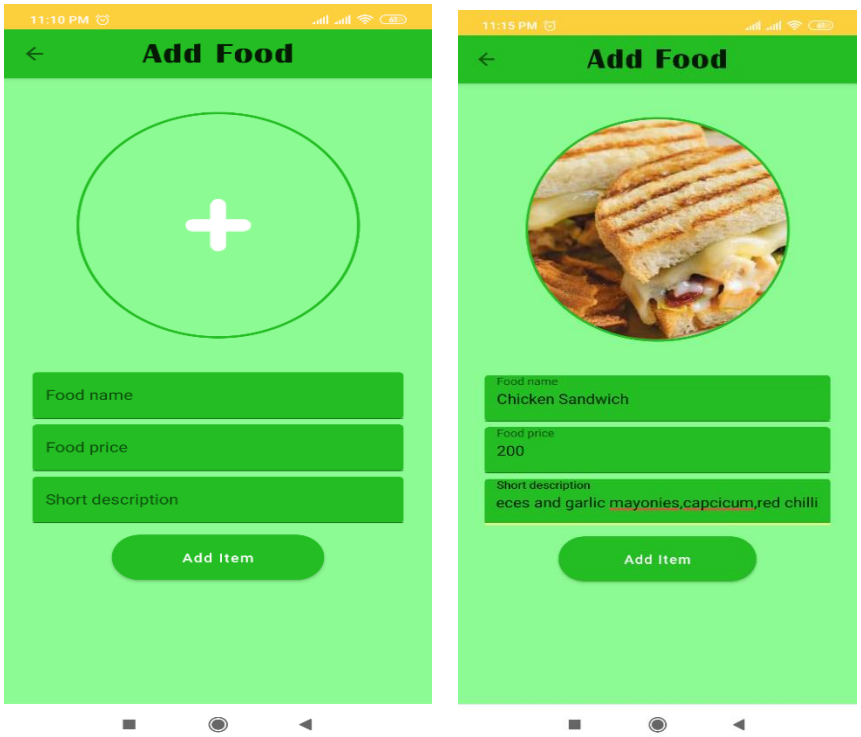


Figure 11: Add item to main menu

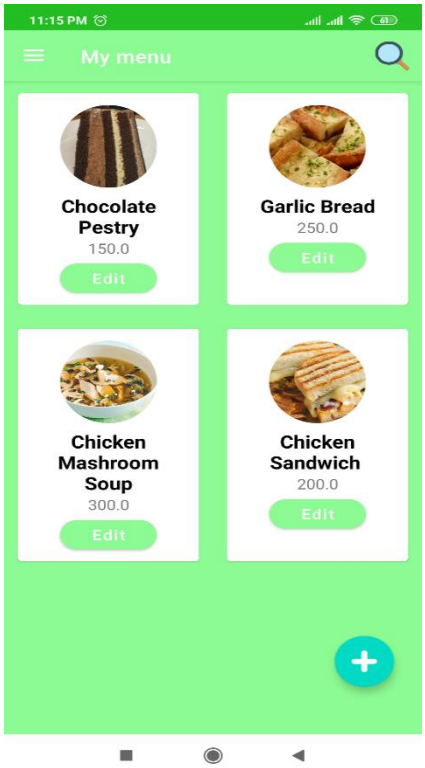


Figure 12: Updated menu after adding new item

Restaurent management can edit their item any item. This edit button will work then.

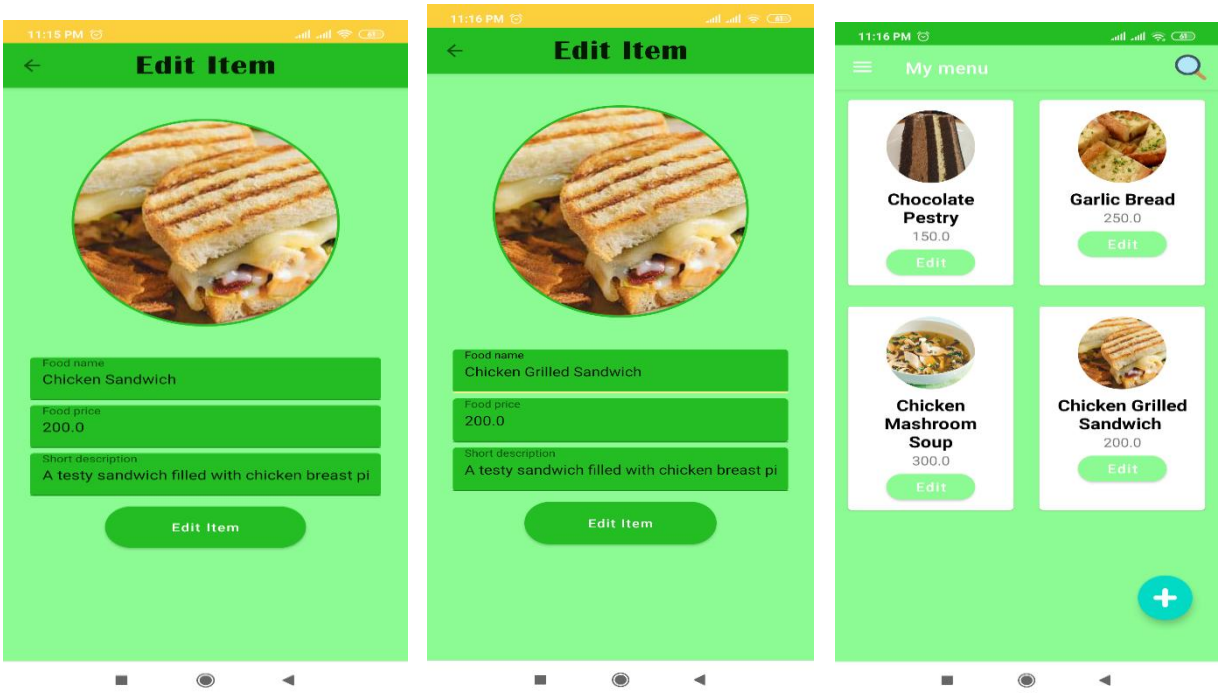


Figure 13: update the item name in main manu

6.2.6 Restaurent QR Code

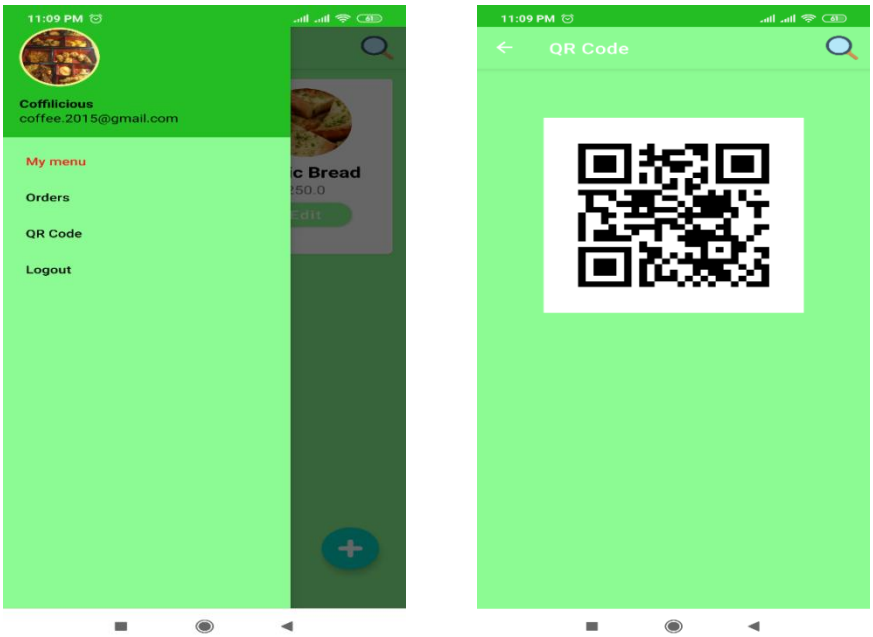


Figure 14: Restaurent owned qr code scanned by user

6.3 Backend Process of the System

The application backside is enforced on base that is straightforward to utilize,
has simple application facilitating, capacity to
line client confirmation, comfort of information analytics
and facilitates strategy of controlling, erasing, and re-ordering information inside the information.

6.3.1 Database Design Analyze

Sometime recently working with data in any field, at first we'd like to get a handle on that in any case will a data have to be be planned. Figure
fifteen appears the data pattern, totally distinctive application table and their connections.
It characterizes how the data is organized and the way the relations among them
are related and defines all the limitations that are connected on the program bundle.

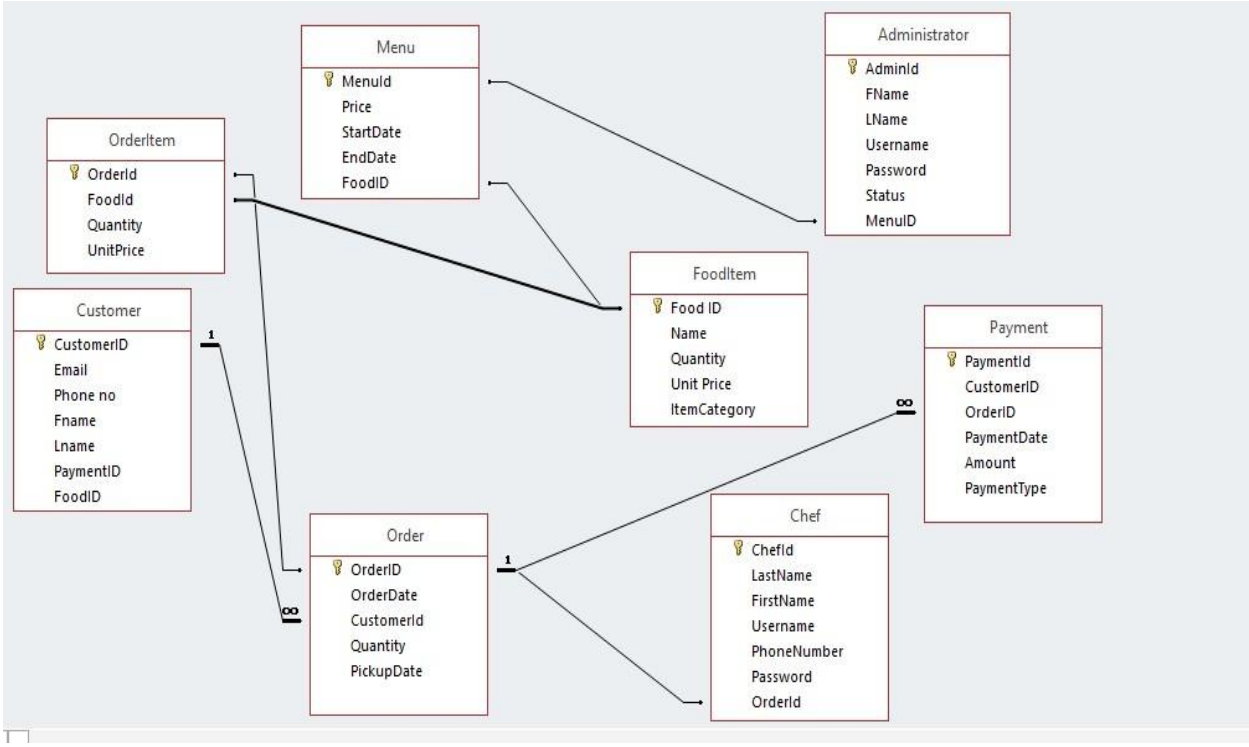


Figure 15:Database Construction for the Eatery Requesting Framework

6.3.2 Firebase Setup & Working Process

As base of operations implement all the back-end applications of a system, thus this can
be for watching, news and administration of the applying. Figure sixteen shows a part of the backend.

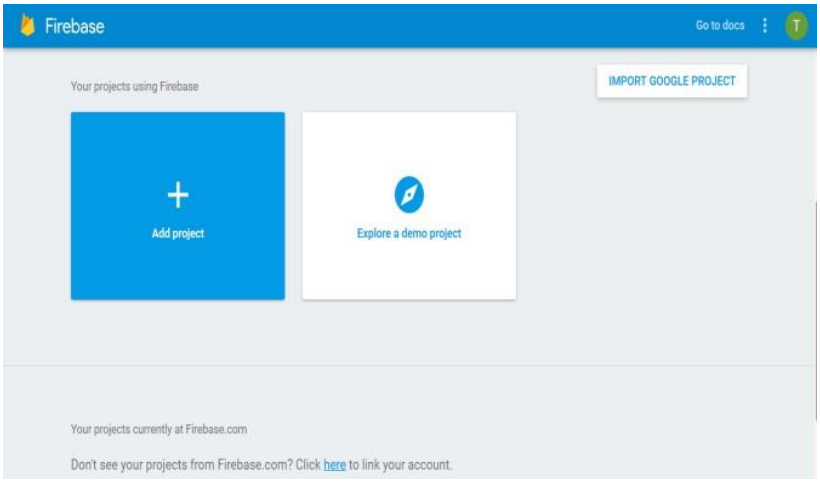


Figure 16: Backend Setup Page

After create a account in firebase, we create a new project called “restaurant”.

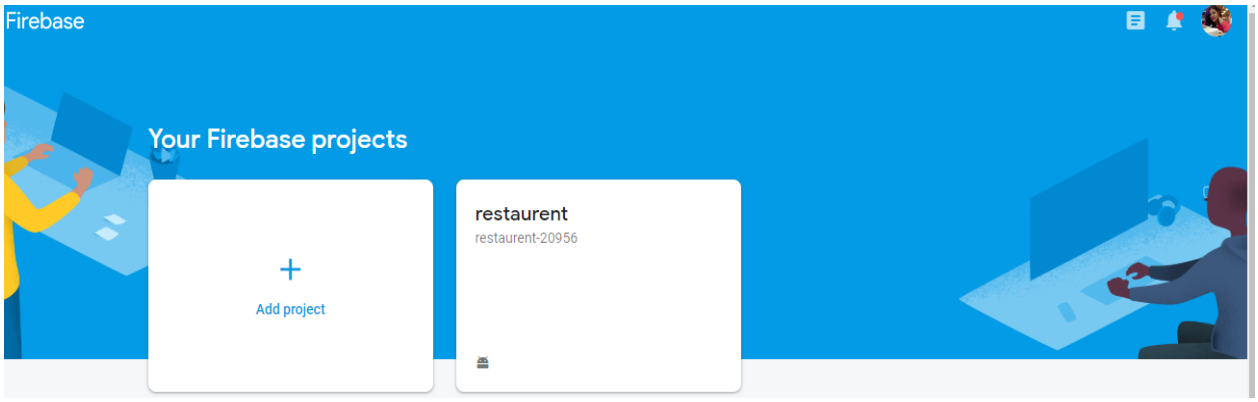
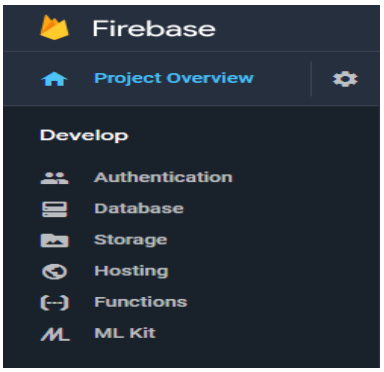


Figure 17: New project create in Firebase

After enter to the restaurant project, we can see these options against this project. That is:



In the ‘Authentication’ part, all the names and information of users and restaurent are shown that are registered to this app.

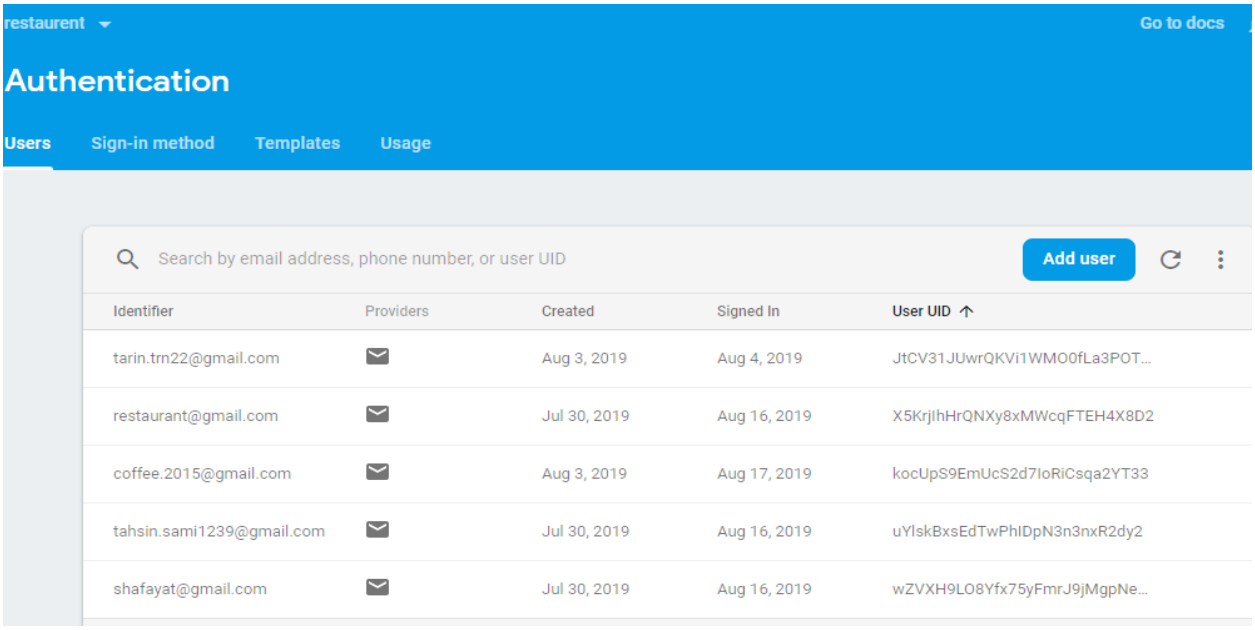


Figure 18: Firebase authentication interface

In ‘Database’, all the data of users, restaurants or all type of order are being stored.

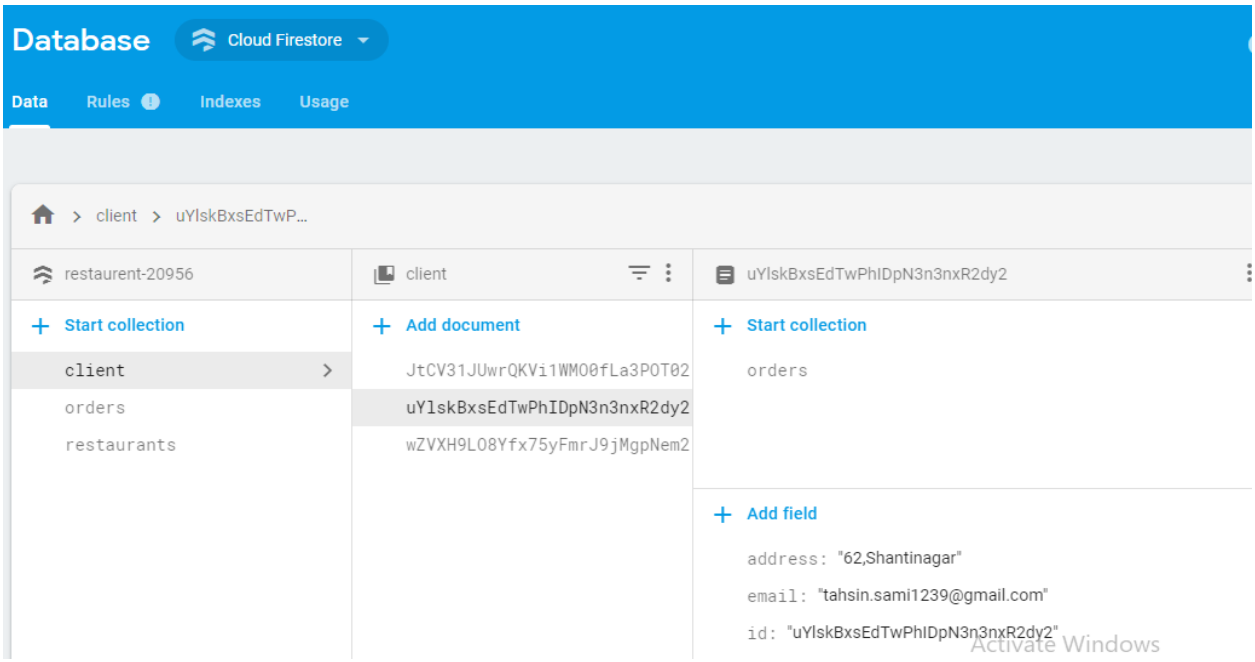


Figure 19: Database interface from firebase

In ‘Storage’, all type of images or cotes that are used in this app are being stored.

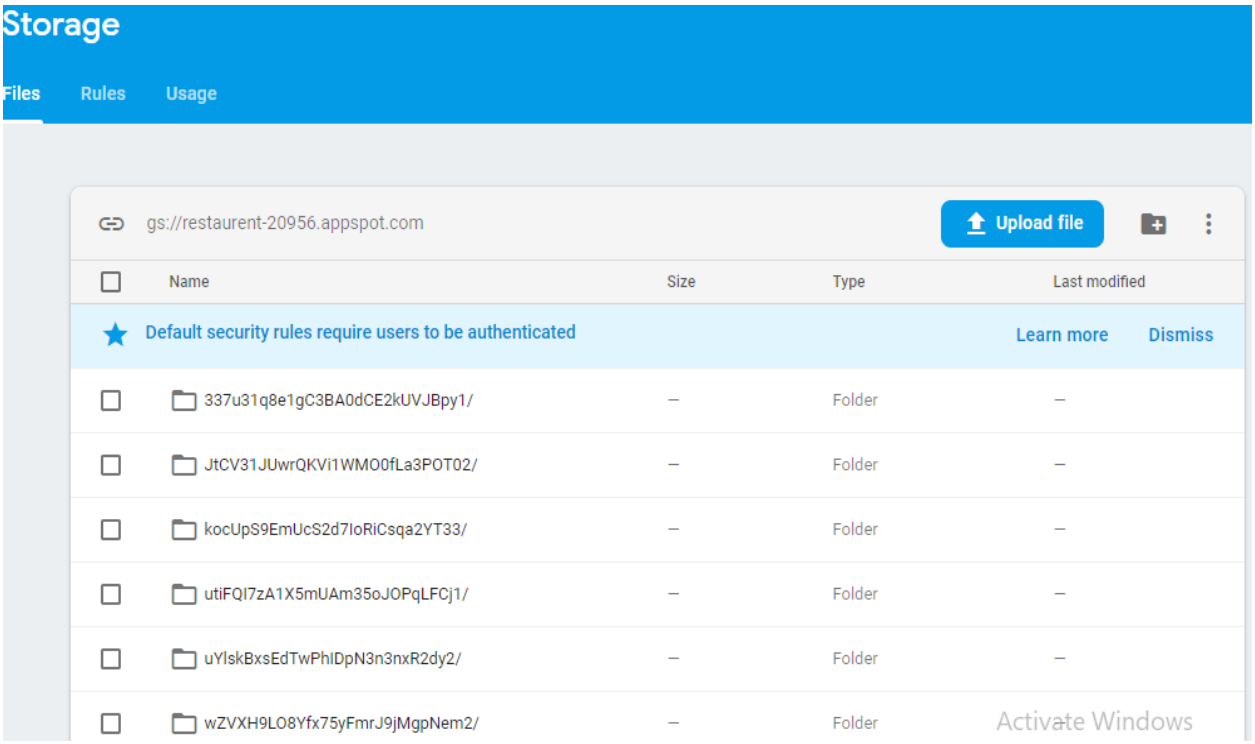


Figure 20: Storage interface in firebase

Chapter 7

System Testing & Evalution

7.1 System Testing Operation

Testing is that the procedure of assessing a program to distinguish varieties between the given input and so the anticipated yield. It's together essential in surveying the choices of a program and evaluating the quality of the merchandise. Now to test our app, we are going see that how this gap requesting prepare is completed from both side Clint & Restaurant.

1. At first user will login and will go to a restaurant from her list that is named 'Coffilicious'. And then it will show the menu that the 'Coffilicious' restaurant belong.

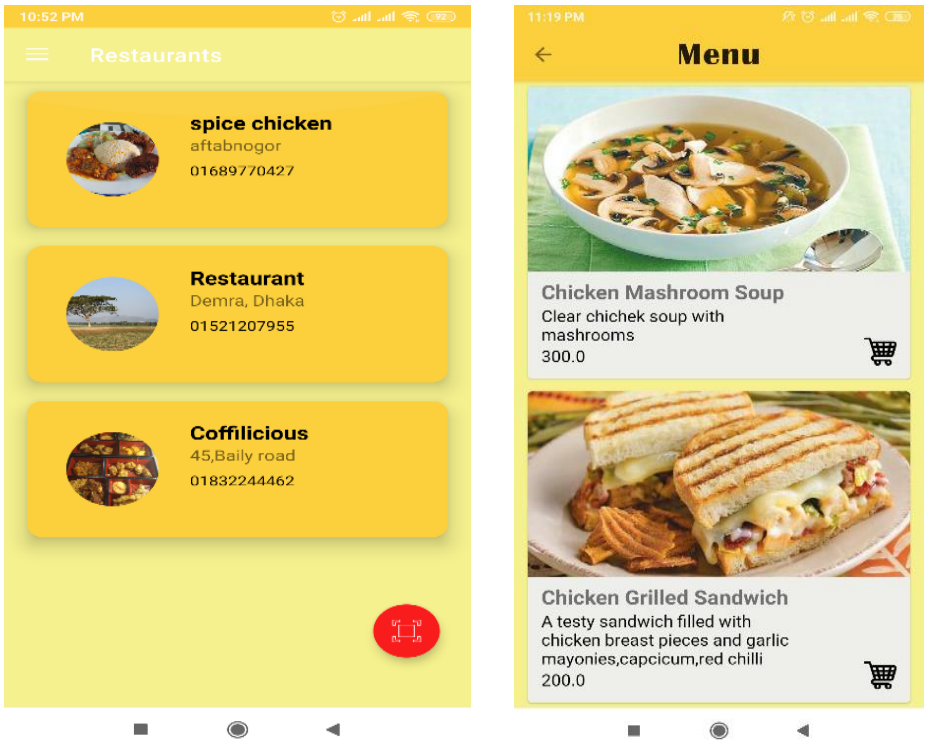


Figure 22: Restaurent menu shown to the user

2. Then user will select items, keep it to the cart and checkout to conferm her order.

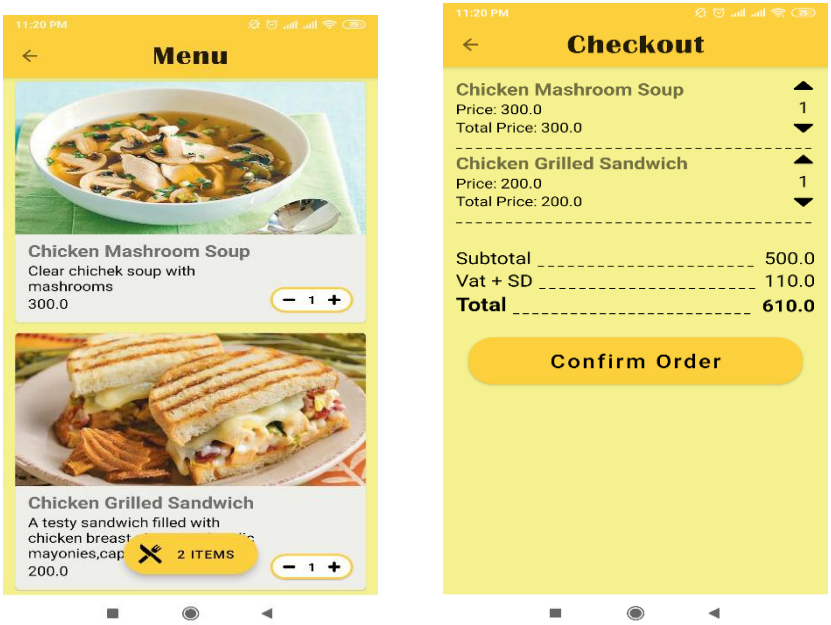


Figure 23: User checkout page

3. Every food item generates with a code number. When user confirm her order then the chief will get a notification with user name, mobile number and code number of food.Then user will paying the bill in the booth, then from booth, it will be confirmed in the kitchen that the user have paid her bill. After that the chief accept the order request and user will get a order accepted message in user's mobile.

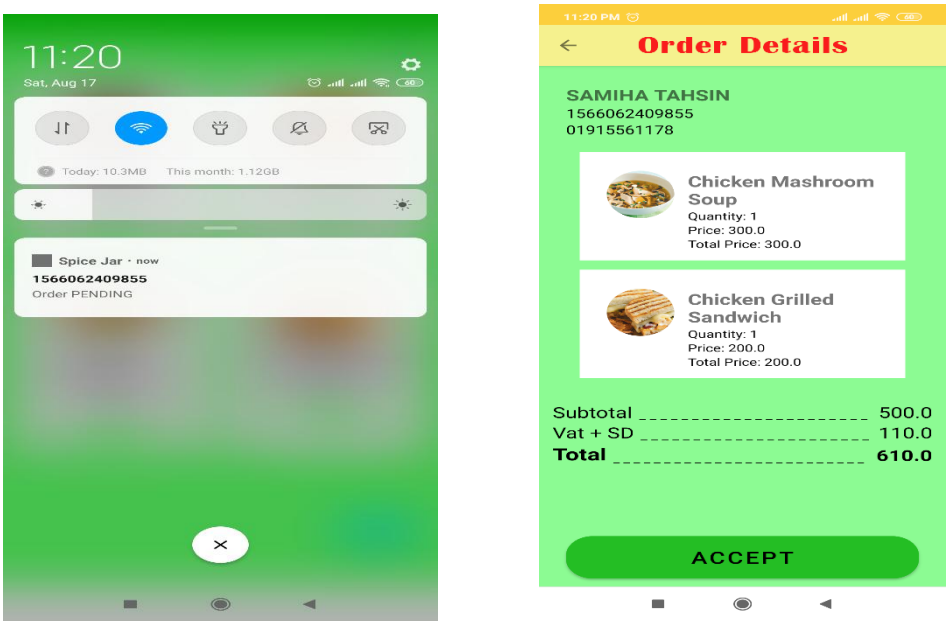


Figure 24: Restaurent confirm the food order

4. Then user get a message to her mobile about the confirmation of her order that it has accepted.

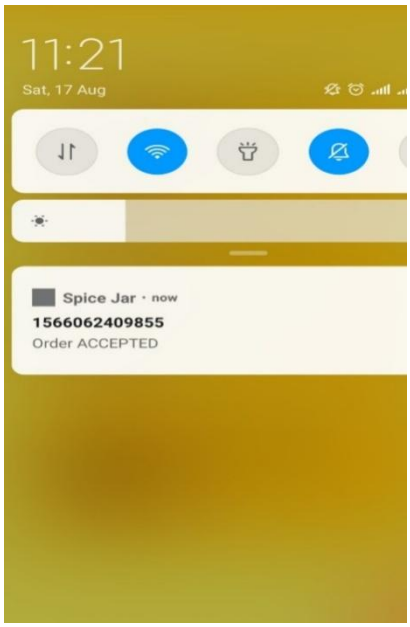
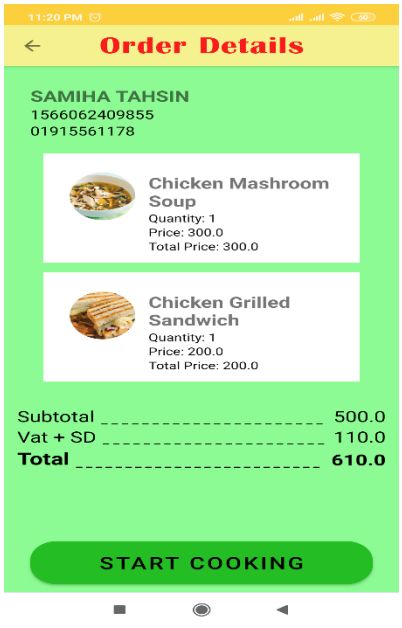


Figure 25: Order confirmation message to user

5. Restaurent will moving forward, use the butten of ‘ START COOKING’ , and again user will get another notification about this.



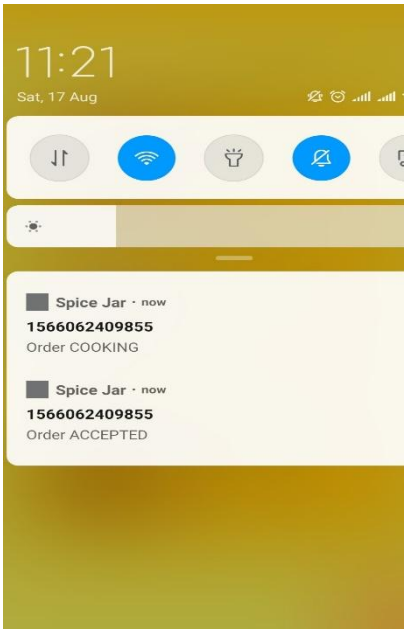


Figure 26: Restaurant confirm about starting of cooking

6. When food is ready for the delivary, Restaurant notified user that food is prepare for the delivary.

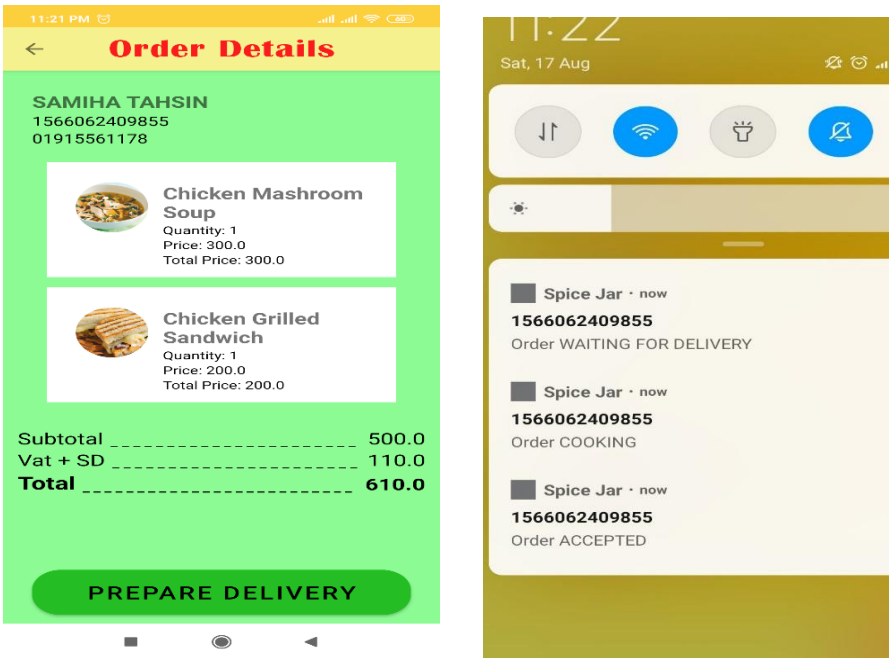


Figure 27: Food Delivary prepare notification

7. Finally when food is delivered by Restaurant, user lastly got the final notification that her food is delivered.

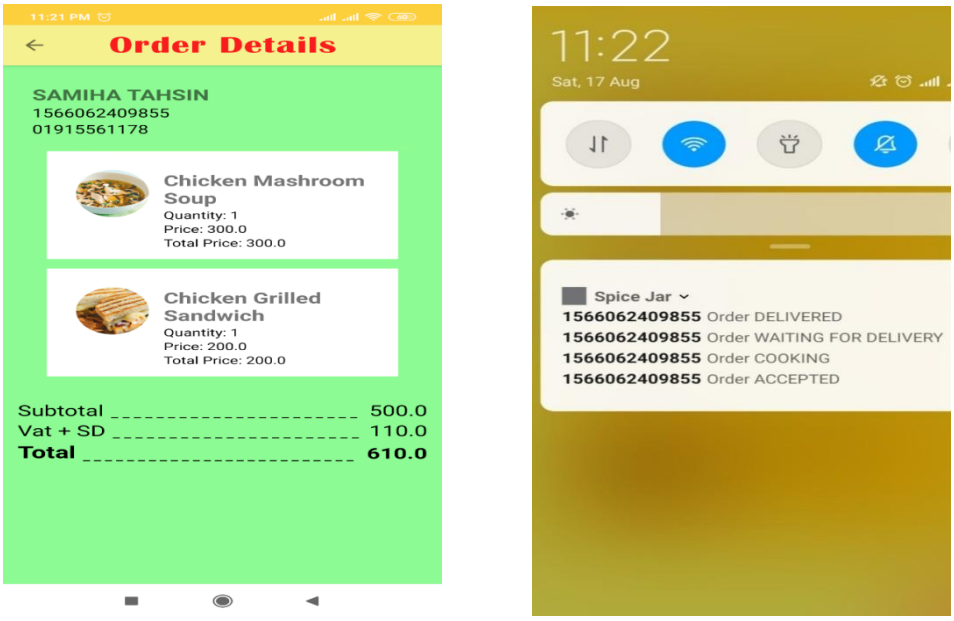


Figure 28: Food is delivered

7.2 Evaluation

Our venture has brought about into in operation illustration which can perform simple action with posting to the server and recovering information. This occasion is completely working and brings the structure which can be utilized in any advancement of application.

7.2.1 Evaluation of the Outcome

Due to the tight plan we have a inclination to fizzled to have sufficient time to create application that would contain full highlights which were expressed within the objectives chapter but the model of the application is speaking to the scope of the venture and have property that were sketched out inside the scope.

7.2.2Evolution of Origin

The machine stage that's chosen for the applying has endless data and instructional exercises which may encourage us though creating the application. This web supply is amazingly enlightening and guides a way to construct application from scratch. Same supplyhttp://developer.android.com/sdk will facilitate U.S. to form the vital environment.

7.2.3 Procedure Evaluation

The method of this venture are made open as expedient advancement with getting genuine result. As old venture would final for 6 month this extend was wrapped up one and for one month. The completion of this work was as a result of perseverance of extend specialist and understudy.
This framework was partitioned into stages and was taken after by simply the once per week news and afterward twofold each week scope. Apparatuses that were utilized all through this venture were no heritable liberated from charge.

Chapter 8

Future & Conclusion

8.1 Future of the System

In our system we have kept the paying or billing system manually. It means after order their food from menu, they have to paying the bill by going to the booth. After paying the bill, chief will confirm their order by sending a notification message in their mobile. But in future we want to upgrade our application by keep a payment option directly to our app. We want to include the bkash, rocket, Nogod, visa card or any kind of American express card system in our Paying option so that it will be more convenient for both user and restaurant.
As all these paying systems are related to banking sector, so we have to elaborate our technical knowledge more to deal with these stuffs. And for now, the user gets the menu by scanning the qr code which is restaurant's own property but we have a plan to make it in network base like when user will entry to any restaurant and when they will be connected to the restaurant'sWi-Fi system, they will automatically get the restaurant menu. I hope in future we can truly elaborate our plans successfully.

There are some other features we are planning to include in our app in future. Such as:

8.1.1 Advance Reservations

Separated from nourishment requesting apps, there are numerous progress booking and table saving apps like Open Table and No hold up that edit the sum of a few time that clients have to be be compelled to require a position to eat. Be that as it may, these apps have the same point, but they have through and through completely diverse approaches. If we will incorporate this reservation office it'll be useful for us because it can pulled in client flow.

8.1.2 Ratings & Reviews

Eateries can get ready completely diverse exercises to boost customer benefit per their surveys.So we want to add the rating system for customer by which they can inform us about our service and rating system will also help others to judge us.

8.1.3 Discount & Deals for Customers

We want to deal our app with discount deals which will make it easier for users to go out and have delicious food at the most affordable rates. So we will try to work our app with different restaurants to bring in new successive business by dealing with discount materials.

8.2 Conclusion

Our target for this venture was to create an android application for client benefit whereas not serves in eateries and in our venture. We have a propensity to achieve this mpoint. This application offers client the adaptivity to observe accounts, orders and perform exercises related with it.We have tried to make our app user friendly. And alsotried to test the application against the server provided by a developer, who develops web application servers.This application, was implemented on laptop devices, so the text size and design were adjustable to devices with bigger screens.On small devices, it would cause problems to control the applications. Developing our application gave us a lot of experience we learned new methods of how to implement application features, how to find the best solutions and improve our problemsolving.Writing this projec twas part of ubject Project Control; so many learned topics were used in this project and gave us the opportunity to understand thembetter.

We would like to thank our supervisor as he spends his time helping us with every delicate part of our project and keep advising us during that time. We are really grateful to him that we have learned a lot of matters from this,He guide us how to deal with the circumstances that we may face in future while doing this type of project. It will be also helpful in our future ITcareer.

References

1. https://www.upwork.com/hiring/mobile/how-mobile-apps-have-transformed-restaurant-and-food-delivery-industry/
2. https://pdfs.semanticscholar.org/41f7/a38be6b154eb62d74bd0d0f308ca5e48b50e.pdf
3. https://soltech.net/software-development-technologies/android-system-development/
4. https://pdfs.semanticscholar.org/e0f8/610c765968217a1ca396ccbce655fc78dd82.pdf
5. https://syndicode.com/2017/10/05/top-6-software-development-methodologies/
6. https://dspace.cvut.cz/bitstream/handle/10467/62692/F3-BP-2016-Hogenauer-Tomas-Android%20klient%20restauracniho%20systemu.pdf?sequence=1
7. https://blog.codemagic.io/what-is-flutter-benefits-and-limitations/

Appendix (Code)

For splash page:

```
package com.example.restaurant.ui

import android.os.Bundle
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.data.pref.AppPreference
import com.example.restaurant.ui.clientdashboard.ClientDashBoardActivity
import com.example.restaurant.ui.getstarted.GetStartedActivity
import com.example.restaurant.ui.restaurantdashboard.RestaurantDashboardActivity

class SplashActivity : BaseActivity() {

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
val preference = AppPreference()

if(preference.id.isNotEmpty()) {
if(preference.userType == "client") {
            startActivityAndFinishCurrent(ClientDashBoardActivity::class.java)
        } else if(preference.userType == getString(R.string.user_type_restaurant)){
            startActivityAndFinishCurrent(RestaurantDashboardActivity::class.java)
        }
    } else {
        startActivityAndFinishCurrent(GetStartedActivity::class.java)
    }

  }
}
```

Store data through network in firebase (For User Order details):

Order Network source:

```
addOnSuccessListener {
            callback.onSuccess(Unit)
        }.addOnFailureListener {
            callback.onError(it)
        }
  }
fun placeOrder(order: Order, callback: OrderNetworkSourceCallback<Unit>) {
    firestore.collection("orders").document(order.id.toString())
        .set(order)
        .addOnSuccessListener {
callback.onSuccess(Unit)
```

```
}.addOnFailureListener{
callback.onError(it)
}
}
fun getOrders(userType: String, uid: String, callback:
OrderNetworkSourceCallback<QuerySnapshot>) {
        firestore.collection(userType).document(uid).collection("orders").orderBy("id",
Query.Direction.DESCENDING)
            .get().addOnSuccessListener {
callback.onSuccess(it)
}.addOnFailureListener {
callback.onError(it)
}
}
fun updateOrderStatus(userType: String, uid: String, order: Order, callback:
OrderNetworkSourceCallback<Unit>) {
firestore.collection(userType).document(uid).collection("orders").document(order.id.toString())
            .update("status", order.status)package
com.example.restaurant.data.networksource.order
    import com.example.restaurant.model.Order
                import com.google.firebase.firestore.*

class OrderNetworkSource {

private val firestore: FirebaseFirestore = FirebaseFirestore.getInstance()

fun placeUsersOrder(userType: String, uid: String, order: Order, callback:
OrderNetworkSourceCallback<Unit>) {
firestore.collection(userType).document(uid).collection("orders").document(order.id.toString()
)
                    .set(order)
                    .
            .addOnSuccessListener {
callback.onSuccess(Unit)
}.addOnFailureListener {
callback.onError(it)
}
}
}
```

Order Network source Callback:

```
package com.example.restaurant.data.networksource.order

import java.lang.Exception
interface OrderNetworkSourceCallback<T>{
fun onSuccess(result: T)
fun onError(exception: Exception)
```

}

For Restaurant Details store in firebase:

Restaurant Network source:

```
packagecom.example.restaurant.data.networksource.restaurant

importandroid.net.Uri
import com.example.restaurant.model.FoodItem
import com.example.restaurant.utils.MENU
import com.example.restaurant.utils.RESTAURANT
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.firestore.QuerySnapshot
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.UploadTask

class RestaurantNetworkSource {

private val firestore: FirebaseFirestore = FirebaseFirestore.getInstance()
private val storage: FirebaseStorage = FirebaseStorage.getInstance()

fun getRestaurantList(callback: RestaurantNetworkCallback<QuerySnapshot>) {
val collection = firestore.collection(RESTAURANT)
      collection.get().addOnSuccessListener {
callback.onSuccess(it)
}.addOnFailureListener {
callback.onError(it)
}
}

fun getMenu(restaurantId: String, callback: RestaurantNetworkCallback<QuerySnapshot>) {
val collection = firestore.collection(RESTAURANT).document(restaurantId).collection(MENU)
      collection.get().addOnSuccessListener {
callback.onSuccess(it)
}.addOnFailureListener {
callback.onError(it)
}
}

fun setMenuItem(restaurantId: String, menuItem: FoodItem, callback:
RestaurantNetworkCallback<Unit>) {
val document =
firestore.collection(RESTAURANT).document(restaurantId).collection(MENU).document(menuI
tem.id.toString())
      document.set(menuItem).addOnSuccessListener {
callback.onSuccess(Unit)
}.addOnFailureListener {
callback.onError(it)
```

```
}
}

fun uploadFoodItemImage(uri: Uri, restaurantId: String, foodId: Long, callback:
RestaurantNetworkCallback<UploadTask.TaskSnapshot>) {
val path = storage.reference.child("$restaurantId/$foodId.jpg")
      path.putFile(uri).addOnSuccessListener {
callback.onSuccess(it)
}.addOnFailureListener {
callback.onError(it)
}
}
}
```

Restaurant Network source Callback:

```
package com.example.restaurant.data.networksource.restaurant

import java.lang.Exception
interface RestaurantNetworkCallback<T>{

fun onSuccess(result: T)
fun onError(exception: Exception)
}
```

Registration page for both User & Restaurant:

Registration activity:

```
package com.example.restaurant.ui.registration

import android.content.Intent
import android.os.Bundle
import androidx.lifecycle.Observer
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.ui.clientdashboard.ClientDashBoardActivity
import com.theartofdev.edmodo.cropper.CropImage
import com.theartofdev.edmodo.cropper.CropImageView
import kotlinx.android.synthetic.main.activity_registration.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast
import android.R.attr.data
import android.app.Activity
import android.net.Uri
import com.bumptech.glide.Glide
import com.example.restaurant.ui.restaurantdashboard.RestaurantDashboardActivity


class RegistrationActivity : BaseActivity() {
```

```
companion object {
const val USER_TYPE = "user_type"
}
private lateinit var userType: String
private val presenter = RegistrationPresenter()
private var uri : Uri? = null

   override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_registration)

      setActionBar(toolbar, true)

userType = intent.getStringExtra(USER_TYPE) ?: ""


initListener()
      initClickListener()
   }

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
if(requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
try {
val result = CropImage.getActivityResult(data)
uri = result.uri
if (uri != null) {
                Glide.with(this@RegistrationActivity).load(uri).into(ivProfile)
            }
         }catch (e: Exception) {
            e.printStackTrace()
         }
      }
else super.onActivityResult(requestCode, resultCode, data)
   }

private fun initClickListener() {
      ivProfile.onClick {
CropImage.activity()
            .setGuidelines(CropImageView.Guidelines.ON)
            .start(this@RegistrationActivity)
}

btnRegister.setOnClickListener {
         presenter.registration(
uri,
```

```
            etEmail.text.toString(),
            etPassword.text.toString(),
            etMobile.text.toString(),
            etAddress.text.toString(),
            etName.text.toString(), userType)
}
}

private fun initListener() {
presenter.emailError.observe(this, Observer {
tilEmail.error = it
    })
presenter.addressError.observe(this, Observer {
tilAddress.error = it
    })
presenter.nameError.observe(this, Observer {
tilName.error = it
    })
presenter.passwordError.observe(this, Observer {
tilPassword.error = it
    })
presenter.phoneNoError.observe(this, Observer {
tilMobile.error = it
    })
presenter.message.observe(this, Observer {
toast(it)
})
presenter.loaderLiveData.observe(this, Observer {
        if(it) showLoader()
else hideLoader()
})
presenter.userLiveData.observe(this, Observer {
        if(userType == "client") {
           startActivityAndFinishCurrent(ClientDashBoardActivity::class.java)
        } else {
           startActivityAndFinishCurrent(RestaurantDashboardActivity::class.java)
        }
})
   }
}
```

Registration Presenter:

```
package com.example.restaurant.ui.registration

import android.net.Uri
import androidx.lifecycle.MutableLiveData
```

```
import com.example.restaurant.base.BasePresenter
import com.example.restaurant.data.repository.auth.AuthRepository
import com.example.restaurant.data.repository.auth.AuthRepositoryCallback
import com.example.restaurant.data.repository.auth.AuthRepositoryImpl
import com.example.restaurant.model.User
import com.example.restaurant.utils.isEmailValid
import com.example.restaurant.utils.isPhoneNoValid
import com.orhanobut.logger.Logger
import java.lang.Exception

class RegistrationPresenter : BasePresenter() {

private val repository: AuthRepository =
AuthRepositoryImpl()
val emailError = MutableLiveData<String>()
val nameError = MutableLiveData<String>()
val phoneNoError = MutableLiveData<String>()
val addressError = MutableLiveData<String>()
val passwordError = MutableLiveData<String>()
val userLiveData = MutableLiveData<User>()

fun registration(photoUri: Uri?, email: String, password: String, phoneNo: String, address:
String, name: String, userType: String) {

val user = User(name, email, address, phoneNo, userType = userType)
if(validate(user, password, photoUri)) {
        loaderLiveData.value = true
repository.registration(
photoUri!!,
userType,
user,
email,
password,
object : AuthRepositoryCallback<User> {
override fun onSuccess(result: User) {
Logger.d(result)
                    userLiveData.value = result
                    loaderLiveData.value = false
                    message.value = "Registration successful"
                }

override fun onError(exception: Exception) {
exception.printStackTrace()
                    message.value = exception.localizedMessage
                    loaderLiveData.value = false
                }
```

```
            })
        }
    }

private fun validate(user: User, password: String, photoUri: Uri?): Boolean {
var isValid = true
        emailError.value = null
        nameError.value = null
        addressError.value = null
        phoneNoError.value = null
        passwordError.value = null
if(!user.email.isEmailValid()) {
isValid = false
            emailError.value = "Enter a valid email"
        }
if(user.name.length <= 2) {
isValid = false
            nameError.value = "Enter a valid name"
        }
if(user.address.length <= 5) {
isValid = false
            addressError.value = "Enter a valid address"
        }
if(user.phoneNo.length != 11 && !user.phoneNo.isPhoneNoValid()) {
isValid = false
            phoneNoError.value = "Enter a valid phone number"
        }
if(password.length <= 5) {
isValid = false
            passwordError.value = "Enter a valid password"
        }
if(photoUri == null) {
isValid = false
            message.value = "Select a profile picture"
        }
return isValid
    }
}
```

Login page for both User & Restaurant:

Login activity:

```
package com.example.restaurant.ui.login

import android.content.Intent
import android.os.Bundle
```

```
import android.text.SpannableString
import android.text.Spanned
import android.text.method.LinkMovementMethod
import android.text.style.ClickableSpan
import android.text.style.ForegroundColorSpan
import android.view.MenuItem
import android.view.View
import androidx.core.content.ContextCompat
import androidx.lifecycle.Observer
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.ui.clientdashboard.ClientDashBoardActivity
import com.example.restaurant.ui.registration.RegistrationActivity
import com.example.restaurant.ui.restaurantdashboard.RestaurantDashboardActivity
import kotlinx.android.synthetic.main.activity_login.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast

class LoginActivity : BaseActivity() {

companion object {
const val USER_TYPE = "user_type"
}
lateinit var userType: String
private val presenter = LoginPresenter()

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_login)
      setActionBar(toolbar, true)

userType = intent.getStringExtra(USER_TYPE) ?: ""

      val string = SpannableString(getString(R.string.new_user_create_an_account))
val clickSpan = object: ClickableSpan() {
override fun onClick(p0: View) {
val intent = Intent(applicationContext, RegistrationActivity::class.java)
            intent.putExtra(RegistrationActivity.USER_TYPE, userType)
            startActivity(intent)
         }
      }

val colorSpan = ForegroundColorSpan(ContextCompat.getColor(applicationContext,
R.color.red))
      string.setSpan(clickSpan, string.indexOf("Create"), string.length,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
```

```
        string.setSpan(colorSpan, string.indexOf("Create"), string.length,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
        tvRegister.text = string
        tvRegister.movementMethod = LinkMovementMethod.getInstance()

        btnLogin.onClick {
            presenter.login(userType, etEmail.text.toString(), etPassword.text.toString())
}

        presenter.messageLiveData.observe(this, Observer {
toast(it)
})
presenter.emailError.observe(this, Observer {
tilEmail.error = it
        })
presenter.passwordError.observe(this, Observer {
tilPassword.error = it
        })

presenter.userLiveData.observe(this, Observer {
            if(userType == "client") {
                startActivityAndFinishCurrent(ClientDashBoardActivity::class.java)
            } else if(userType == getString(R.string.user_type_restaurant)) {
                startActivityAndFinishCurrent(RestaurantDashboardActivity::class.java)
            }
})
presenter.loaderLiveData.observe(this, Observer {
            if(it) showLoader()
else hideLoader()
})
    }
```

Login Presenter:

```
package com.example.restaurant.ui.login

import androidx.lifecycle.MutableLiveData
import com.example.restaurant.base.BasePresenter
import com.example.restaurant.data.repository.auth.AuthRepository
import com.example.restaurant.data.repository.auth.AuthRepositoryCallback
import com.example.restaurant.data.repository.auth.AuthRepositoryImpl
import com.example.restaurant.model.User
import com.example.restaurant.utils.isEmailValid
import com.orhanobut.logger.Logger
import java.lang.Exception

class LoginPresenter : BasePresenter() {
```

```
private val repository : AuthRepository =
        AuthRepositoryImpl()
val userLiveData = MutableLiveData<User>()
val messageLiveData = MutableLiveData<String>()
val emailError = MutableLiveData<String?>()
val passwordError = MutableLiveData<String?>()

fun login(userType: String, email: String, password: String) {
if(validate(email, password)) {
loaderLiveData.value = true
            repository.login(userType, email, password, object :
                AuthRepositoryCallback<User> {
override fun onSuccess(result: User) {
                    Logger.d("Login success")
userLiveData.value = result
loaderLiveData.value = false
                    messageLiveData.value = "Login successful"
}

override fun onError(exception: Exception) {
                    exception.printStackTrace()
messageLiveData.value = exception.localizedMessage
loaderLiveData.value = false
}

            })
        }
    }

private fun validate(email: String, password: String): Boolean {
var isValid = true
        emailError.value = null
        passwordError.value = null
        if(!email.isEmailValid()) {
            isValid = false
            emailError.value = "Enter a valid email"
}
if(password.length <= 5) {
            isValid = false
            passwordError.value = "Password must have at least 6 characters"
}
return isValid
    }
}
```

For User Interface Part:

Clint Dash Board Activity:

```
package com.example.restaurant.ui.clientdashboard

import android.os.Bundle
import com.google.android.material.floatingactionbutton.FloatingActionButton
import com.google.android.material.snackbar.Snackbar
import androidx.navigation.findNavController
import androidx.drawerlayout.widget.DrawerLayout
import com.google.android.material.navigation.NavigationView
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import android.view.Menu
import android.view.MenuItem
import androidx.core.view.GravityCompat
import androidx.navigation.ui.*
import com.bumptech.glide.Glide
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.data.pref.AppPreference
import com.example.restaurant.ui.SplashActivity
import com.example.restaurant.ui.dialogs.LogoutDialog
import kotlinx.android.synthetic.main.nav_header_client_dash_board.view.*
import org.jetbrains.anko.support.v4.drawerListener

class ClientDashBoardActivity : BaseActivity() {

companion object {
const val MOVE_TO = "move_to"
}
private lateinit var appBarConfiguration: AppBarConfiguration
private val preference = AppPreference()
private lateinit var navMenuItem : MenuItem
private var gotoDestination = false
   override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_client_dash_board)
val toolbar: Toolbar = findViewById(R.id.toolbar)
       setActionBar(toolbar, false)
val moveTo = intent.getIntExtra(MOVE_TO, 0)
val drawerLayout: DrawerLayout = findViewById(R.id.drawer_layout)
val navView: NavigationView = findViewById(R.id.nav_view)
val navController = findNavController(R.id.nav_host_fragment)
// Passing each menu ID as a set of Ids because each
       // menu should be considered as top level destinations.
appBarConfiguration = AppBarConfiguration(
```

```
setOf(
            R.id.nav_restaurants, R.id.nav_orders
), drawerLayout
        )
setupActionBarWithNavController(navController, appBarConfiguration)
        navView.setupWithNavController(navController)
//        navController.navigate(R.id.nav_restaurants)
navMenuItem = navView.menu.getItem(0)

val header = navView.getHeaderView(0)
        header.tvUserName.text = preference.userSession.name
header.tvEmail.text = preference.userSession.email
Glide.with(this).load(preference.userSession.profilePhotoUrl).into(header.ivProfile)
        navView.setNavigationItemSelectedListener { menuItem ->
            if (menuItem.itemId == R.id.nav_logout) {
                LogoutDialog(object: LogoutDialog.Callback{
override fun onLogoutClick() {
preference.logout()
                    startActivityAndFinishCurrent(SplashActivity::class.java)
                }
            }).show(supportFragmentManager, "logout")
        } else {
if (navMenuItem.itemId != menuItem.itemId) {
navMenuItem = menuItem
gotoDestination = true
}
        }
        drawerLayout.closeDrawer(GravityCompat.START, true)
return@setNavigationItemSelectedListener true
        }
drawerLayout.drawerListener {
        this.onDrawerClosed {
            if (gotoDestination) {
                navController.navigate(navMenuItem.itemId)
gotoDestination = false
}
}
        }
        if(moveTo!=0) {
            navController.navigate(navView.menu.getItem(0).itemId)
        }
    }
override fun onSupportNavigateUp(): Boolean
val navController = findNavController(R.id.nav_host_fragment)
return navController.navigateUp(appBarConfiguration) || super.onSupportNavigateUp()
    }
```

```
}
```

Clint Order Checkout Activity:

```
package com.example.restaurant.ui.checkout

import android.content.Intent
import android.os.Bundle
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.model.FoodItem
import com.example.restaurant.model.Order
import com.example.restaurant.ui.clientdashboard.ClientDashBoardActivity
import kotlinx.android.synthetic.main.activity_checkout.*
import kotlinx.android.synthetic.main.activity_checkout.toolbar
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast

class CheckoutActivity : BaseActivity() {

companion object {
const val ORDER = "order"
      const val ORDER_REQUEST_CODE = 101
   }

private var order: Order = Order()
var subTotal = 0.0
var vat = 0.0
private val presenter = CheckoutPresenter()
val dataIntent: Intent = Intent()

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_checkout)
      setActionBar(toolbar, true)
order = intent.getParcelableExtra(ORDER)
dataIntent.putExtra(ORDER, order)
      setResult(ORDER_REQUEST_CODE, dataIntent)
      rvCheckoutItems.layoutManager = LinearLayoutManager(this)
      rvCheckoutItems.adapter = CheckoutItemAdapter(object: CheckoutItemAdapter.Callback{
override fun onItemChange(item: FoodItem, count: Long) {
when {
            count == 0L ->order.items.remove(item.id.toString())
order.items[item.id.toString()]!!.size> count -
>order.items[item.id.toString()]!!.removeAt(order.items[item.id.toString()]!!.size-1)
```

```
else ->order.items[item.id.toString()]!!.add(item)
            }
dataIntent.putExtra(ORDER, order)
            setResult(ORDER_REQUEST_CODE, dataIntent)
if(order.items.isEmpty()) {
                finish()
            } else {
                (rvCheckoutItems.adapter as CheckoutItemAdapter).setOrder(order.items)
                calculateSubTotal()
            }
        }

    }).apply {
setOrder(order.items)
}
btnPlaceOrder.onClick {
        presenter.placeOrder(order)
}

    presenter.loaderLiveData.observe(this, Observer {
        if(it) showLoader()
else hideLoader()
})
presenter.message.observe(this, Observer {
toast(it)
})

presenter.orderPlaced.observe(this, Observer {
        if(it) {
val intent = Intent(this@CheckoutActivity, ClientDashBoardActivity::class.java)
            intent.putExtra(ClientDashBoardActivity.MOVE_TO, 1)
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
            startActivity(intent)
        }
})

    calculateSubTotal()

  }

fun calculateSubTotal() {
subTotal = 0.0
order.items.forEach {
        var multiplier = it.value.size
        subTotal += it.value[0].price * multiplier
}
```

```
tvSubTotal.text = subTotal.toString()
order.subTotal = subTotal
calculateVat()
    }

fun calculateVat() {
vat = subTotal * .22
        tvVat.text = vat.toString()
        calculateTotal()
    }

fun calculateTotal() {
        tvTotal.text = (subTotal + vat).toString()
order.total = subTotal + vat
}

}
```

Restaurant Menu Activity for User:

```
package com.example.restaurant.ui.restaurantmenu

import android.content.Intent
import android.os.Bundle
import android.view.View.GONE
import android.view.View.VISIBLE
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.data.pref.AppPreference
import com.example.restaurant.model.FoodItem
import com.example.restaurant.model.Order
import com.example.restaurant.model.User
import com.example.restaurant.ui.checkout.CheckoutActivity
import com.orhanobut.logger.Logger
import kotlinx.android.synthetic.main.activity_restaurant_menu.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast
import java.util.function.BiFunction

class RestaurantMenuActivity : BaseActivity() {

companion object {
const val RESTAURANT = "restaurant"
        const val id = "ID"
}
```

```
private var cart = mutableMapOf<String, MutableList<FoodItem>>()

private val presenter = RestaurantMenuPresenter()
private val preference = AppPreference()
lateinit var restaurant: User

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_restaurant_menu)
      setActionBar(toolbar, true)
restaurant = intent.getParcelableExtra(RESTAURANT)
      rvMenu.layoutManager = LinearLayoutManager(applicationContext)
      rvMenu.adapter = RestaurantMenuAdapter(object: RestaurantMenuAdapter.Callback{
override fun onAddToCart(item: FoodItem, count: Long) {

if(cart[item.id.toString()] != null) {
when {
                  count == 0L ->cart.remove(item.id.toString())
cart[item.id.toString()]!!.size> count -
>cart[item.id.toString()]!!.removeAt(cart[item.id.toString()]!!.size-1)
else ->cart[item.id.toString()]!!.add(item)
               }
            } else {
cart[item.id.toString()] = mutableListOf(item)
            }
            Logger.d(cart)
            refreshCart()
         }

      })

      btnCheckout.onClick {
         val order = Order(items = cart, client = preference.userSession, restaurant =
restaurant)
val intent = Intent(applicationContext, CheckoutActivity::class.java)
         intent.putExtra(CheckoutActivity.ORDER, order)
         startActivityForResult(intent, CheckoutActivity.ORDER_REQUEST_CODE)
}

      presenter.menu.observe(this, Observer {
(rvMenu.adapter as RestaurantMenuAdapter).addItems(it)
})
presenter.loaderLiveData.observe(this, Observer {
         if(it) showLoader()
else hideLoader()
})
```

```
presenter.message.observe(this, Observer {
toast(it)
})

presenter.getMenu(restaurant.id)
    }

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
super.onActivityResult(requestCode, resultCode, data)
if(requestCode == CheckoutActivity.ORDER_REQUEST_CODE) {
            data?.let {
                val order: Order = data.getParcelableExtra(CheckoutActivity.ORDER)
cart = order.items
(rvMenu.adapter as RestaurantMenuAdapter).addOrderItems(cart)
                refreshCart()
}

}
    }

private fun refreshCart() {
if(cart.isNotEmpty()) {
            btnCheckout.visibility = VISIBLE
if(cart.size == 1) {
                btnCheckout.text = "${cart.size} item"
} else {
                btnCheckout.text = "${cart.size} items"
}
            (rvMenu.adapter as RestaurantMenuAdapter).addOrderItems(cart)
        } else {
            btnCheckout.visibility = GONE
}
    }
}
```

For Restaurant Interface:

Add Menu Item Activity:

```
package com.example.restaurant.ui.addmenuitem

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
import androidx.lifecycle.Observer
import com.bumptech.glide.Glide
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.theartofdev.edmodo.cropper.CropImage
import com.theartofdev.edmodo.cropper.CropImageView
import kotlinx.android.synthetic.main.activity_add_menu_item.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast

class AddMenuItemActivity : BaseActivity() {

private val presenter = AddMenuItemPresenter()
var uri : Uri? = null

   override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_add_menu_item)
      setActionBar(toolbar, true)
      ivFoodItem.onClick {
CropImage.activity()
            .setGuidelines(CropImageView.Guidelines.ON)
            .start(this@AddMenuItemActivity)
}

btnAddItem.onClick {
         presenter.setMenuItem(etFoodName.text.toString(), etFoodPrice.text.toString(),
            etFoodDescription.text.toString(), uri)
}
      presenter.menuAdded.observe(this, Observer {
         if(it) finish()
})
presenter.loaderLiveData.observe(this, Observer {
         if(it) showLoader()
else hideLoader()
})
presenter.message.observe(this, Observer {
toast(it)
})
presenter.descriptionError.observe(this, Observer {
tilFoodDescription.error = it
      })
presenter.nameError.observe(this, Observer {
tilFoodName.error = it
      })
presenter.priceError.observe(this, Observer {
```

```
tilFoodPrice.error = it
        })

    }

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
if(requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
try {
val result = CropImage.getActivityResult(data)
uri = result.uri
if (uri != null) {
                Glide.with(this@AddMenuItemActivity).load(uri).into(ivFoodItem)
            }
        }catch (e: Exception) {
            e.printStackTrace()
        }
    }
else super.onActivityResult(requestCode, resultCode, data)
    }
}
```

Edit Menu Item Activity:

```
package com.example.restaurant.ui.editmenu

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.lifecycle.Observer
import com.bumptech.glide.Glide
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.model.FoodItem
import com.theartofdev.edmodo.cropper.CropImage
import com.theartofdev.edmodo.cropper.CropImageView
import kotlinx.android.synthetic.main.activity_edit_menu_item.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.toast

class EditMenuItemActivity : BaseActivity() {

companion object {
const val FOOD = "food"
}
```

```
private val presenter = EditMenuItemPresenter()
var uri : Uri? = null

   override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_edit_menu_item)
      setActionBar(toolbar, true)
      ivFoodItem.onClick {
CropImage.activity()
            .setGuidelines(CropImageView.Guidelines.ON)
            .start(this@EditMenuItemActivity)
}

      val food = intent.getParcelableExtra<FoodItem>(FOOD)

Glide.with(this).load(food.photoUrl).placeholder(R.drawable.ic_add_white_with_padding).into
(ivFoodItem)
      etFoodName.setText(food.name)
      etFoodPrice.setText(food.price.toString())
      etFoodDescription.setText(food.description)

      btnEditItem.onClick {
         presenter.editMenuItem(food.photoUrl, etFoodName.text.toString(),
etFoodPrice.text.toString(),
            etFoodDescription.text.toString(), uri, food.id)
}
      presenter.menuAdded.observe(this, Observer {
         if(it) finish()
})
presenter.loaderLiveData.observe(this, Observer {
         if(it) showLoader()
else hideLoader()
})
presenter.message.observe(this, Observer {
toast(it)
})
presenter.descriptionError.observe(this, Observer {
tilFoodDescription.error = it
      })
presenter.nameError.observe(this, Observer {
tilFoodName.error = it
      })
presenter.priceError.observe(this, Observer {
tilFoodPrice.error = it
      })
```

```
    }

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
if(requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
try {
val result = CropImage.getActivityResult(data)
uri = result.uri
if (uri != null) {
                Glide.with(this@EditMenuItemActivity).load(uri).into(ivFoodItem)
            }
        }catch (e: Exception) {
            e.printStackTrace()
        }
    }
else super.onActivityResult(requestCode, resultCode, data)
    }
}
```

Restaurant “My menu” Fragment:

```
package com.example.restaurant.ui.mymenu

import android.content.Intent
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.GridLayoutManager
import com.example.restaurant.R
import com.example.restaurant.base.BaseFragment
import com.example.restaurant.model.FoodItem
import com.example.restaurant.ui.addmenuitem.AddMenuItemActivity
import com.example.restaurant.ui.editmenu.EditMenuItemActivity
import kotlinx.android.synthetic.main.fragment_my_menu.*
import org.jetbrains.anko.sdk27.coroutines.onClick
import org.jetbrains.anko.support.v4.toast

class MyMenuFragment : BaseFragment() {

private val presenter = MyMenuPresenter()

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)

presenter.loaderLiveData.observe(this, Observer {
```

```
        if(it) showLoader()
else hideLoader()
})
presenter.message.observe(this, Observer {
toast(it)
})
    }

override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
returninflater.inflate(R.layout.fragment_my_menu, container, false)
    }

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
super.onViewCreated(view, savedInstanceState)

        rvMenu.layoutManager = GridLayoutManager(context, 2)
        rvMenu.adapter = MenuAdapter(object: MenuAdapter.Callback{
override fun onEditButtonClick(food: FoodItem) {
val intent = Intent(context, EditMenuItemActivity::class.java)
            intent.putExtra(EditMenuItemActivity.FOOD, food)
            startActivity(intent)
        }

        })
        btnAddItem.onClick {
            val intent = Intent(context, AddMenuItemActivity::class.java)
            startActivity(intent)
}

        presenter.menu.observe(this, Observer {
(rvMenu.adapter as MenuAdapter).addItems(it)
})

    }

override fun onResume() {
super.onResume()
presenter.getMyMenu()
    }
}
```

Restaurant Order Details Activity:

```
package com.example.restaurant.ui.orderdetails

import android.os.Bundle
import android.os.PersistableBundle
import android.view.View.INVISIBLE
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.example.restaurant.model.Order
import com.example.restaurant.utils.ACTION_STATUS
import kotlinx.android.synthetic.main.activity_restaurant_order_details.*
import org.jetbrains.anko.sdk27.coroutines.onClick

class RestaurantOrderDetails : BaseActivity() {

companion object {
const val ORDER = "order"
}

var order: Order? = null
   private val presenter = RestaurantOrderDetailsPresenter()

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_restaurant_order_details)
      setActionBar(toolbar, true)

order = intent.getParcelableExtra(ORDER)
      rvItems.layoutManager = LinearLayoutManager(this)


order?.let { order ->
tvName.text = order.client.name
tvPhoneNo.text = order.client.phoneNo
tvOrderId.text = order.id.toString()
         rvItems.adapter = RestaurantOrderItemsAdapter(order.items)
         tvSubTotal.text = order.subTotal.toString()
         tvVat.text = (order.total - order.subTotal).toString()
         tvTotal.text = order.total.toString()
if (order.status <ACTION_STATUS.size - 2) {
            btnOrderNextStatus.text = ACTION_STATUS[order.status + 1]
         } else {
            btnOrderNextStatus.visibility = INVISIBLE
btnOrderCancel.visibility = INVISIBLE
```

```
}
}

        presenter.loaderLiveData.observe(this, Observer {
            if(it) showLoader()
else hideLoader()
})

presenter.updatedOrder.observe(this, Observer {
            order = it
            if(it.status <ACTION_STATUS.size-2) {
                btnOrderNextStatus.text = ACTION_STATUS[it.status + 1]
            } else {
                btnOrderNextStatus.visibility = INVISIBLE
btnOrderCancel.visibility = INVISIBLE
}
})

        btnOrderCancel.onClick {
            order?.let{
                presenter.updateOrderStatus(it, 5)
}
        }

btnOrderNextStatus.onClick {
            order?.let {
presenter.updateOrderStatus(it, it.status + 1)
}
        }

}
}
```

QR Code:(QR Code Generate Fragment):

```
package com.example.restaurant.ui.qrcode

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidmads.library.qrgenearator.QRGContents
import androidmads.library.qrgenearator.QRGEncoder
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
```

```
import com.example.restaurant.base.BaseFragment
import com.example.restaurant.data.pref.AppPreference
import kotlinx.android.synthetic.main.activity_qr_code.*
import org.jetbrains.anko.imageBitmap
import org.jetbrains.anko.support.v4.dimen
import java.lang.Exception

class GenerateQrCodeFragment : BaseFragment() {

private val session = AppPreference()


override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
returninflater.inflate(R.layout.activity_qr_code, container, false)
    }

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
super.onViewCreated(view, savedInstanceState)

val qrgEncoder = QRGEncoder(session.id, null, QRGContents.Type.TEXT,
dimen(R.dimen._200sdp))
try{
val bitmap = qrgEncoder.encodeAsBitmap()
            ivQrCode.imageBitmap = bitmap
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }
}
```

QR Code : (Scan Activity):

```
package com.example.restaurant.ui.qrcode

import android.Manifest
import android.os.Build
import android.os.Bundle
import com.example.restaurant.R
import com.example.restaurant.base.BaseActivity
import com.google.zxing.BarcodeFormat
import com.google.zxing.Result
import kotlinx.android.synthetic.main.activity_scan.*
```

```
import me.dm7.barcodescanner.zxing.ZXingScannerView
import org.jetbrains.anko.toast
import com.karumi.dexter.PermissionToken
import com.karumi.dexter.listener.PermissionDeniedResponse
import com.karumi.dexter.listener.PermissionGrantedResponse
import com.karumi.dexter.listener.single.PermissionListener
import android.Manifest.permission
import android.app.Activity
import android.content.Intent
import com.karumi.dexter.Dexter
import com.karumi.dexter.listener.PermissionRequest


class ScanActivity: BaseActivity(), ZXingScannerView.ResultHandler {


override fun handleResult(p0: Result?) {
if(p0 != null) {
val intent = Intent()
        intent.putExtra("ID", p0.text)
        setResult(Activity.RESULT_OK, intent)
        finish()
    }
  }

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_scan)
  }

override fun onResume() {
super.onResume()

    Dexter.withActivity(this)
        .withPermission(Manifest.permission.CAMERA)
        .withListener(object : PermissionListener {
override fun onPermissionGranted(response: PermissionGrantedResponse) {/* ... */
qrCodeScanner.startCamera()
            qrCodeScanner.setResultHandler(this@ScanActivity)
        }

override fun onPermissionDenied(response: PermissionDeniedResponse) {/* ... */
            toast("Can not access camera")
            finish()
        }
```

```
override fun onPermissionRationaleShouldBeShown(
                permission: PermissionRequest,
                token: PermissionToken
            ) {
                token.continuePermissionRequest()
            }
        }).check()


    }

private fun setScannerProperties() {
        qrCodeScanner.setFormats(listOf(BarcodeFormat.QR_CODE))
        qrCodeScanner.setAutoFocus(true)
        qrCodeScanner.setLaserColor(R.color.colorAccent)
        qrCodeScanner.setMaskColor(R.color.colorAccent)
    }
}
```

Dialogs (Two Types used) :

Loader:

```
packagecom.example.restaurant.ui.dialogs

import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.DialogFragment
import com.example.restaurant.R

class Loader : DialogFragment() {

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        setStyle(DialogFragment.STYLE_NO_FRAME, R.style.MyAlertDialogTheme)
isCancelable = false
}

override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
returninflater.inflate(R.layout.dialog_loader, container, false)
```

```
    }

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
super.onViewCreated(view, savedInstanceState)

dialog?.window?.setBackgroundDrawable(ColorDrawable(Color.parseColor("#00000000")))
    }
}
```

Log Out Dialog:

```
package com.example.restaurant.ui.dialogs

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.DialogFragment
import com.example.restaurant.R
import kotlinx.android.synthetic.main.dialog_logout.*
import org.jetbrains.anko.sdk27.coroutines.onClick

class LogoutDialog(val callback: Callback) : DialogFragment() {

override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
returninflater.inflate(R.layout.dialog_logout, container, false)
    }

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
super.onViewCreated(view, savedInstanceState)

        btnCancel.onClick {
dismiss()
}
btnLogout.onClick {
            callback.onLogoutClick()
            dismiss()
}
}
interface Callback {
fun onLogoutClick()
    }
}
```