Expertise
and insight
for the future

Shiva Tiwari, Sandeep Thapa

Android Application, Ours

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Information Technology

Date: 15 March 2019

Author(s) Title Number of Pages Date	Shiva Tiwari & Sandeep Thapa Android Application, OURS 49 pages + appendices 15 MARCH 2019
Degree	Bachelor's Degree
Degree Programme	Software Engineering
Specialisation option	Software Engineering
Instructor(s)	Janne Salonen

The aim of this thesis was to build an E-commerce android application and document step by step guidelines and findings which would provide insight into an android application development. It helps developers to know the basic requirements and knowledge for developing an android application.

Smartphones are the most widely used consumer electronic devices. Internet connection has made it much simpler for people to browse anything through handheld mobile devices. In addition, people buy services and make payments to each using smartphones. Keeping these things on mind, an android application, 'Ours', was developed to connect businesses and customers for easy buying and selling of goods and services. Also, realizing the impact of online advertisements Ours' application has been integrated with advertisements for uplifting sales of business and giving users the ability to get discount coupons in the application instead of filling up space in email. Ours application also has integrated google map API.

Businesses using this application will have features where they can add employees (sellers, delivery person), create and manage shelf products, share free coupons among customers and allow them to share it among friends. Also, communication between customers and sellers has been made easy by adding messaging service which helps them connect between each other. Ours' application can also act as a social media application inside a e-commerce application since it has all the features of a social media app (sharing stories, messaging and connecting between friends)

Ours' application was developed using Android Studio and Java programming language. Firebase was used for storing and retrieving data since it is a real time database. It has built in QR code features from Zxing which helps to keep track on the orders and business's sales. Google Maps from Google has added more features within this application.

The purpose of the author is to implement his study and findings to build a fully functioning e-commerce android application. Through the Author's findings, any developer can build a simple and fully working android application.

Keywords	Firebase Real time Database, Google Maps APIs, Android Studio SDK, Java programming Language, QR code

Contents

Abbreviations

QR Code	Quick Response Code
IDE	Integrated Development Environment
API	Application Programming Interface
OS	Operating System
APK	Android Application Package
SDK	Software Development Kit
TV	Television
XML	Extensive Markup Language
GPL	General Public License
JSON	JavaScript Object Notation
UI	User Interface

1 Introduction

Smartphones and internet have made people's life easier these days. Majority of the people owning smartphones spend most of their time surfing through internet. Due to easy access to internet, online shopping also has gained its value over time. Many businesses have succeeded selling their products through the means of internet. Websites and applications are the main platforms which are used for online shopping. [1] Considering these factors, a prototype e-commerce android application, Ours' is developed that helps business like restaurants, cafes, night clubs, salons and grocery stores, to advertise and sell their products to their customers. These businesses often use cash registers, terminals for keeping records on sales products and make safe payments. However, cash registers and terminal could be replaced by smartphones if smartphones are equipped with essential infrastructure for handling mobile payments and keeping track on products. [2]
Focusing on these issues, new version of advertisements via an easily reachable android application, 'Ours', is built. This application has many useful features that will help business to reach their potential customers in addition provide other useful services to them. The author's choice for developing an android application is due to larger market share of Android devices compared to competing platforms.

2 'Ours' Application

Ours' application is an android application built using Java programming language. Business using this application can create their own list of items they want to sell, add employees from Ours' social circle, create working time for employees and create sales coupons. Whereas, on the consumers' benefits, consumers can see business around them, view sales products and reserve time if available. Consumers are able to buy products within few clicks. Moreover, they can easily view and buy products by scanning QR code of specific business either from employees' phones, business master phone, or any printed copy that has QR code representing any specific services provided by an entity. On other hand, employees are given free hands on their mobile. They can choose items they prefer to sell most. Whenever a consumer makes an order employee can preview the orders individually after getting access from business administration. Then, after the order is checked for preferred delivery method i.e. either the customer wants it to get delivered or wants to pick up. When it is time to hand over the order to the right customer, the delivery person can easily scan the QR code of the order from customer's

smartphone and complete the order. The payment is real time so both employee and consumer are notified within a fraction of second. The payment can be done via credit cards, google pay (if possible), or cash as per business terms and services.
Consumers can follow businesses and get notified if any offers are made by them. This application also uses Google Maps, which is integrated in the application to make interface for business to distribute coupons on maps. Consumers can walk nearby and collect the coupons distributed by business registered in application. Coupons obtained must be used as specified in the advertisement offered by the business as they have a fixed quantity of items that can be bought, and all the coupons also have an expiry date. So, the coupons obtained during the advertisement campaigns must be used before the expiry date.

Ours' application also has a social circle platform integrated inside the main application that helps registered users connect between themselves. Also, users can share their stories on their social page. Furthermore, users can add friends from Ours' social circle and share the coupons they have gathered.

3 Tools

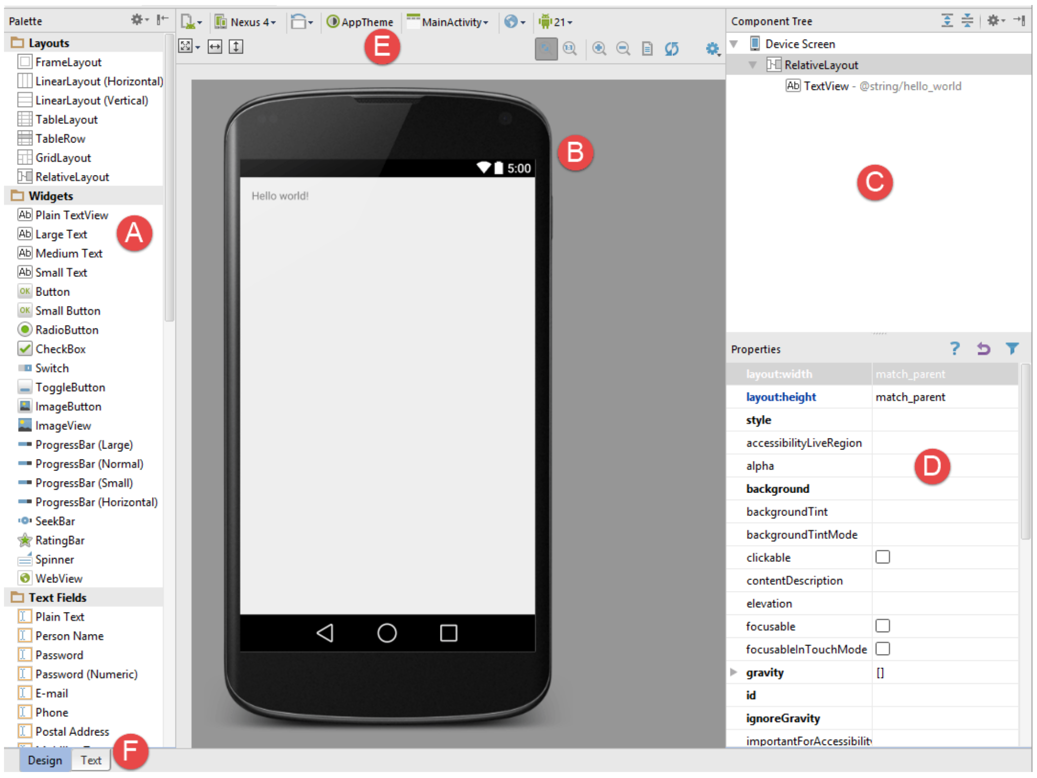
Android applications are written using Kotlin or JAVA programing language. The IDE used for developing Ours' application is Android Studio provided by Google. Moreover, developers can use other tools like Unity and Cordova for building an android application. The thesis presented focuses on developing a native android application using Java, Firebase Real time database and multiple Google APIs whose functions are described later in the report. Brief description about the tools used ,their functionalities and the methods which are carried out while using these services are explained in their respective fields.

3.1 Android studio

Android studio is the official IDE developed by Google to develop applications for android operating system. Developers can get free access to android studio from google developer site. The developers can easily download and install android studio depending upon developer's operating system preference from the Google developer portal. Android Studio is available for Linux, Mac OS, and Windows. The pre-requisites for the installation can be found in the Google developer site. The latest version of android studio available for developers is 3.3. It comprises of many useful features like, visual layout editor, APK analyzer, fast emulator, intelligent editor, flexible build system, real time profilers and numerous others. Android SDK also provides necessary tools and APIs to build an android application. Developers can develop applications not only for android smartphones but also develop application for other android devices such as Wear OS and android TV. All the APIs that are needed for developers are provided by Google. [3]

Android studio provides Gradle based build support for android application development which helps users develop and refactor application allowing quick bug fixes. Developers can also choose from multiple layouts and templates provided by android studio to start a simple android application and learn the basics of application development. Android studio also provides emulators or android virtual device to run and test the application created without having to connect to a smartphone for testing purposes.

The home layout of android is shown in figure 1.



layout:width match_parent
layout:height match_parent
style
accessibilityLiveRegion
alpha
background
backgroundTint
backgroundTintMode
clickable ☐
contentDescription
elevation
focusable ☐
focusableInTouchMode ☐
gravity []
id
ignoreGravity
importantForAccessibility

Figure 1. Android Studio homepage. Reprinted from techtopia.com. [4]

The figure 1 is an example of Android studio work space view. The workspace contains android project structure and layout editor. The layout editor contains palette, component tree, and editor for component’s attributes. Different components of android studio workspace marked above have their own functionalities.

A. Palette

   The palette consists of various range of view components provided by android SDK. Developers can easily access to the items which are listed in different categories. Items are dragged and dropped positioning on the editor layout. [3]

B. Device Screen

   Device screen provides visual representation of user interface layout as it is being designed. It contains the screens of the android devices available in android studio which can be changed from the toolbar. [3]

C. Component Tree

   Component tree represents a set of hierarchical data provided by android studio. Developers can drag and drop view components from palette to the component tree to build a better layout. Selecting an element from the component tree will show the corresponding view in the layout. [3]

D. Properties

   All the component views which are listed in the palette has their own set of properties. These properties help to adjust the behavior and appearance of the view. Properties panel gives access to the selected view in the layout allowing changes to be made. [3]

E. Toolbar

   The toolbar provides access to the wide range of options such as zoom in and zoom out, changing the android devices screen, changing the layout to landscape or portrait mode. On selecting any view from the layout, toolbar also provides different sensitive buttons which is relevant to the view. [3]

F. Mode Switching Tabs

   The tabs are located along the lower edge of the Designer view. There are two tabs one with design tab and text tab. Design tab contains the view which looks like the mobile screen. On other hand, text tab contains the view where developers can create and add view components using xml.

3.2 Extensible markup language

XML is a markup language which is both human readable and machine readable. Developers can create XML blocks for data to be used in an application that can be toggled during the application runtime. There are no standard rules or tags that should be used for creating XML documents. XML tags are created by developer according to the data that must be used during the application development and deployment. During the development of an android application creating multiple layouts are carried out using xml. These layouts are inflated over the activity or fragment by using Java in application. [5]

An example of creating layout in android studio using XML is as shown below.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/an-
droid"

              android:layout_width="match_parent"

              android:layout_height="match_parent"

              android:orientation="vertical" >

    <TextView android:id="@+id/text"

              android:layout_width="wrap_content"

              android:layout_height="wrap_content"

              android:text="Hello, I am a TextView" />

</LinearLayout>
```

Listing 1. XML layout in android studio

XML document is created with a tag that starts with < and ends with >. In the above listing Linear layout and TextView are tags that have both opening and closing tags. Attributes are added inside tags that includes the data to be stored in an XML documents.

An example of inflating the xml layout by using java code is as shown below.

```
LayoutInflater inflater = (LayoutInflater) context.getSystem-
Service(Activity.LAYOUT_INFLATER_SERVICE);

View discountitemesview = inflater.inflate(R.layout.fragment_com-
pany_discount_normal_items, null);
```

Listing 2. XML layout for java code inflation

In the above listing the layout defined in the layout file is inflated to the android application by using the LayoutInflater object.

3.3 JAVA Programming Language

Java Programming language is an Object-Oriented Programming language which was developed by Sun Microsystems in 1995. In November 13, 2006 Sun released the Java programming language as an open source and made it free for developers under the terms of GNU General Public License(GPL). Java is most preferred programming language to build android applications for wide ranges of android devices. Java codes run on Java Virtual Machine. [6]

Although Java Programming language is the most widely used programming language, there are other programming languages which can be used to build android applications for android devices. Kotlin, C/C++, C# are few other programming languages which are used by android developers. In addition to that, android applications can be developed by using cross-platform JavaScript frameworks like Apache Cordova and PhoneGap.

3.4 Firebase

Firebase is a mobile and web application platform developed by Firebase in 2011 which was later acquired by Google in 2014. It was first developed to ensure availability of online data- base to the user for better development of application and put the products into effects easily. It is growing huge in market place for application development due to its special features for handling database in the server. Developers using firebase database in their application, don’t have to worry about server side coding. The firebase comprises of real time database system which helps developers to simply focus on building better user experiences.[7] Mobile applications like Shazam, Fabulous, Sky scanner, and various other applications are built using this real time database.

3.4.1 Firebase products

Firebase is preferred by android application developers due to its features like Analytics, Cloud Messaging, User Authentication, Real time Database, Storage, Hosting, Remote Configure, Test lab, Notification, App Indexing among others. These features of firebase make application development life cycle easier compared to traditional database management systems. [7]

Figure 2. illustrates the different services provided by firebase for application developers.

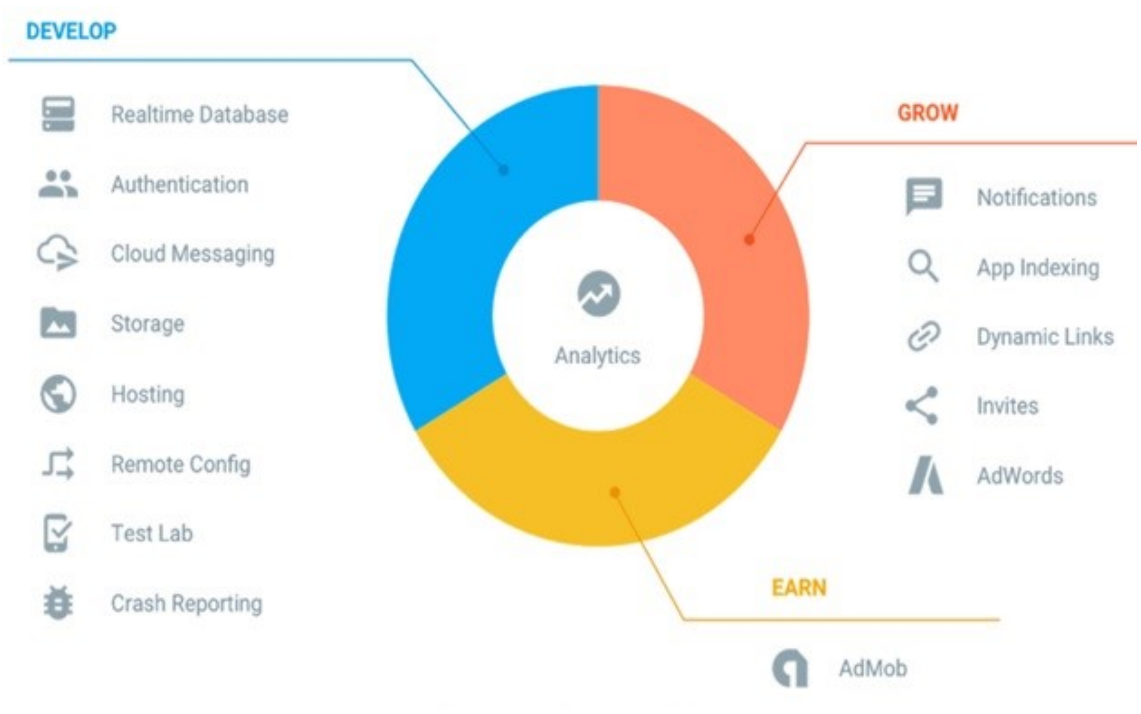


Fig 2. Firebase services

As shown in figure 2 firebase offers multiple services including real-time database with direct link to application which enables developers to store and retrieve data simultaneously. Other services include authentication service and hosting web pages and applications. Also, it offers dashboard designed for viewing the application user analytics for gathering data.

3.4.2 Handling Firebase Database

Firebase database is easy to handle as it allows developers to create applications without server-side scripting. Likewise, all the user data is stored in easy to read JSON format. Developers can easily access these data in real time database using simple queries and get the response in JSON format. [8]

The code for referencing the firebase database is shown below .

```
DatabaseReference mFirebaseDatabaseReference = FirebaseDatabase.get-
Instance().getReference();
```

Listing 3. Linking firebase database to application.

In the above listing DatabseReference is a function defined by Firebase which connects the database from firebase API to application being created in android studio. Using the above function an instance of database can be created in application for further use by referencing mFirebaseDatabaseReference variable.

An example for storing data to the database reference is described in code listing below.

```
mFirebaseDatabaseReference.child("example").setValue("string");
```

Listing 4. Storing string data to firebase database.

The code above stores string value to the “example” child in firebase database by using reference variable created in Listing 3. In the listing mentioned above custom build objects can also be stored to the reference database.

Retrieving the stored data from firebase database, developer can use below code to get the data in JSON format.

```
mFirebaseDatabaseReference.child("example").addValueEventLis-
tener(newValueEventListener(){
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot){
if(dataSnapshot.exists()){
String value=dataSnapshot.getValue(String.class)}
```

```
 }
@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
 }
});
```

Listing 5. Retrieving data stored in database.

In the above listing the data that was saved using mFirebaseDatabaseReference can be retrieved using datasnapshot.getValue(data).The predefined function will then iterate through the database to find the data to be retrieved .If the data is not found in the database than the function returns database error to the user .

Images and video files related to the application can be stored directly to the firebase storage without the requirement of additional file servers by using following code.

```
StorageReference filePath = FirebaseStorage
.getInstance()
.getReference()
.child("Images")
.child("image.jpg);

filePath.putFile(uri)
.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnap-
shot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
 }
```

Listing 6. Storing images and videos in firebase database.

Developers can upload images and videos to firebase storage using its Storage reference. The mentioned above listing is an example of uploading image file to the firebase storage. The file that is being stored in the storage should be specified with file extension. If the file link is found, then it is stored and saved in database and if the file location is not found then the application will display error relating to the fie location not specified.

3.5 Google Maps APIs

Google provides open source APIs for android developers which can be integrated in android application. Google maps is one of the API provided by Google. This API is the industrial standard for geolocation and map-based applications for web and mobile development. Google has been constantly developing and updating their Map API. As a result of continuous development and additions Google Maps is one of the most resourceful map APIs. Furthermore, Google provides Direction APIs, Places APIs, and many more APIs as open source for developers to use in their mobile applications. These additional APIs make it simpler for developers to deliver efficient and accurate geolocation-based applications. Ours' application has also integrated Google maps for better user experiences regarding business location and direction for users to find restaurant or any other service providers.
Developers willing to integrate Google maps in their application should get API key for Google map. Developers can register their developer account and get access to APIs key in the Google Application Console. [9]

4 Android Application Components

Application components are building blocks for an android application. The application manifest file, AndroidManifest.xml, consists of these components interacting throughout the application. The four major components of the android application are Activities, Services, Broadcast Receivers, and Content Providers. [3]

4.1 Activities

An android application consists one or more activities. An activity represents a single screen containing user interface. If application has multiple activities, a specific activity is chosen as application launcher activity. Activity often contains fragments or view layouts which tracks on users' actions and inflate views to screen. Views in android application are created in XML and inflated using java code. Whenever a user opens an android application, the activity goes through multiple cycles. When an activity is created in an android application , the runtime environment for that activity is started and when the processes of an activity is completed the life cycle of that activity is killed . [10]

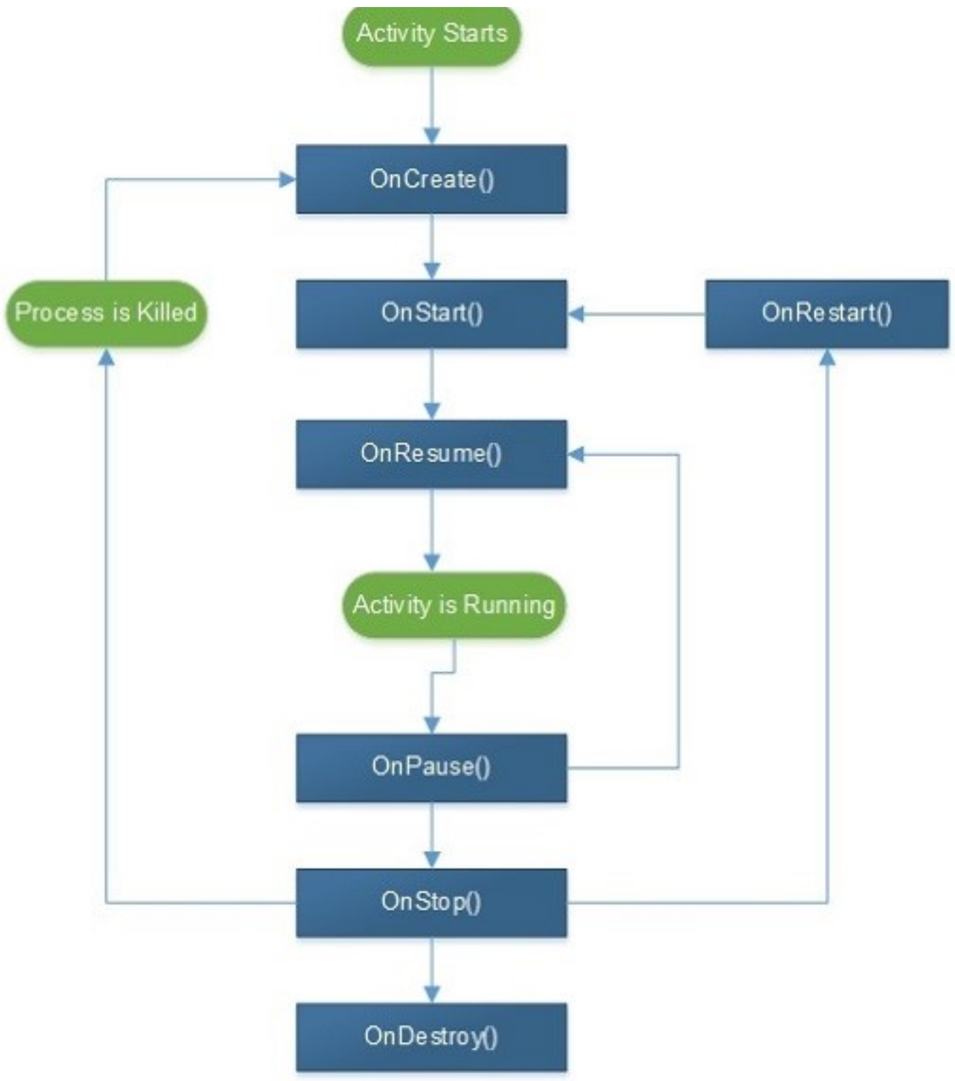


Figure 3. Life cycle of an activity in android application. Reprinted from tutorialspoint.com. [10]

In the figure above when an activity is created, the activity calls onCreate() callback function inflating the view associated with the activity. After on create function, the activity calls OnStart() function which helps the activity to be displayed to the user . Unless user minimizes or closes the application, the activity life cycle doesn’t change. If the user minimizes the application, activity still run in the background calling onPause() callback function. If the user reopens the application the activity calls onResume(). If user closes the application and the activity is no more visible to the user, the activity calls onStop() callback function. OnDestroy() callback function is called before the onStop() callback functions. OnDestroy() callbacks function removes all the resources and content of the view before destroying the activity.

4.2 Services

Service is a component of android application for updating the data sources, activities, triggering notifications and other broadcast intents. These components run in the background so even the user is navigating through different views, the background operations run smoothly without hindering the user experience. [11]A service class which is subclass of Service is presented in appendix 1. under services class.

4.3 Broadcast Receivers

Broadcast Receivers are also known as intent listeners. These components help an application to listen the intent and respond to the messages. A broadcast receiver class is a subclass of Broadcast Receiver. An example of usage of the Broadcast Receiver component is described in the appendix 1 under broadcast receivers title.[12]

4.4 Content Providers

Content Providers are android components which deals with data supply beyond the application boundaries. Content providers manage data such as user information and various media files such as pictures and videos used in or served by the application. A subclass of Content Provider is shown in appendix 1. With content providers class heading.[12]

5 Structure of Application

5.1 Ours' application work flow

The main idea behind the development of Our's application is to make the connection between businesses and customers safe efficient and fast. The main working principle of Our's application is shown in figure below.

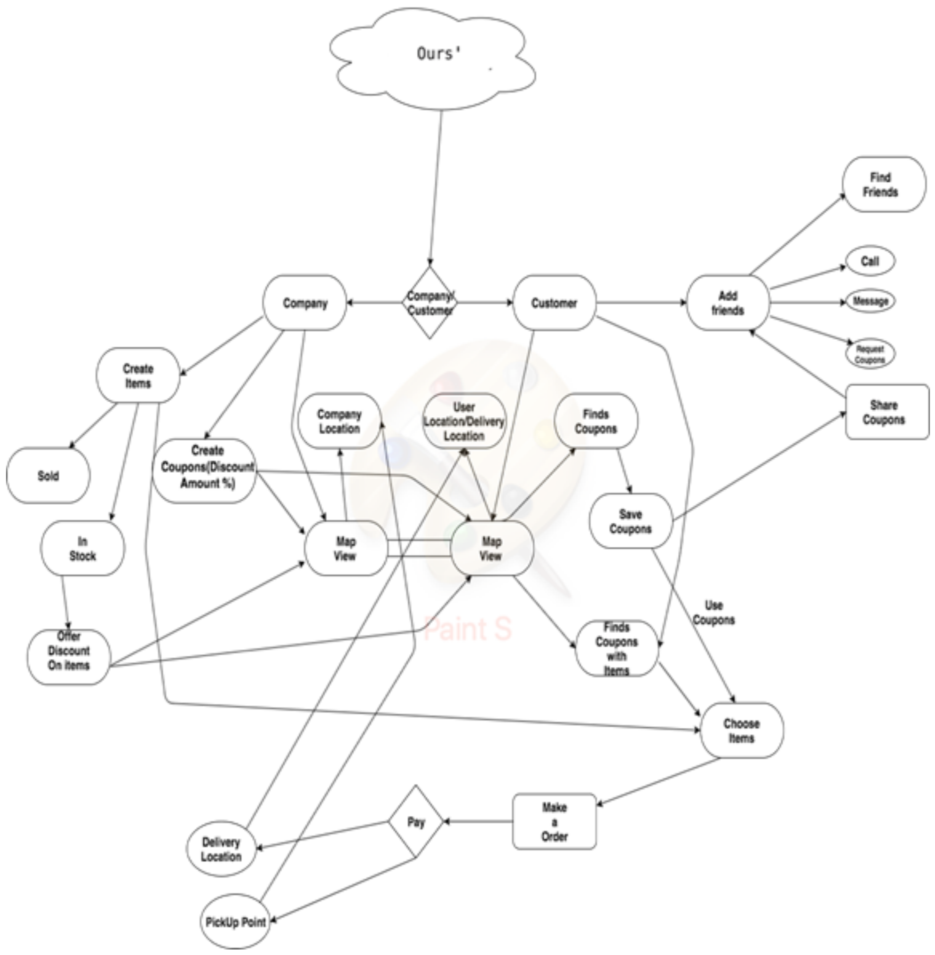


Figure 4. Our's application work flow diagram.

The figure 4 illustrates methodology to connect many businesses with customers by the application. The application is also an advertising platform for companies which focuses on advertising their products directly to consumers by using common advertisement platform. Different categories of the application shown in the figure will be explained in detail in their own headings .

6 Ours' Application User Interface

Ours' application consists two login interfaces , one for customer and the other for business sector. The figure below is the home screen of the application.

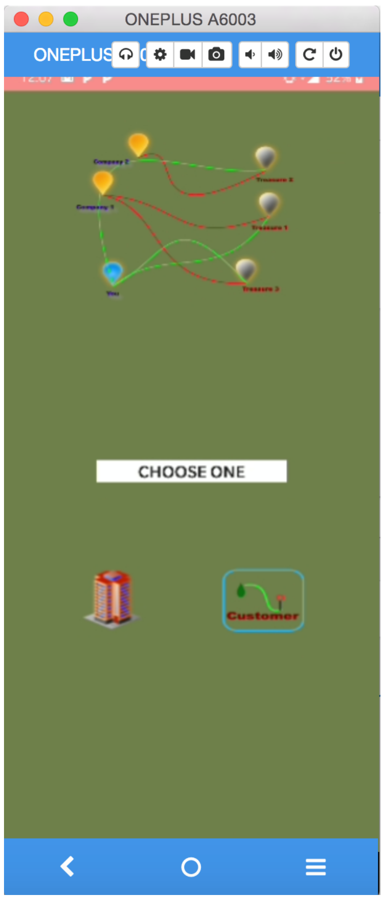


Figure 5. Homepage of Our's application.

In the above figure of Our's application homepage the user can choose from two of the options available. One is for businesses and the other for customers or normal users. When the user clicks in one of the options then they are redirected to login page .Users are able choose either of the interface based on their roles and requirements. Both users have their own views to work with which are created to make the user experience different and efficient for different customer groups. Every user in this platform are given their own unique ids along with a unique QR code. The code is used throughout the application as a user's identity.

The figure 7 shows an example of QR code used in application

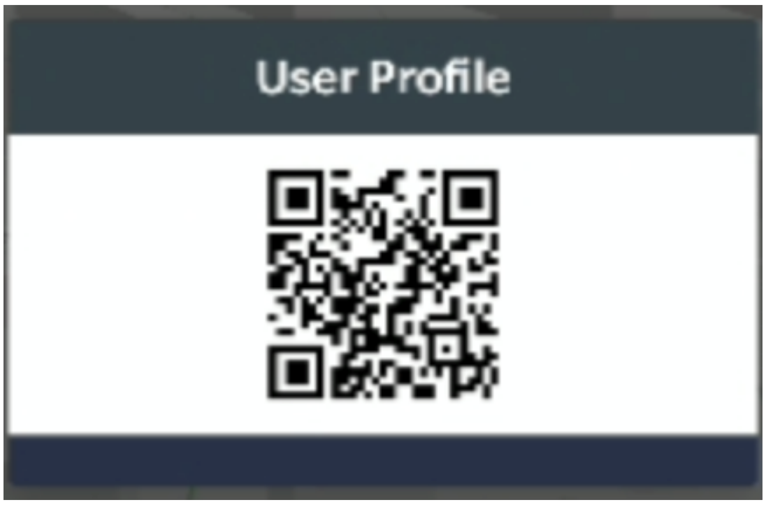


Figure 6. QR code of a user profile

The QR code in this application is created using Zxing. The library is listed as dependeny in the gradle file of the application as shown below.

```
repositories {
jcenter()
}
dependencies {
 implementation 'com.journeyapps:zxing-android-embedded:3.0.2@aar'
 implementation 'com.google.zxing:core:3.2.0'
}
```

Listing 7. Dependeny of a gradle file for using Zxing.

This library helps to embed string value or any url to the QR code. An example for embedding a string value inside QR code is shown below.

```
MultiFormatWriter multiFormatWriter = new MultiFormatWriter();
try {
    BitMatrix bitMatrix = multiFormatWriter.encode("String Value",
BarcodeFormat.QR_CODE, 300, 300);
    BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
    Bitmap bitmap = barcodeEncoder.createBitmap(bitMatrix);
    ImageView barcode=findViewbyId(R.id.barcode);
    barcode.setImageBitmap(bitmap);
} catch (WriterException e) {
    e.printStackTrace();
}
```

Listing 8. Code for embedding a string value to a QR code.

The listed code above creates an instance of MultiFormatWriter. Bitmatrix creates QR code with suitable size embedding string value within. Then, the QR with the string value changed into bitmap format and is placed over an image view name barcode. Barcode is an image view which is created in repective layout where the QR has to be shown.

6.1 Business User Interface

6.1.1 Sign Up and Log In

Business users who are willing do business with Ours' application need to provide enough information before signing up. After filling up full information about the company, Ours' application will direct the user towards its respective User Interface. Addressing the obvious privacy concerns the application only collects and stores vital user information of both business customers and end users. It collects data like company name, company address, E-mail address and password which is shown in figure below.

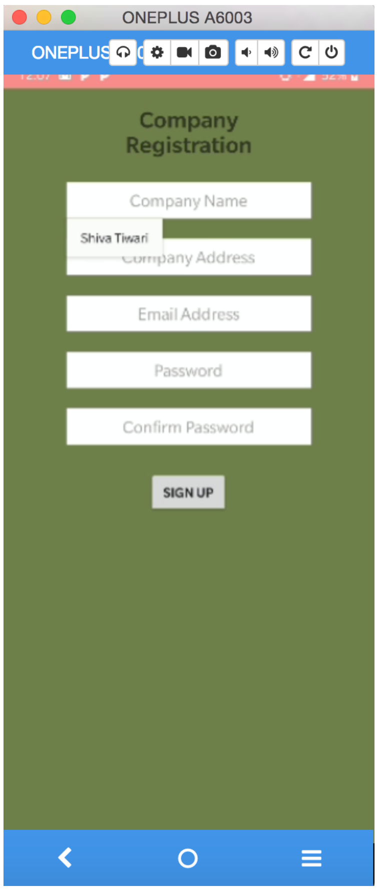


Figure 6. Company registration page of Ours' application.

In the figure above the user registration for company is done. For the company to be registered all the data required in respective field have to be filled and when the Sign-Up button is pressed the data is then sent to the database and a user account is created. The code through the data is sent and stored in database is given in appendix 2 under sign up heading.

The sign-up process creates an account after which the user is directed to its own user interface. Whereas, if the user has already signed up in Our's application before and they have their login credentials then they can easily log in using their own username and password. The figure below shows the sign in screen for business users for login.

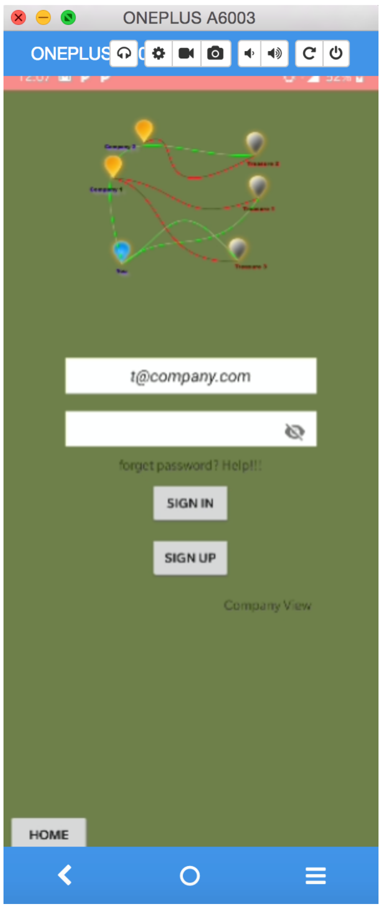


Figure 7. Login page of Ours’ application for businesses.

In the figure above a user can sign in and use the services of application. User needs to provide the username and password created during the sign up process and can log in to the services. Also, if a user doesn’t have required credentials then they can click on sign up button and create an account and log in from this page. The working principle is shown in appendix 2 as login heading.

6.1.2 Home View

Service providers or businesses can get to their home page after successful log in . Home screen comprises of many buttons and features with their own respective functionalities. The home view for businesses is shown in figure below.

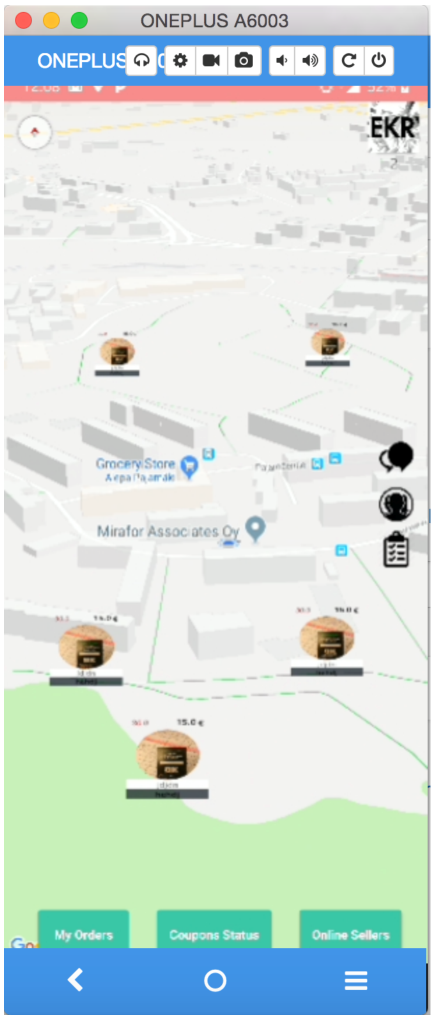


Figure 8. Homepage for service providers.

The figure above shows the business home page. The page consists of a map which has been integrated using google map API. In the map the user can also see the coupons and advertisements that they have distributed which can be collected by customers . Also, from the buttons provided in homepage the user can see the orders they have received and can process them for pick up or delivery.

6.1.3 Managing Products

Business user who have registered and are selling items through Ours' application can create their own items. To create an item company logo or name in the home view has to be clicked , after which a dialogue bx appears with my items on top of the list .There is also an option t add item in the box which can be clicked and the dialogue box or creating a new item appears. To create an item, business user should provide information such as item name, item value, item description, item stocks, and image file that represent the item to be displayed in the application. The required information can be supplied by filling up a simple form in the application. The figure below shows the view where user can create their own item.

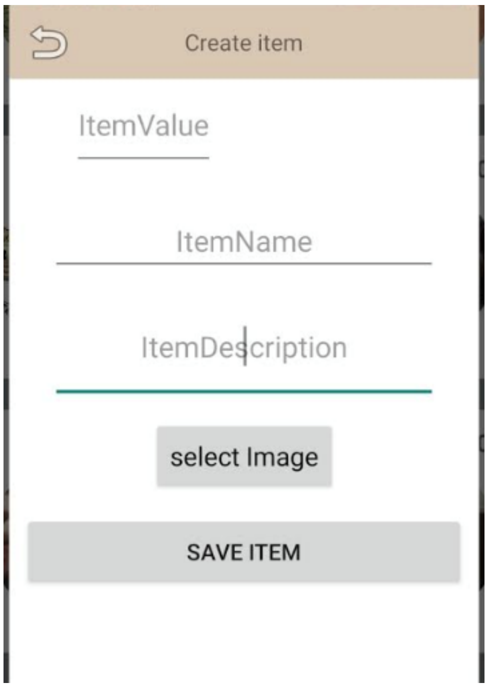


Figure 9. Add item dialogue box.

In the figure shown above after filling the information requested an item in the store is created. After the item is created it gets assigned a unique item identifier and a unique QR code which can be used instead of the details of the item for further use of item by consumers.

The item created using the dialogue box mentioned in figure 9 contains its item name, description, item value, and image related to item. The figure also has background view containing list of items that were created. These items are categorized as company's normal items.

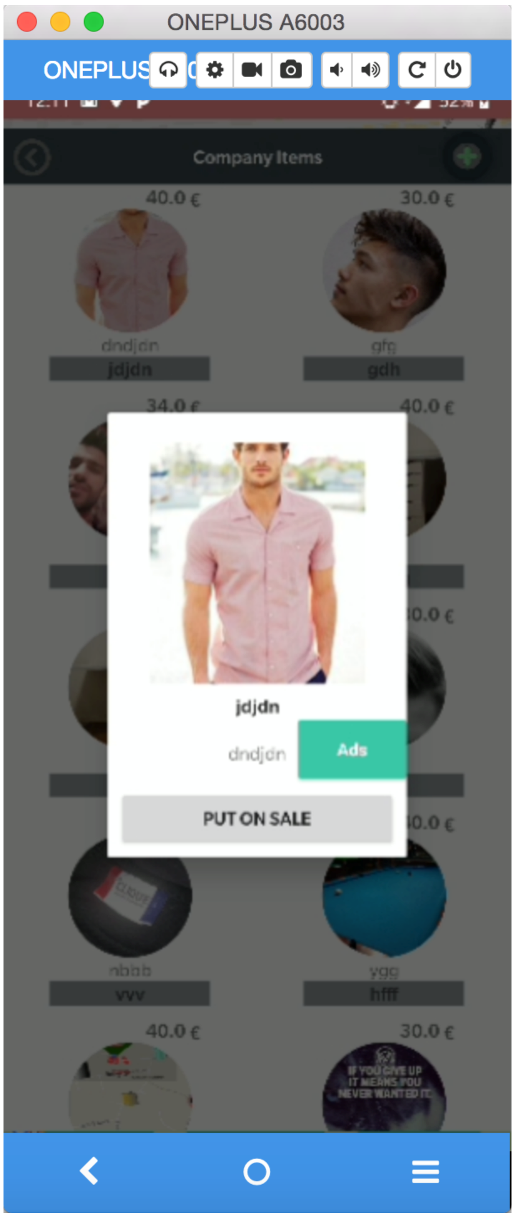


Figure 10. Item view in application.

In the figure above the details of the item have been clearly specified along with its image. Also, other items of company can also be seen in the background the user can put the item on sale by offering discount to consumers from same dialogue box shown above.

In addition to creating items and making list of items in store as stock, companies can put these items on sale and offer discounts to their customers. Company must enter the discount amount and deadline for the offer. Figure below shows the layout where user can enter the values and put the item that has been selected on sale.

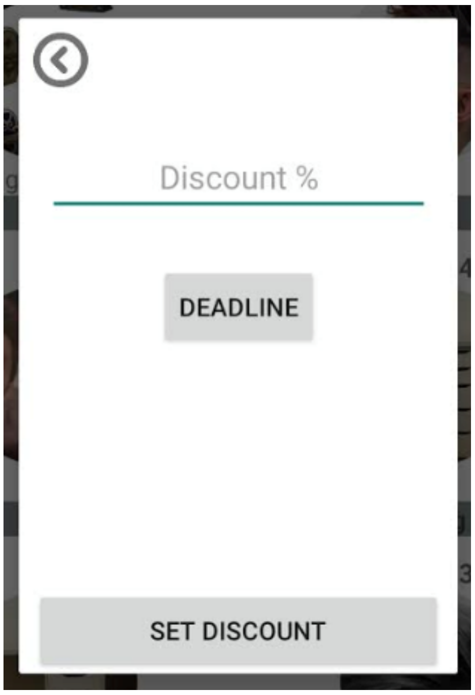


Figure 11. Item discount creation view.

The application stores the items in two different list view, one with normal items, and next view for discount items. The figure shown below shows the layout for list of discount items. The items contain their details and also the discount price for the item.

6.1.4 Managing Staffs

Apart from managing items in the application, business users can add employees to provide them access to various employee associated functionalities. Employees can be added and assigned numerous roles such as store manager and delivery person to provide access control. The business users can add their staffs by searching through their e-mail address or by scanning the user's unique QR code provided during the sign-up process by Ours'.

The figure below shows the layout of the page in application where an employee can be added to a group by an administrator.

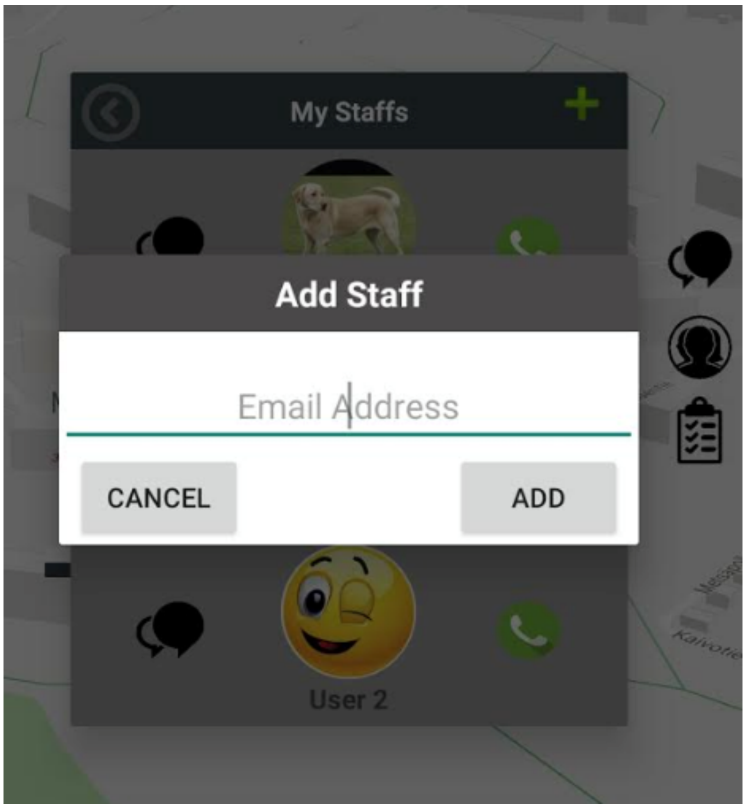


Figure 12. Adding staff member view of Ours' application.

The layout to add staff members to a business organization group is done through a clear dialogue box where the accounts are searched by email address. As soon as the business user adds a staff, the notification is sent to the staff member. Background view consists of list view of staff members. Business user can also call their staffs and send message to them.

6.1.5 Making Schedule

Ours application provides business customers the ability to create working schedule for their employees. To create the work list, the information such as per working date, starting time, ending time and the description of the work are required. Business users can also directly assign work to the worker. Below figure shows the layout where business users are requested to fill the work information to create a work list.

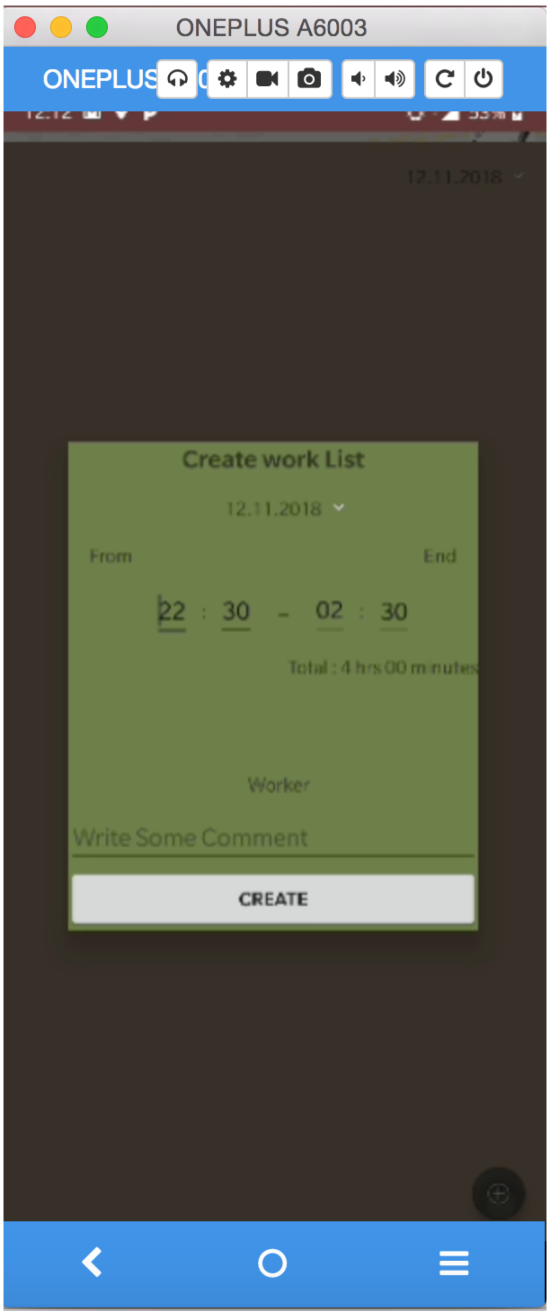


Figure 13. Creating work list view.

In the above figure a work schedule for an employer can be created by entering the starting time and ending time and assigning it to the worker by entering their username.

6.1.6 Online sellers and their sales status through graph

In addition to stock management and employee management, Ours' application also provides analytics ability. Whenever the worker who has been assigned for a work tries to log in, the logging up notification is pushed to the respective business user. Business user are able to accept or decline the the worker log in request. The figure shown below is the view from business user interface where a worker has requested for a log in.

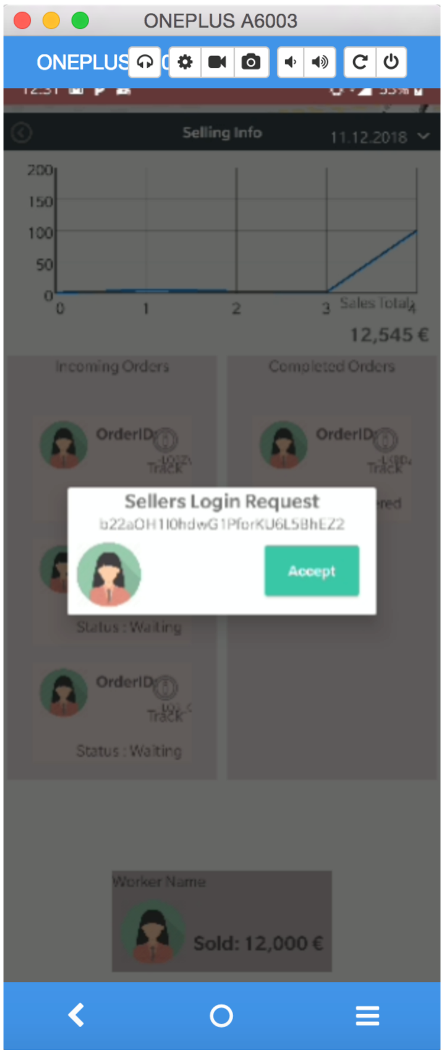


Figure 14. View of login request from an employer.

The background view in the above figure has the view for the workers list and sales report of the business. The view contains the list of workers, list of incoming orders, and list of completed orders. The graph on the top shows the report of the sales for that day.

Business users are also able to keep track on the worker's work flow. The figure shown below is an example of the status which are shown to the business user.

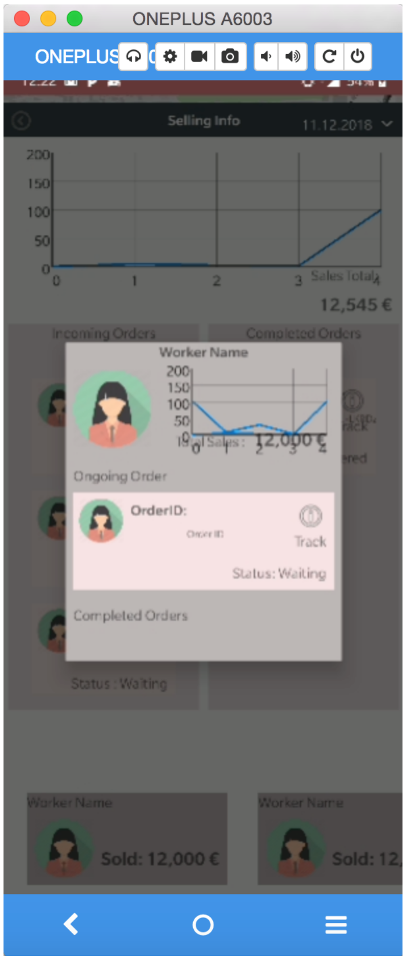


Figure 15. View of employee activities analytics.

In the figure shown above employers can view seller's sales graph, their ongoing order, and completed orders. The graph shows the sales report of the worker for working hours that have been tracked using analytics of Ours' application.

6.1.7 Distributing Coupons in Maps

Apart from creating items, managing staffs, keeping the records on incoming and outgoing orders, business users can create coupons with advertisements and discounts for customers to attract them towards the company. To build stronger relationship with their existing customers or to attract new ones, the business users' interface is equipped with google maps that allows them to target certain geographical locations for advertisement. The maps API provided by Google has multiple features which have been utilized for location tracking and advertisement. The figure below shows the view of ads and coupons in google map inside Ours' application.

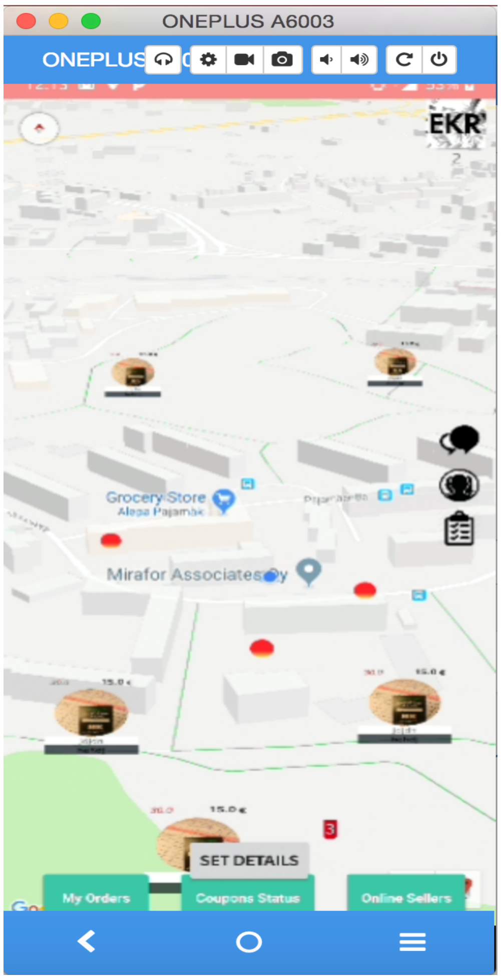


Figure 16. Advertisements view in Ours' application.

In the above image the business users can list the advertisements by touching on google map view of application. The event handler tracks the touch on the map area and displays the view where users can enter the data to be advertised in the location they have selected in google map. The code that handles this event is listed in appendix 2 under event listener title.

Subsequently on clicking set details button, a dialog box like the figure below is shown to the screen.

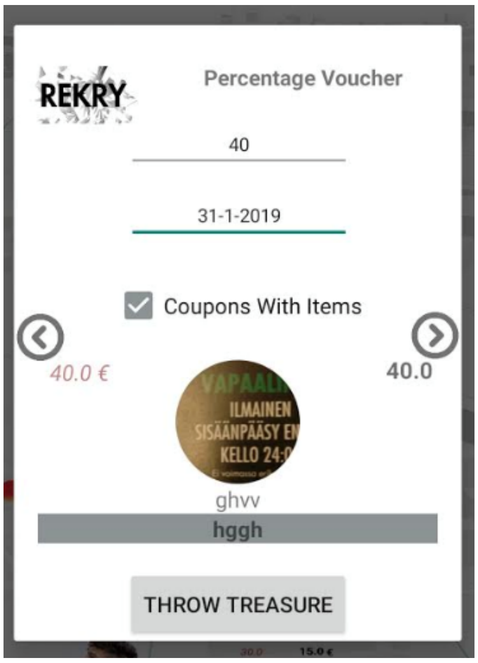


Figure 17. Coupons detail view box.

In the above image business users are requested to enter details corresponding to discount percentage and deadline for the coupon and select option on either the coupon can be applied to single item or is free to use in any product. Coupon consisting of items are displayed to map in order give customers an option for choosing any of the item in order to use the coupon. This helps business user to know the items which their customers are taken with. On other hand, coupons with no items are the type of coupons which can be used for any items while purchasing from the company. After the coupons are created, they can be placed in chosen locations as treasure, the coupons are inflated over map as markers. These markers are inflated to the view based on the type of the coupon. When the markers are clicked by users who are collecting coupons they inflate to show the coupon and reveal the detail of the coupon.

6.2 Customers User interface

Alike the business users, the end users have their own user interface in Ours' application. This interface allows the end users to access the features of the application as well as to their coupons and friends circle which are explained on their own sub-heading below.

6.2.1 Sign Up and Log in View

In order to use all features, customers are requested to sign up by filling registration form in Our's application. The data collected from users are their name, age, e-mail address and password. The figure below illustrates the Sign-up view for customers.

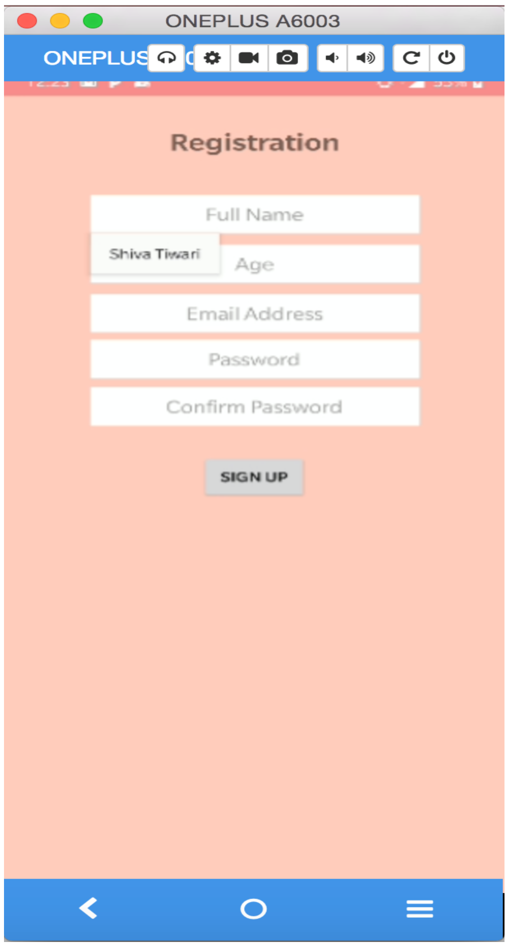


Figure 18. User registration view box.

In the above figure users are requested to input information on each field as per input placeholders. After all the required entries in the sign-up form are filled, the users can click the sign-up button to access the end-user UI.

In the event the user has their account on Ours' application they can sign in effortlessly by logging in with their username and password. The figure below shows the layout for user to log in.

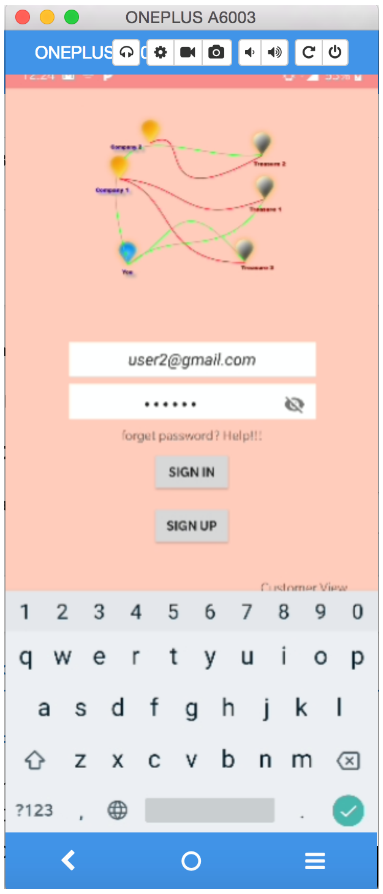


Figure 19. Login page for users .

In the above figure login view of Our's application for customers is shown. After the user has registered their account in sign up process they are provided with username and password which is required to log in and run the application. After successful log in the user is directed towards their home map view. Also, if a user forgets their username or password there is also an option to get a new password through this view. The user has to click on the forgot password button and will be redirected towards another page where after entering the data related t their username and password , their account will be reset.

6.2.2 Home View

Users on successful sign up or log in, are directed to the users' home screen. The home screen consists of a map fragment on main activity, along with image buttons and normal buttons. The buttons call different functions in the applications. The map on the home screen contains various markers representing various coupons distribute by different businesses in the users' geographical vicinity.

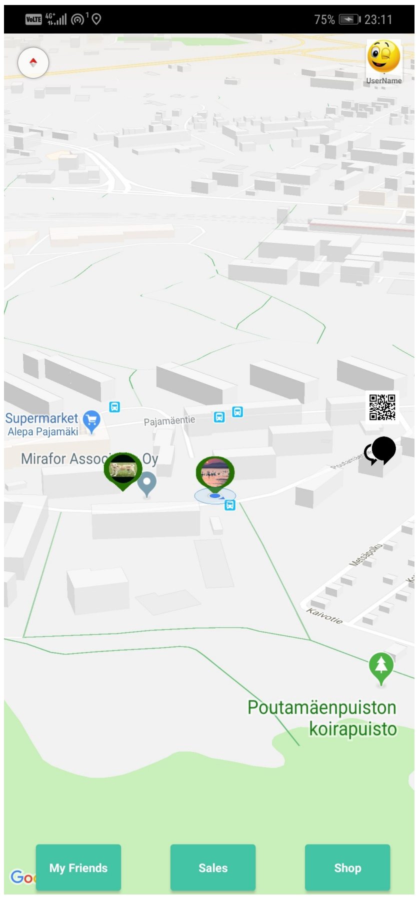


Figure 20. User home view in application

In the above figure home view of a user after successful login is shown. Google map is integrated in the home view of user as well which makes user experience effective for searching restaurants and other businesses. From the map view users can select restaurants and search foods that they are offering and place order from the same view. The functionalities of all the buttons and images in the home screen of user view are explained in tables.

The image buttons inflated on home screen are listed in table below.

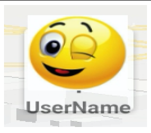
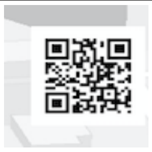
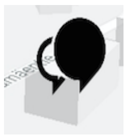
Image	**Description**	**Task**
	User Image Button	On Click () shows/hides Menu buttons.
	QR Scanner Button	On Click () opens camera for scanning QR code.
	Message Button	On Click () displays Message Dialog Fragment.

Table 1. Description of activities done by images on user home screen.

As described in the table above all the images acting as buttons in the home screen of user have different task that they performed when clicked. The code used to represent the functionalities of these images when clicked are listed in appendix 3 under home view images heading.

Normal Buttons with Firebase UI styles inflated on home screen are listed in table below with their respective buttons and functions.

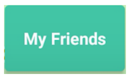
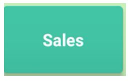
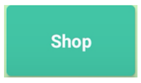
Buttons	Description	Function
	My Friends Button	On Click () displays My Friends Dialog Fragment.
	Sales button	On Click () displays Sales Items Dialog Fragment
	Shop Button	On Click () displays List of Company

Table 2. Buttons on home view and task description

In the above table buttons shown in home view of user are listed and the tasks that are done when a user clicks on the buttons are labelled .Also, when a user clicks on the own image button a list of buttons are inflated and shown to the user. Those buttons and the tasks they performed are explained in list below.

- My coupons button when clicked displays coupons caught and stored by user.
- My orders button when clicked displays orders made by user.
- Work button when clicked displays work log of user in dialogue box.
- Setting button when clicked displays user setting of Ours' application.
- Sign out button when clicked logs out the user from Ours' application.

6.2.3 Follow Business / buy products from Business

Users can search for registered business in the application and follow them. Business sectors are categorized into Restaurants, Cafe, Night clubs, Electronics stores, Salons, and Grocery Stores as shown in the figure below.

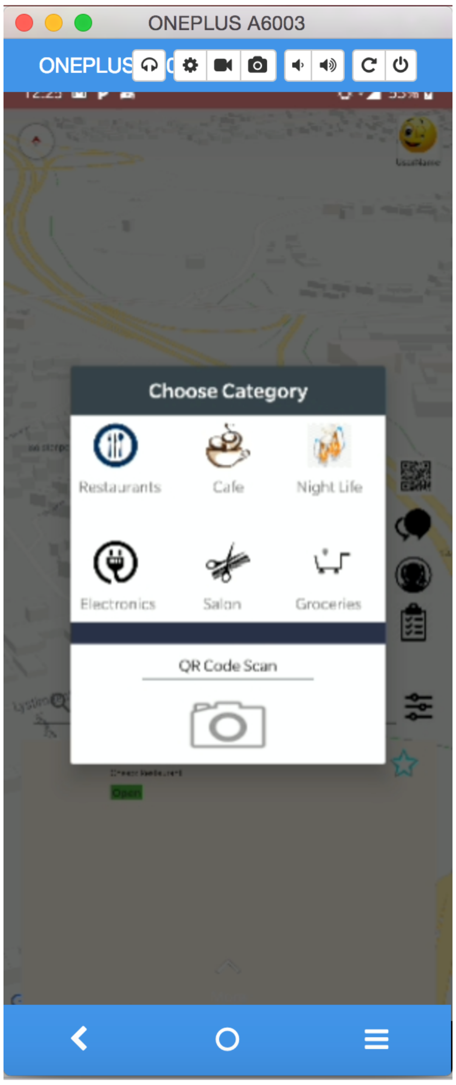


Figure 21. Business category view in application.

In the above image business category is shown which opens the list of restaurants when clicked on restaurant and other business when clicked on respective category. It comprises of Image buttons, Text views and Edit text. User can click on any of six buttons and search for nearby business. However, on clicking image button with camera image background, camera opens for scanning the QR code of the business sector. On successful scanning of the code, the consumers are directed towards the business home view where list of selling item is inflated. The offers' notifications are sent to the users who are followers of the business. User following business can see sales products in their sales news feed. Apart from providing information about the companies, Ours' application mainly focuses on making the process of buying and selling easier for both businesses and customers and helping the businesses maintain stronger relationships with their customers. As every business user in this application are provided with their unique QR code, consumer on other side can easily scan QR code of the business and view the products they are selling. The figure below is the view of a restaurant when a user clicks the restaurant name after searching it.´

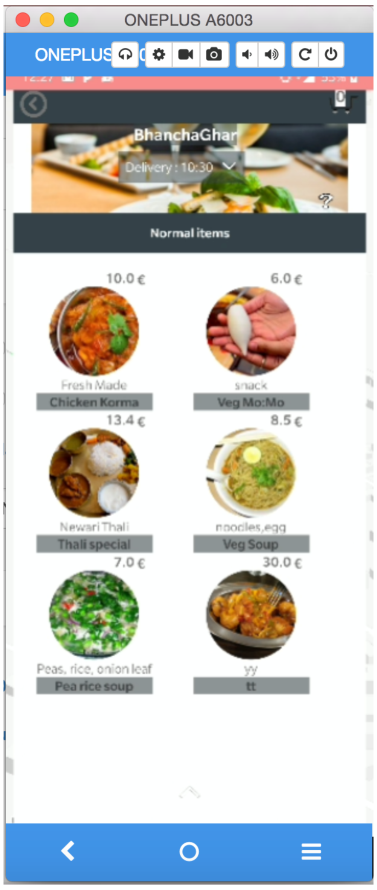


Figure 22. View of a company's home page

The figure above shows an example of the business home view screen. It has business user profile image, name, option to choose between delivery and pick up, preview of items in list view. The items are listed in grid view using customized adapter which extends Array Adapter. The view inflated on the grid view list item consists of Image View and Text Views.

When a user views the home page of a business and finds some items to purchase they can get the detail of the item before purchasing it by clicking the image in home view. The figure below shows the item view inflated over the dialog box.

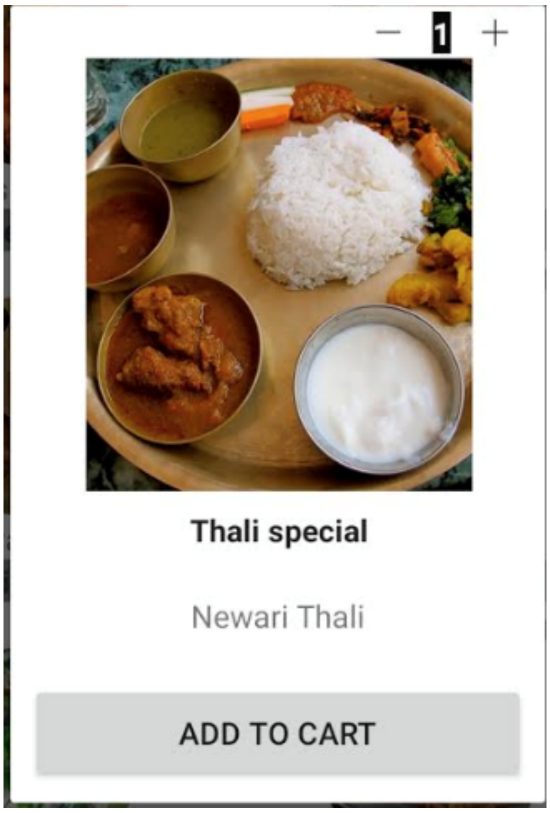


Figure 23. Item description image.

In the above image description of a single item is shown. It contains image view of the the item, item name, item's description, stock number, and a button to add the item to the cart. If a consumer adds any item to cart, they can view their cart list on clicking the image button showing cart image as background on business home view screen. Cart view contains list of items added to the cart by the end user, receipt for the cart transection, and Pay button to confirm the order. When a user makes an order, they are given a QR code of the order. QR code image helps a seller from that company to scan the code and complete the order transaction. For example, when a consumer makes an order online, consumer can track info about the particular order. Consumer are notified about order status, each item conditions, item order status and live tracking location of the order. Whenever a delivery person delivers the order package, QR code of the order is scanned and the order is then completed. Google maps integration in the application helps to keep track on location of order package and inflate location over map in real time. This helps consumer to know real time information about their orders and the time till which the item they ordered is ready and delivered.

6.2.4 Add friends and share coupons with friends

Users as consumers in this application are equipped with the services like adding their friends. This application helps consumers to keep friends together and share their coupons to their friends as gift voucher. The figure below shows the view where a user can add a friend in their social circle in Ours' application

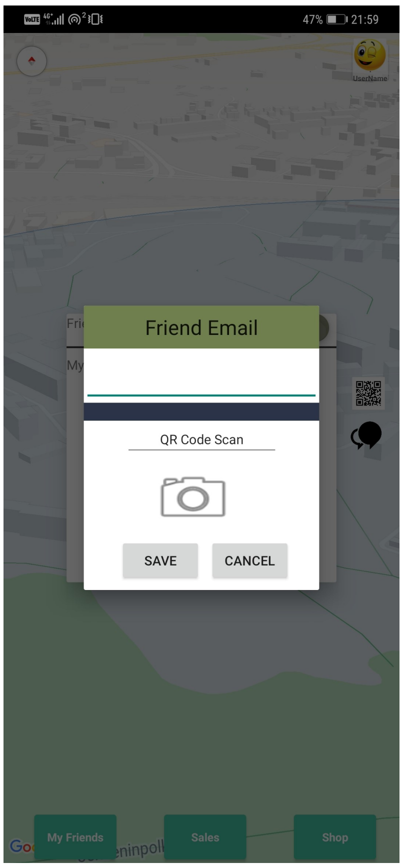


Figure 24. Add friend view .

In the above image a user can add friend by adding their e-mail address in the text field and pressing save button. Also, adding friend can be done by scanning the QR code if another user by clicking QR code scan button . After save button is pressed the user whose email address is entered or QR code is scanned is notified about the request and can either accept or reject the friend request. The decision is also then notified to the end user who made the request.

6.2.5 Get hired as sellers

End users, apart from buying products from business, adding friends and sharing coupons with them, can also work as sales person, cashier and delivery person. Whenever a business company adds a user as their staff, user can see list of work places they are added as staff member as shown in image below.

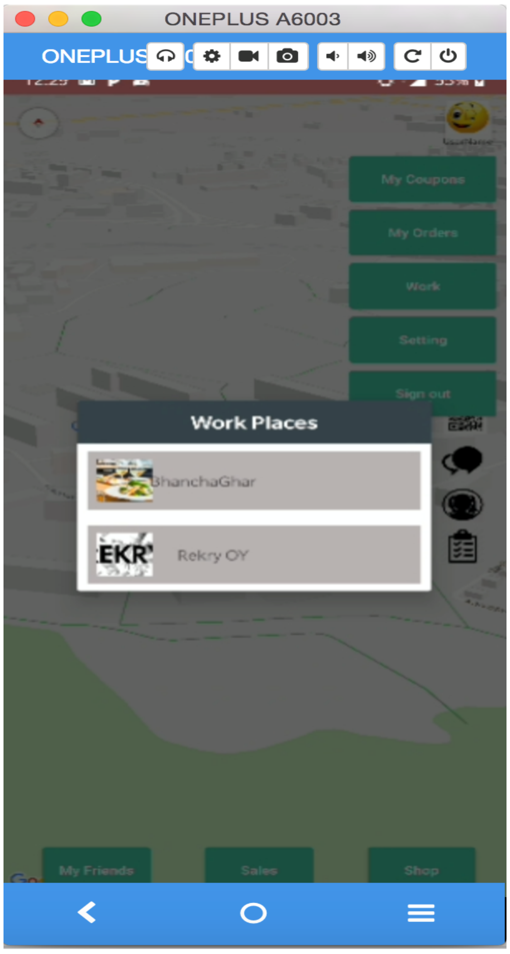


Figure 25. View of added workplaces of a user.

In the above figure work places in which the user has been assigned as a worker is listed. A user can only be listed as an employee by the business owner of respective association. If a user wants to start working for a company that they have been listed as a worker, they must select the company that they intend to work from the view shown above. After the selection has been done the request is sent to the business user and they must give the permission to start work for the respective user .

After the user is given permission from the employer to log in and start working they are directed towards the view which is shown in figure below.

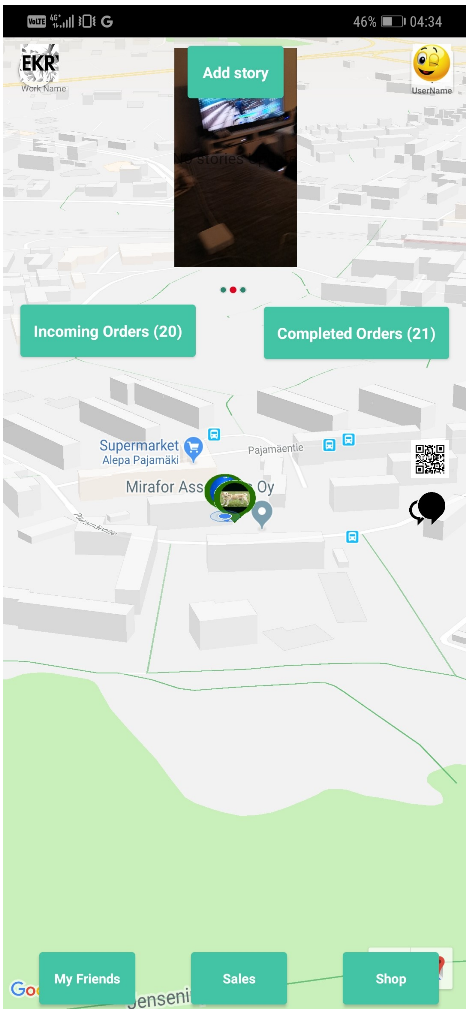


Figure 26. Work view of a user working as a seller.

The figure above is an example of the view shown after successful work log in. The view contains image button, normal buttons, view pager with video view and image view. Sellers can view list of incoming, completed orders, and add stories for business advertisement provided they are granted access by the business user to do so. Modern mobile phones are in the hands of everyone so users working as a salesperson are given freedom to use their mobile devices for advertising and selling the products. The main purpose of this feature is to use mobile phones over cash register and terminal because mobile phones can be installed with ours' application which helps from inventory, selling the products, keeping sales records and secure mobile payments. Business users can easily view each worker sales status in real time and keep records for long run.

6.2.6 Map View with Coupons

Ours' application is inflated with google map fragment on main activity. The coupons distributed by any business users are shown to users as map markers. The figure below shows an example of the layout during a test period.

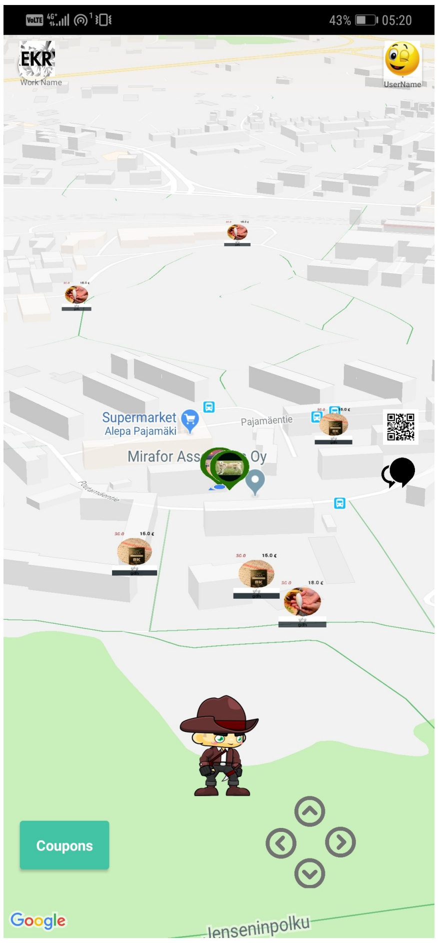


Figure 27. Map view with coupons shown.

In the above figure of map with coupons listed in them, when a user clicks and holds on the map view the function to inflate the resulting view is called. The view contains information about live treasures info and nearby business coupons. The coupons which are within the range of 300 meters are shown in the google map. In order to search for more coupons around, users can use gyroscope sensor of the phone to move user search marker around the map view. The user marker which is used in this case is temporary. This allows users to navigate between different locations in map for searching the coupons which are around on the location where the User Coupons search marker is located.

When a user clicks on a coupon the view is inflated showing the details added by the advertiser as shown in the figure below.

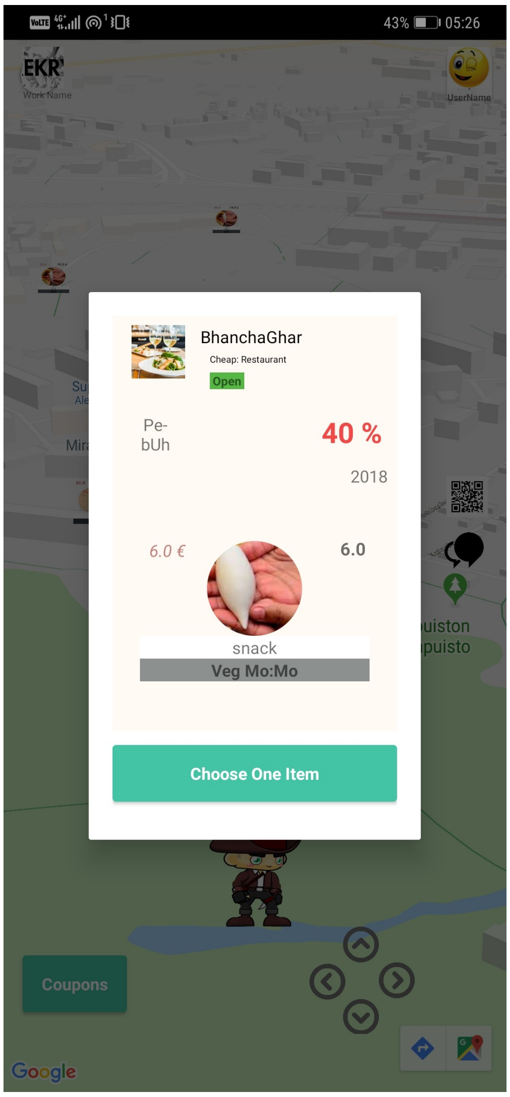


Figure 28. Inflated view of a coupon.

The Coupon Marker click view shown in figure above is an example of a coupon thrown on Map by Bhanchaghar restaurant. The coupon has the details like coupon Id, coupon discount percentage, deadline for the offers, and list of items that are available as offer for that coupon. Users, who can find these coupons can easily save coupon to their own database. In order to save to coupon, user should choose one of the items from the list and save the coupon. Whenever a coupon is found by user, the coupon marker is removed from google map view and the business user owning the coupon is notified about the update. There are many types of coupons such as, coupon with items, coupon with no items and coupons for making extra cash. The templates for creating these types of coupons are given to the business users, so that they can choose which type of template fits their requirements. If a user is willing to cancel the coupon searching mode to normal home screen view, user can press on google map and return to home screen easily. Also, user can view the coupons they have collected from my coupons meu .A view of coupons of a test user is shown in figure below.

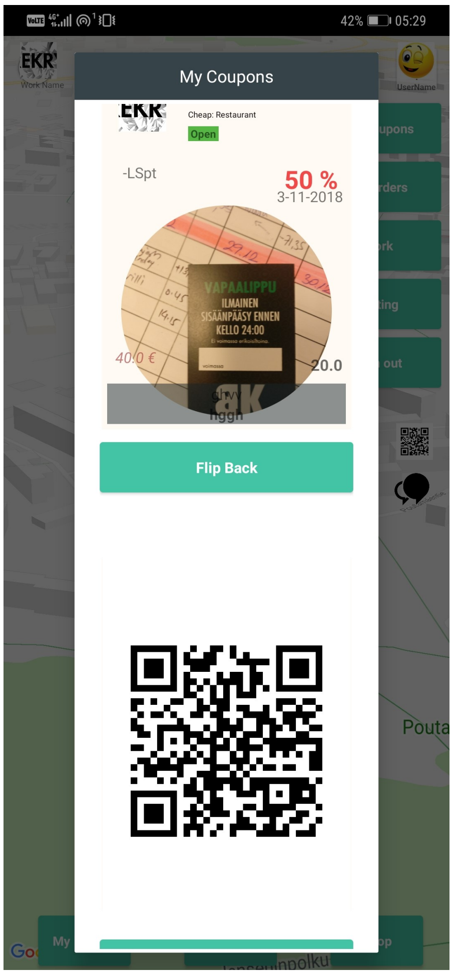


Figure 29. My coupons view of a test user account.

Users as consumer can keep records on their coupons and share coupons to their friends as gift. The figure above shows the list of two coupons which a user has saved to his/her database. First coupon shown on the list with 50% discount and selected item has front view where as second coupon on the list has its back view. Each coupon is integrated with QR code which helps sales person to scan the coupons and do further processing when a user tries to use these coupons. The view shown in figure above which is inflated over dialog box is just an example of coupons list of a test user.

6.2.7 User Profile view

Along with feature like sellers advertising the business by posting their work stories, users can also share their own personal stories to their friends. The sales person stories are visible to the business stories layout where as the user snap stories can be viewed on users own profile view. The figure below shows an example of test user profile, containing QR code of the user, and layout for adding stories.

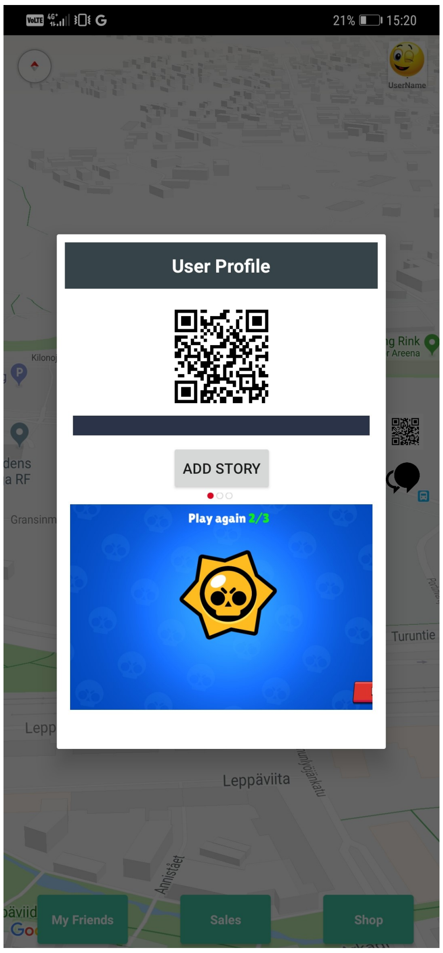


Figure 30. User profile view with user stories.

QR code on the view shown above is placed over Image view where as snap stories are placed over view pager. View pager is integrated with custom adapter which extends pager adapter. Each item view in pager adapter contains either Image View, or Video View according the file that need to be inflated over the view pager view. The view like figure above is inflated over dialog box whenever user clicks on his/ her own marker in google map or press long click on user image button.

6.2.8 Friends profile

Friend’s profile view is inflated either by clicking on friend’s marker on google map or clicking the friend image from contacts list. Both actions call the same function in the application as mentioned in appendix 4. The figure below is an example of friend’s profile which consists of friend’s image, name, daily stories, and their coupons.

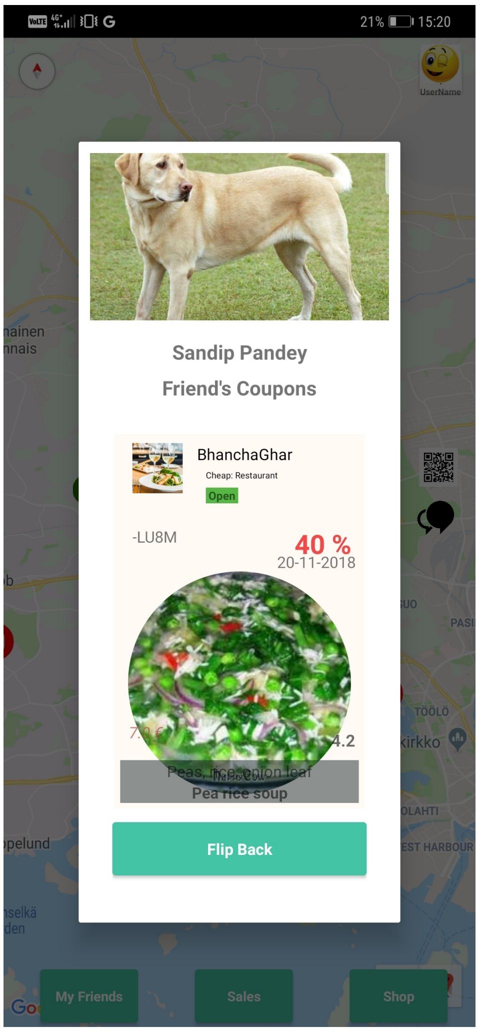
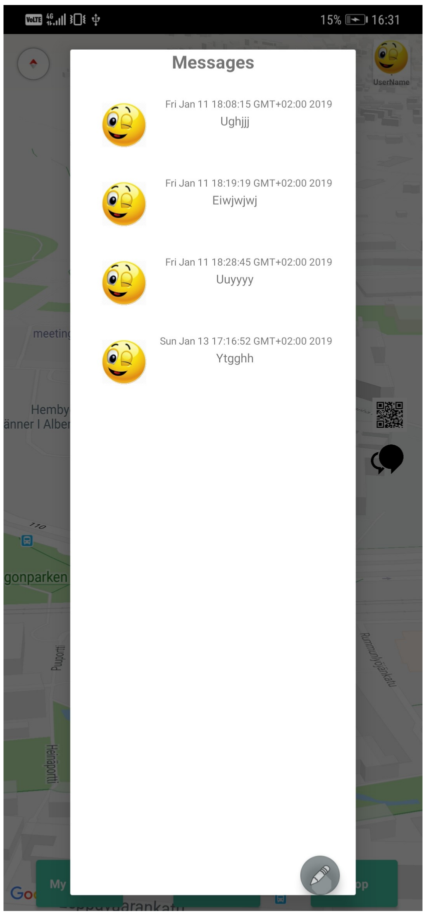


Figure 31. Friends profile view

In the image above, basic profile of a user’s friend is shown. In this view the image shared by the user and their username is shown along with the coupons that they have collected. User can see their friends’ coupons and request those coupons for them if needed. The request is then sent to the user who has the coupons and can accept or reject the request the request has been accepted then the user requesting the coupons gets them and can use it for purchasing the respective item.

6.3 Messaging

Although business users and consumer users have different interfaces and different sets of features, they share same type of view for messaging. Our's application has a messaging service which helps business users connect with their employees and customers. Also, normal users can connect with friends by using messaging service provided by our application. The figure below shows an example of messaging view from Ours' application.



Messages

Fri Jan 11 18:08:15 GMT+02:00 2019
Ughjjj

Fri Jan 11 18:19:19 GMT+02:00 2019
Eiwjwjwj

Fri Jan 11 18:28:45 GMT+02:00 2019
Uuyyyy

Sun Jan 13 17:16:52 GMT+02:00 2019
Ytgghh

Figure 32. View of a message service between test users.

In the above image of message view of OURS' application list of messages of the particular user is inflated on list view. Message object class consists of message id, Message Text, message created date, sender ID, and receiver ID which is listed in appendix 5. When a user sends a message hey are stored in the database and sent to the user in the other end at the same time. Also, notification of incoming message is sent to the receiving user .Messaging service helps customers , businesses and friends have better communication and utilize OURS' application as a social media application also.

7 Application Testing

To test the usability of the application and improve our application on the feedback we get during our testing trial we ran a trial session of our application with Ravintola Bhanchaghar. For our first phase of testing the employees of the restaurant used the business account and few voluntary customers were registered as customers. A small meet and discussion were held in the same restaurant so that users can be familiar with the application and get to know about the functionalities of the application. Android devices was used for testing purposes in which Ours' application was installed.

I began the meeting by explaining the reason for the development of our application and further development plans we had for the application which needed their feedback for development. I gave a tour of our application and all the functions that can be used in this application. Since, the design layouts for the user and business views did not have best user experience design that was the first feedback we got during our trial. All the users were requested to use the application for about two days and give us feedback on what could be improved to provide better service with our application. After the trial period was over we got multiple feedbacks that helped improve our application. During the order selection process by a customer the application crashed, and the customers were not able to make orders was the feedback that we got from multiple test users. This helped us find the problem in our database connection and debug it so that application would run smoothly, and users could make orders normally. Another major issue with our application which we learned from our trial testing phase was that the application had user interfaces which could be upgraded and made more user friendly. This trial testing was done mostly to get the feedback and upgrade, debug and develop our application before the deployment phase.

8 Discussion

This project is carried out as the first version of Ours' application. This application is not fully developed to its potential and the author believes the application can make huge impact on building strong relationship between businesses and consumers on further development. Finding coupons, sharing collected coupons to friends, handling the orders and business holders being able to keep track of their inventory are the main features that are included in the application.

This application helps to update cash registers and terminals with mobile phone payment, and also work as advertising platform for business users for selling their products. If the application is designed with better user interface, this might be game changing advertising platform for many business sectors. Apart from these future planning's, the application also keeps data about users' locations, their orders, their interest of coupons types and their usage and best likeable items of any business. The sellers are provided with the feature where each seller can choose list of items which they are willing to advertise and sell more efficiently. These new ideas might help any business users for running profitable business.

9 Conclusion

The aim of this thesis is to build an e-commerce android application for business users and consumer users. These goals have been successfully achieved on completion of this project. The author can put all his findings to build an android application using Java programming language, real time firebase database, android SDK and Google Maps APIs. The application runs smoothly on all android phones available on the market.

This thesis provides android developers basic information on how to build an android application. Developers are also able to get more info about the integration of Firebase real time database and Google Maps APIs in the application.

References

1. Green D. Smartphones brought huge changes to shopping in 2017 [Internet]. Nordic.businessinsider.com. 2019 [cited 17 February 2019]. Available from: https://nordic.businessinsider.com/mobile-shopping-exploded-this-year-2017-12?r=US&IR=T

2. Importance of POS Systems in Bars and Restaurants - ARC [Internet]. ARC. 2019 [cited 20 February 2019]. Available from: http://www.arcireland.com/importance-of-pos-systems-in-bars-and-restaurants/

3.Hellman E. Android programming. 1st ed. Chichester, West Sussex: Wiley; 2014.

4. A Guide to the Android Studio Designer Tool - Techotopia [Internet]. Techotopia.com. 2019 [cited 7 January 2019]. Available from: https://www.techotopia.com/index.php/A_Guide_to_the_Android_Studio_Designer_Tool

5. McGrath M. XML. 1st ed. Southam: Computer Step; 2007.

6. Schildt H. Java: the Complete Reference, Eleventh Edition. 11th ed. New York: McGraw-Hill Education; 2018.

7. Moroney L. The definitive guide to Firebase. 1st ed. New York: Apress; 2017.

8. Installation & Setup on Android | Firebase Realtime Database | Firebase [Internet]. Firebase. 2019 [cited 17 January 2019]. Available from: https://firebase.google.com/docs/database/android/start

9. Geo-location APIs | Google Maps Platform | Google Cloud [Internet]. Google Cloud. 2019 [cited 17 March 2019]. Available from: https://cloud.google.com/maps-platform/?apis=maps

10. Android Activity Life Cycle [Internet]. Tutorialspoint.com. 2019 [cited 24 January 2019]. Available from: https://www.tutorialspoint.com/xamarin/images/android_activity_lifecycle.jpg

11. Components of an Android Application - GeeksforGeeks [Internet]. GeeksforGeeks. 2019 [cited 24 January 2019]. Available from: https://www.geeksforgeeks.org/components-android-application/

12. Android Application Components [Internet]. www.tutorialspoint.com. 2019 [cited 5 February 2019]. Available from: https://www.tutorialspoint.com/android/android_application_components.htm

Appendix 1. Services, Broadcast Receivers and Content Providers

Services

```
public class ExampleService extends Service {

}
```

Broadcast Receivers

```
public class MyReceiver extends BroadcastReceiver {

   public void onReceive (context, intent) {}

}
```

Content Providers

```
public class MyContentProvider extends  ContentProvider {

   public void onCreate(){}

}
```

Appendix 2. Signup, Login and Event Listeners

```
public class LogInActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    private EditText userEmail;
    private EditText password;

    private FirebaseDatabase database = FirebaseDatabase.getInstance();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loginactivity);

        mAuth = FirebaseAuth.getInstance();

        userEmail=findViewById(R.id.emailAddress);
        password=findViewById(R.id.password);

        TextView signuphere=findViewById(R.id.signupTextview);
        Button signIn = findViewById(R.id.signIn);
        signuphere.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                opensignupdailogbox();
            }
        });

        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startSignin();
            }
        });
    }

    private void startSignin() {
        String email=userEmail.getText().toString();
        String passwordvalue=password.getText().toString();

        if (TextUtils.isEmpty(email)||TextUtils.isEmpty(passwordvalue)) {
            Toast.makeText(LogInActivity.this, "Fields is Empty",
                    Toast.LENGTH_SHORT).show();
        }
        else{
            mAuth.signInWithEmailAndPassword(email, passwordvalue)
                    .addOnCompleteListener(this, new OnCompleteLis-
tener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task)
{
                            if (task.isSuccessful()) {
                                // Sign in success, update UI with the signed-
in user's information

                                Toast.makeText(LogInActivity.this, "Authenti-
cation Successful",

                                        Toast.LENGTH_SHORT).show();
                                FirebaseUser user = mAuth.getCurrentUser();
                                updateUI(user);

                            } else {
                                // If sign in fails, display a message to the
user.
```

```
                                Toast.makeText(LogInActivity.this, "Authenti-
cation failed.",
                                        Toast.LENGTH_SHORT).show();
                            }
                        }
                    });

        }



    }

    private void updateUI(final FirebaseUser user) {

        // check here if the user is seller or consumer
        if (user != null) {
            // User is signed in


            if(user.getEmail().contains("@gmail.com")){
                Toast.makeText(LogInActivity.this,"Customer:
"+user.getEmail(),
                        Toast.LENGTH_SHORT).show();

                Intent intent=new Intent(LogInActivity.this, ConsumerMapActiv-
ity.class);
                startActivity(intent);
                finish();

            }
            if(user.getEmail().contains("@company.com")){
                // User is signed out
                Toast.makeText(LogInActivity.this,"Seller: "+user.getEmail(),
                        Toast.LENGTH_SHORT).show();

                Intent intent=new Intent(LogInActivity.this, SellerMapActiv-
ity.class);
                startActivity(intent);
                finish();
            }

        }
        else {
            Toast.makeText(LogInActivity.this,"No user",
                    Toast.LENGTH_SHORT).show();

        }



    }


    private void opensignupdailogbox() {
        FragmentTransaction ft = getSupportFragmentManager().beginTransac-
tion();
        ft.addToBackStack("Sign Up Fragment");

        DialogFragment fragobj = new SignUpFragment();

        fragobj.show(ft, "Sign Up Fragment");

    }
    @Override
    public void onStart() {
        super.onStart();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser!=null){
            updateUI(currentUser);
        }

    }
    @Override
    public void onStop() {
        super.onStop();

    }


}
```

Appendix 3. Sellers Interface Listeners

Sellers My Items Click Listener:

```
myItems.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        android.support.v4.app.FragmentTransaction ft = getSupportFragmentMan-
ager().beginTransaction();
        ft.addToBackStack("Seller Normal Items");
        DialogFragment fragobj = new Company_Normal_Items();
        fragobj.show(ft, "Seller Normal Items");


        android.support.v4.app.FragmentTransaction ft1 = getSupportFragment-
Manager().beginTransaction();
        ft.addToBackStack("Seller sales Items");
        DialogFragment fragobj1 = new Company_Discount_Items();
        fragobj1.show(ft1, "Seller sales Items");

    }
});
```

Appendix 4. Consumers Contacts View

```
contactsbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        android.support.v4.app.FragmentTransaction ft = getSupportFragmentMan-
ager().beginTransaction();
        ft.addToBackStack("My Contacts");

        DialogFragment fragobj = new Fragment_UserContacts();

        fragobj.show(ft, "My Contacts");

    }
});
```

Appendix 5. Chat Box User Interface

```
public class MessagesFragment extends DialogFragment {
    MessageListAdapter messageGridAdaopter;
    EmployeeChooseAdaptor friendslistViewAdapter;

    ArrayList<Friend> friendlist = new ArrayList<>();

    String UserID = FirebaseAuth.getInstance().getCurrentUser().getUid();
    HashMap<String,Message> messageslisthashmap=new HashMap<>();
    ArrayList<Message> messageslist=new ArrayList<>();
    DatabaseReference mFirebaseDatabaseReference = FirebaseDatabase.get-
Instance().getReference();
    DatabaseReference dd;

    public MessagesFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(@NonNull final LayoutInflater inflater,@NonNull
final ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_messages, container,
false);
        ListView messagelistview=view.findViewById(R.id.messagelistview);
        FloatingActionButton compose = view.findViewById(R.id.compose);

        friendslistViewAdapter = new EmployeeChooseAdaptor(getContext(),
R.layout.select_employee_eachview, getfriendlist());
        messageGridAdaopter = new MessageListAdapter(getContext(), R.lay-
out.each_message_view, getMessageslist());
        compose.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                View compose = inflater.inflate(R.layout.compose_message, con-
tainer,true);
                final Dialog d = new Dialog(getContext());
                final EditText messageeditor=compose.findViewById(R.id.mes-
sageeditor);
                ListView friendlistView = compose.findViewById(R.id.friend-
listView);
                friendlistView.setAdapter(friendslistViewAdapter);
                friendlistView.setOnItemClickListener(new AdapterView.OnItem-
ClickListener() {
                    @Override
                    public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
                        Toast.makeText(getContext(), "Sent to "+friend-
list.get(position).getFriendName(),
                                Toast.LENGTH_SHORT).show();
                        long currentDateTime = System.currentTimeMillis();
                        dd= mFirebaseDatabaseReference.child("Mes-
sages").push();
                        String messageId=dd.getKey();
                        Message message=new Message(messageedi-
tor.getText().toString(),messageId,currentDateTime,friendlist.get(posi-
tion).getFriendID(),UserID);

                       dd.setValue(message);
```

```
                        d.dismiss();
                    }
                });
                d.setContentView(compose);
                d.show();
            }
        });

        messagelistview.setAdapter(messageGridAdaopter);
        messagelistview.setOnItemClickListener(new AdapterView.OnItemClick-
Listener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int po-
sition, long id) {

               android.support.v4.app.FragmentTransaction ft = getFragmentMan-
ager().beginTransaction();
                ft.addToBackStack("eachMessageviewfragment");
                Friend friend= friendsMessagelist.get(position);
                Bundle bundle=new Bundle();
                bundle.putSerializable("Friend",friend);
                DialogFragment fragobj = new Eachmessage_Fragment();
                fragobj.setArguments(bundle);
                fragobj.show(ft, "eachMessageviewfragment");
            }
        });


        //gridAdapter = new MessageListAdapter(this, R.layout.company_treas-
ure_items_list_view,companyItems);

        return view;
    }



    private ArrayList<Friend> getfriendlist() {
        friendlist.clear();
        HashMap<String, Friend> friendlisthashmap = ((ConsumerMapActivity)
getActivity()).getFriendlist1();
        for (HashMap.Entry<String, Friend> entry : friendlisthashmap.en-
trySet()) {
            String key = entry.getKey();
            Friend value = entry.getValue();
            friendlist.add(value);


        }
        return friendlist;

    }

private ArrayList<String> friendIDs=new ArrayList<>();
    private ArrayList<Friend> friendsMessagelist=new ArrayList<>();

    private HashMap<Friend,String> recentmessageHashMap=new HashMap<>();

    private ArrayList<Friend> getMessageslist(){
        mFirebaseDatabaseReference.child("Messages").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()){
                    friendIDs.clear();
```

```
                        friendsMessagelist.clear();

                            for(DataSnapshot data:dataSnapshot.getChildren()){
                                String message1=data.child("message").get-
Value(String.class);
                                String messageid=data.child("messageId").get-
Value(String.class);
                                long date=data.child("date").getValue(Long.class);
                                String recieverID=data.child("recieverID").get-
Value(String.class);
                                String senderID=data.child("senderID").get-
Value(String.class);

                                if(recieverID.equals(UserID)){
                                    if(friendIDs.contains(senderID)){

                                    }else{
                                        friendIDs.add(senderID);
                                    }

                                }
                                if(senderID.equals(UserID)){
                                    if(friendIDs.contains(recieverID)){

                                    }else{
                                        friendIDs.add(recieverID);
                                    }

                                }

                            }

                            for(Friend fr:friendlist){
                                for(String friendID:friendIDs){
                                    if(fr.getFriendID().equals(friendID)){
                                        friendsMessagelist.add(fr);
                                        messageGridAdaopter.notifyDataSetCh-
anged();
                                    }

                                }
                            }


                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {

                }
            });

            return friendsMessagelist;
        }

}
```