Shopping List Management Application

A project report submitted for
Android Apps Development Lab (Semester VII)

by

Smith Dabreo (8382)

Brian Dias (8384)

Under the guidance of

Prof. Nilesh Patil

(sign with date)

DEPARTMENT OF INFORMATION TECHNOLOGY
Fr. Conceicao Rodrigues College of Engineering
Bandra (W), Mumbai - 400050
University of Mumbai

Approval Sheet

Project Report Approval

This project report entitled by **Shopping List Management Application** by **Smith Dabreo and Brian Dias** is approved as project for **Android Apps Development Lab** in Fourth year Engineering (Sem - VII), Information Technology.

Examiners

1._____
2._____

Date:

Place:

Introduction

Shopping is an important part of our life. Most families tend to buy something or other everyday. According to a study an average person goes to the grocery store for shopping 1.6 times a week and spends 43 minutes there, not including the time spent getting to and from the store. Therefore, on average we spend about 60 hours grocery shopping per year. Additionally, most of us tend to forget all the things we want to buy. There are some people who take sheets of paper with them while shopping or even some people make a note on their mobile phone. However, these methods are inefficient and waste time. A dedicated shopping list app can make a daily shoppers life extremely easy. He/she can make a list well in advance to his/her liking even edit the list whenever they want to add and delete products. Furthermore people with these app can share the list with their family and friends for ease of use. We tend to build an application which can do all this and much more which can be used in our daily lives and make our lives easier and save time.

List of Figures

Table of Content

Chapter 1
Introduction

Introduction to Android:

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc. It contains a Linux-based Operating System, middleware and key mobile applications. It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. It provides support for messaging services (SMS and MMS), web browser, storage (SQLite, Firebase), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

Features of Android

- It is open-source.
- Anyone can customize the Android Platform.
- There are a lot of mobile applications that can be chosen by the consumer.
- It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

Chapter 2:
Widgets

Widgets are an essential aspect of home screen customization. One can imagine them as "at- a glance" views of an app's most important data and functionality that is accessible right from the user's home screen.

The widely used android widgets are given below:

- Android Button: A button performs event handling on button click.
- Android TextView: A **TextView** displays text to the user and optionally allows them to edit it.

Android Button

Android Button represents a push-button. A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it. We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

XML Code:

```
<Button
   android:id="@+id/Button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentBottom="true"
   android:layout_centerInParent="true"
   android:background="#006bff"
   android:text="@string/button"
   android:textColor="#FFFFFF" />
```

Java Code:

```
Button button;
button =(Button)findViewById(R.id.Button);
button.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      openMainAct();
   }
});
```

Result:

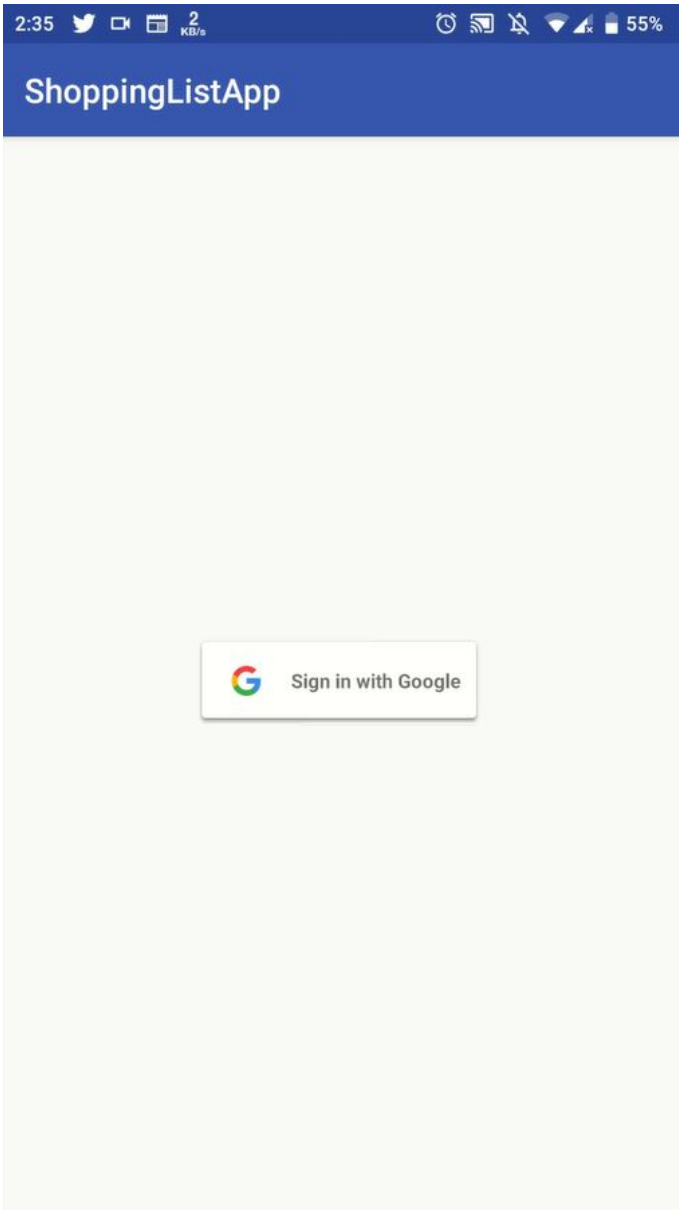

Figure 2.1

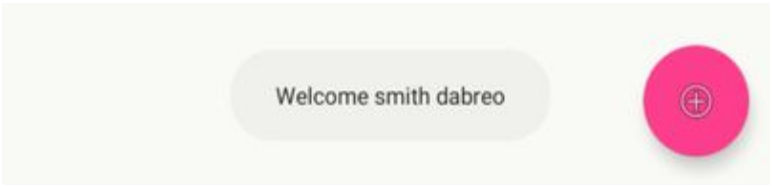

Figure 2.2

Chapter 3
Layouts

A Layout dictates the alignment of widgets (such as Text, Buttons, EditText box) as we see in the Android Application. All the visual structure we see in an android app is designed in a Layout. Every Layout is defined in an xml file which is located in App > res > Layout in New Android Studio.

The six different layouts are

1. Linear Layout
2. Relative Layout
3. Table Layout
4. Grid View
5. Tab Layout
6. List View
7. Container

Container

A convenience widget that combines common painting, positioning, and sizing widgets.

A container first surrounds the child with padding (inflated by any borders present in the decoration) and then applies additional constraints to the padded extent (incorporating the `width` and `height` as constraints, if either is non-null). The container is then surrounded by additional empty space described from the margin.

During painting, the container first applies the given transform, then paints the decoration to fill the padded extent, then it paints the child, and finally paints the foregroundDecoration, also filling the padded extent.

We have majorly used Relative Layout.

XML Code:

```
<RelativeLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center_horizontal"
   android:layout_marginTop="@android:dimen/notification_large_icon_height">

   <TextView
      android:id="@+id/text2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentStart="true"
```

```
        android:layout_alignParentTop="true"
         android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"
         android:layout_weight="1"
        android:fontFamily="sans-serif-condensed-light"
        android:text="@string/text2"
        android:textAlignment="center"
        android:textColor="#f97d00"
        android:textSize="35dp"
        android:textStyle="bold|italic" />


</RelativeLayout>
```

Frame and Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <LinearLayout
        android:id="@+id/botMsgLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start|center_vertical"
        android:layout_marginTop="4dp"
        android:layout_marginBottom="4dp"
        android:layout_marginEnd="16dp"
        android:layout_marginStart="8dp"
        android:background="@drawable/bot_bg_bubble"
        android:gravity="start|center_vertical"
        android:orientation="vertical">

        <TextView
            android:id="@+id/chatMsg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
            android:padding="12dp"
            android:text="abcdefgh"
```

```
        android:textSize="18sp" />

    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp">

    </FrameLayout>

  </LinearLayout>
</FrameLayout>
```

Scroll View:

```
<ScrollView
  android:id="@+id/chatScrollView"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_above="@+id/inputLayout">

  <LinearLayout
    android:id="@+id/chatLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"></LinearLayout>


</ScrollView>
```

Result:

Figure 3.1

Create Shopping List Code snippet:

```
FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(view -> {
  AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
  builder.setTitle("Create Shopping List");

  EditText editText = new EditText(MainActivity.this);
  editText.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_FLAG_CAP_WORDS);
  editText.setHint("Type a name");
  editText.setHintTextColor(Color.GRAY);
  builder.setView(editText);

  builder.setPositiveButton("Create", (dialogInterface, i) -> {
     String shoppingListName = editText.getText().toString().trim();
     addShoppingList(shoppingListName);
  });

  builder.setNegativeButton("Cancel", (dialogInterface, i) -> dialogInterface.dismiss());

  AlertDialog alertDialog = builder.create();
  alertDialog.show();
});

userShoppingListsRef =
rootRef.collection("shoppingLists").document(userEmail).collection("userShoppingLists");

RecyclerView recyclerView = findViewById(R.id.recycler_view);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
TextView emptyView = findViewById(R.id.empty_view);
ProgressBar progressBar = findViewById(R.id.progress_bar);

Query query = userShoppingListsRef.orderBy("shoppingListName",
Query.Direction.ASCENDING);

FirestoreRecyclerOptions<ShoppingListModel> firestoreRecyclerOptions = new
FirestoreRecyclerOptions.Builder<ShoppingListModel>()
      .setQuery(query, ShoppingListModel.class)
      .build();
```

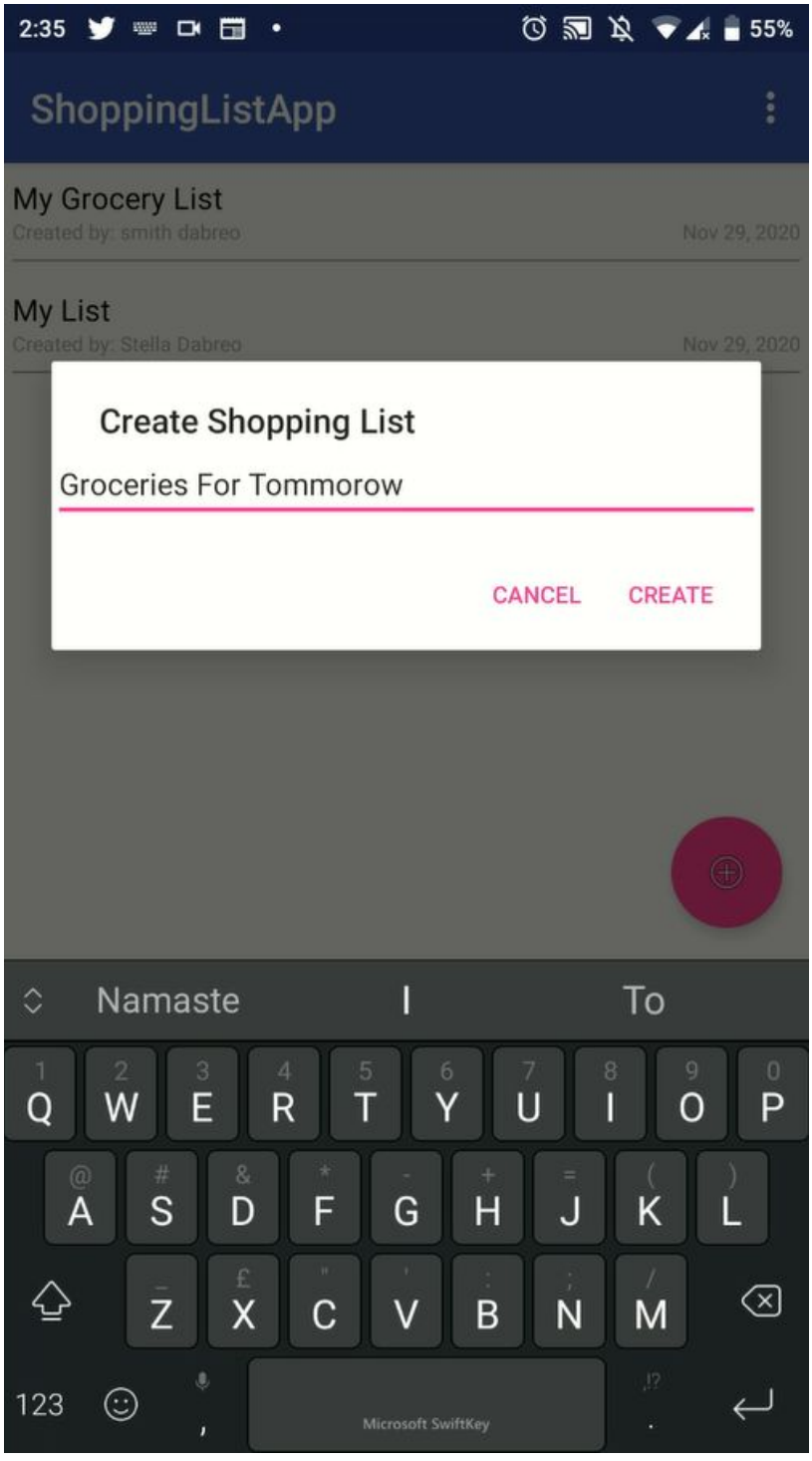
2:35
55%
ShoppingListApp
My Grocery List
Created by: smith dabreo
Nov 29, 2020
My List
Created by: Stella Dabreo
Nov 29, 2020
Create Shopping List
Groceries For Tommorow
CANCEL
CREATE
Namaste
I
To
Microsoft SwiftKey

Figure 3.2

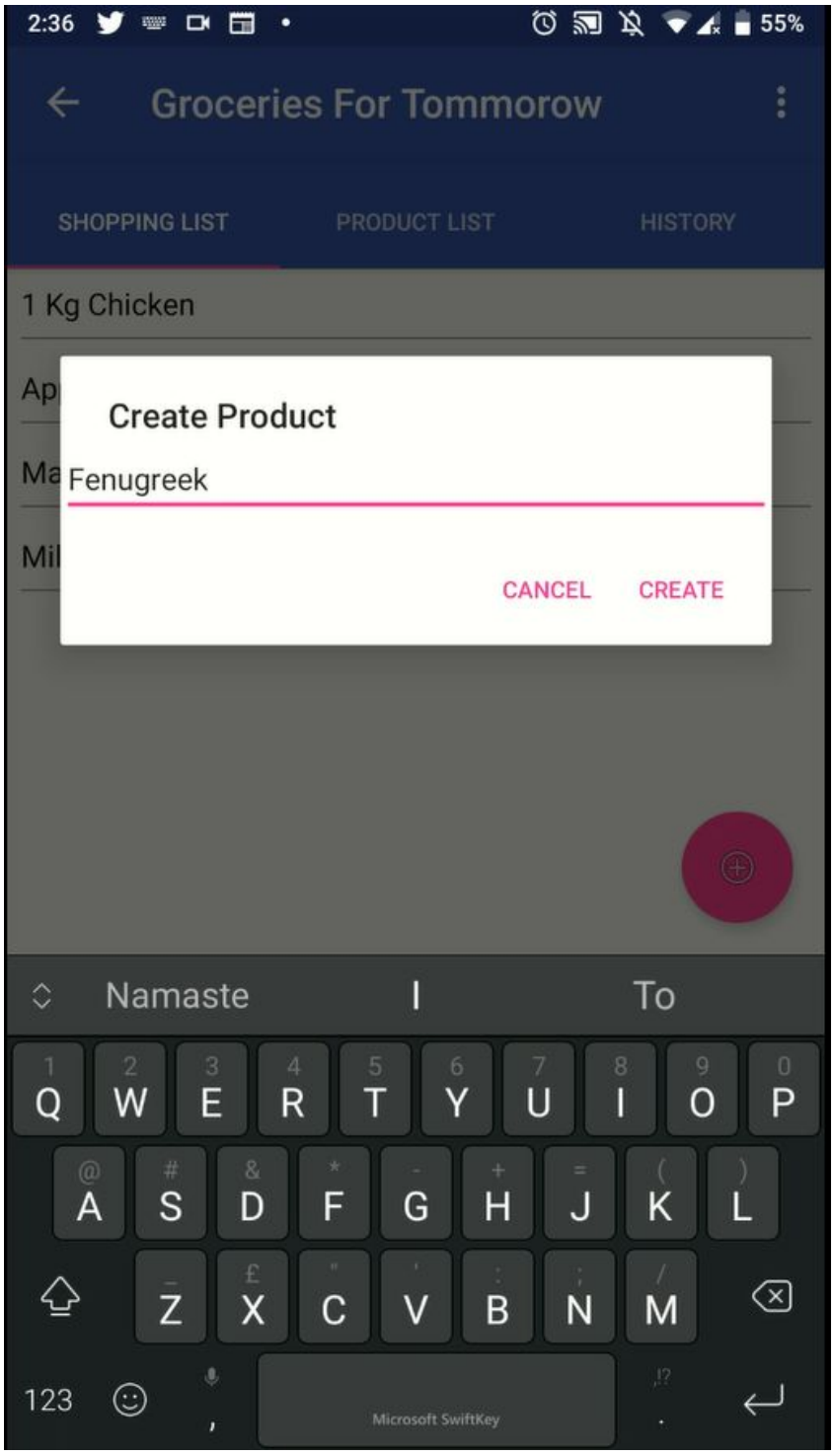
2:36
55%
Groceries For Tommorow
SHOPPING LIST
PRODUCT LIST
HISTORY
1 Kg Chicken
Create Product
Fenugreek
CANCEL
CREATE
Namaste
I
To
Microsoft SwiftKey

Figure 3.3

Chapter 4
Intents

Android Intent is the message that is passed between components such as activities, content providers, broadcast receivers, services etc. It is generally used with startActivity() method to invoke activity, broadcast receivers etc. The dictionary meaning of intent is intention or purpose. So, it can be described as the intention to do action. The LabeledIntent is the subclass of android.content.Intent class. There are two types of intents in android: implicit and explicit. Implicit Intent doesn't specify the component. Explicit Intent specifies the component.

Explicit intents specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name. You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.

Implicit intents do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

Starting an activity

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.

If you want to receive a result from the activity when it finishes, call startActivityForResult(). Your activity receives the result as a separate Intent object in your activity's onActivityResult() callback. For more information, see the Activities guide.

Starting a service

A Service is a component that performs operations in the background without a user interface. With Android 5.0 (API level 21) and later, you can start a service with JobScheduler. For more information about JobScheduler, see its API-reference documentation.

For versions earlier than Android 5.0 (API level 21), you can start a service by using methods of the Service class. You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.

If the service is designed with a client-server interface, you can bind to the service

from another component by passing an Intent to bindService(). For more information, see the Services guide.

Delivering a broadcast
A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().
The rest of this page explains how intents work and how to use them. For related information, see Interacting with Other Apps and Sharing Content.

Android Intent

Here, we have three activity pages. First is the home screen, from here the user can navigate to the explore activity and to the chat room.

Java Code:

FrontActivity.java INTENT

```
public class FrontPage extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_front_page);
        Button button;
        button =(Button)findViewById(R.id.Button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openMainAct();
            }
        });
    }
    public void openMainAct(){
        Intent intent = new Intent(this, MainActivity.class);
    }
```

MainActivity.java INTENT:

```
public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_front_page);
        Button button;
        button =(Button)findViewById(R.id.Button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openMainAct();
            }
        });
    }
    public void openMainAct(){
        Intent intent = new Intent(this, Main2Activity.class);
        startActivity(intent);
    }
}
```

Chapter 5
Activity

An activity is the single screen in android. It is like window or frame of Java. By the help of activity, you can place all your UI components or widgets in a single screen. All activities in your Android apps are represented by an activity class. These activity classes are subclasses of android.app.Activity. The Activity class contains a set of methods that corresponds to the life cycle states an activity can be in.

Activity Life Cycle:

Any Android activity goes through a certain life cycle during its life inside the Android app. This life cycle is illustrated here:

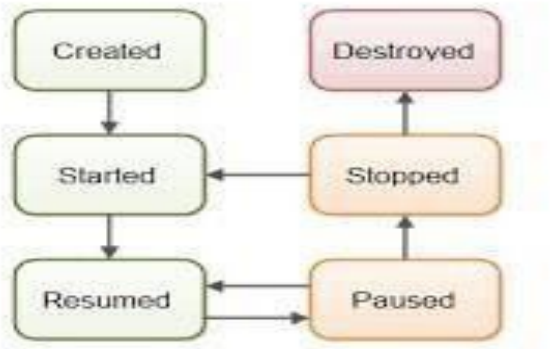

Figure 5.1

Android Activity

onCreate( ) is method called when activity is first created. You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity.

Java Code:

```
public class FrontPage extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_front_page);
        Button button;
```

```
        button =(Button)findViewById(R.id.Button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openMainAct();
            }
        });
    }
    public void openMainAct(){
        Intent intent = new Intent(this, MainActivity.class);
    }
}
```

Main Activity code Snippet:

```
private void signOut() {
  Map<String, Object> map = new HashMap<>();
  map.put("tokenId", FieldValue.delete());

  rootRef.collection("users").document(userEmail).update(map).addOnSuccessListener(aVoid
-> {
     firebaseAuth.signOut();

     if (googleApiClient.isConnected()) {
        Auth.GoogleSignInApi.signOut(googleApiClient);
     }
  });
}

private void addShoppingList(String shoppingListName) {
  String shoppingListId = userShoppingListsRef.document().getId();
  ShoppingListModel shoppingListModel = new ShoppingListModel(shoppingListId,
shoppingListName, userName);

userShoppingListsRef.document(shoppingListId).set(shoppingListModel).addOnSuccessListene
r(aVoid -> Log.d("TAG", "Shopping List successfully created!"));
}
```


12:56
61%
ShoppingListApp
Sign-out
My Grocery List
Created by: smith dabreo
Nov 29, 2020
My List
Created by: Stella Dabreo
Nov 29, 2020

Figure 5.2

Chapter 6

API

API stands for “application program interface.” In short, this allows developers to access the platform or application of another party. ... RESTful APIs are commonly referred to as RESTful web services because they implement REST principles as well HTTP protocols.

Google SignIn API:

Firebase enables users with google’s SignIn API. This API makes process of authenticating users very easy and all the necessary info like the email is stored using this API for future uses. This makes the App more secure and users are authenticated via their Gmail accounts. This makes it more accessible for user as well as the developer.

API snippet from ShoppingListActivity.java:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_shopping_list);

  GoogleSignInAccount googleSignInAccount = GoogleSignIn.getLastSignedInAccount(this);
  if (googleSignInAccount != null) {
     userEmail = googleSignInAccount.getEmail();
  }

  googleApiClient = new GoogleApiClient.Builder(this)
        .addApi(Auth.GOOGLE_SIGN_IN_API)
        .build();

  firebaseAuth = FirebaseAuth.getInstance();
  rootRef = FirebaseFirestore.getInstance();
```

```
  authStateListener = firebaseAuth -> {
      FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
      if (firebaseUser == null) {
          Intent intent = new Intent(ShoppingListActivity.this, LoginActivity.class);
          startActivity(intent);
      }
  };
```

From LoginActivity.java:

```
@Override
  protected void onStart() {
      super.onStart();
      firebaseAuth.addAuthStateListener(authStateListener);
  }

  private void firebaseSignInWithGoogle(GoogleSignInAccount googleSignInAccount) {
      AuthCredential authCredential =
GoogleAuthProvider.getCredential(googleSignInAccount.getIdToken(), null);
      firebaseAuth.signInWithCredential(authCredential).addOnCompleteListener(this, task -> {
          if (task.isSuccessful()) {
              String userEmail = googleSignInAccount.getEmail();
              String userName = googleSignInAccount.getDisplayName();
              String tokenId = FirebaseInstanceId.getInstance().getToken();

              UserModel userModel = new UserModel(userEmail, userName, tokenId);

rootRef.collection("users").document(userEmail).set(userModel).addOnSuccessListener(aVoid
-> Log.d("TAG", "User successfully created!"));
          } else {
```

```
                Log.d("TAG", "Failed with: " + task.getException());
            }
        });
    }
}
```

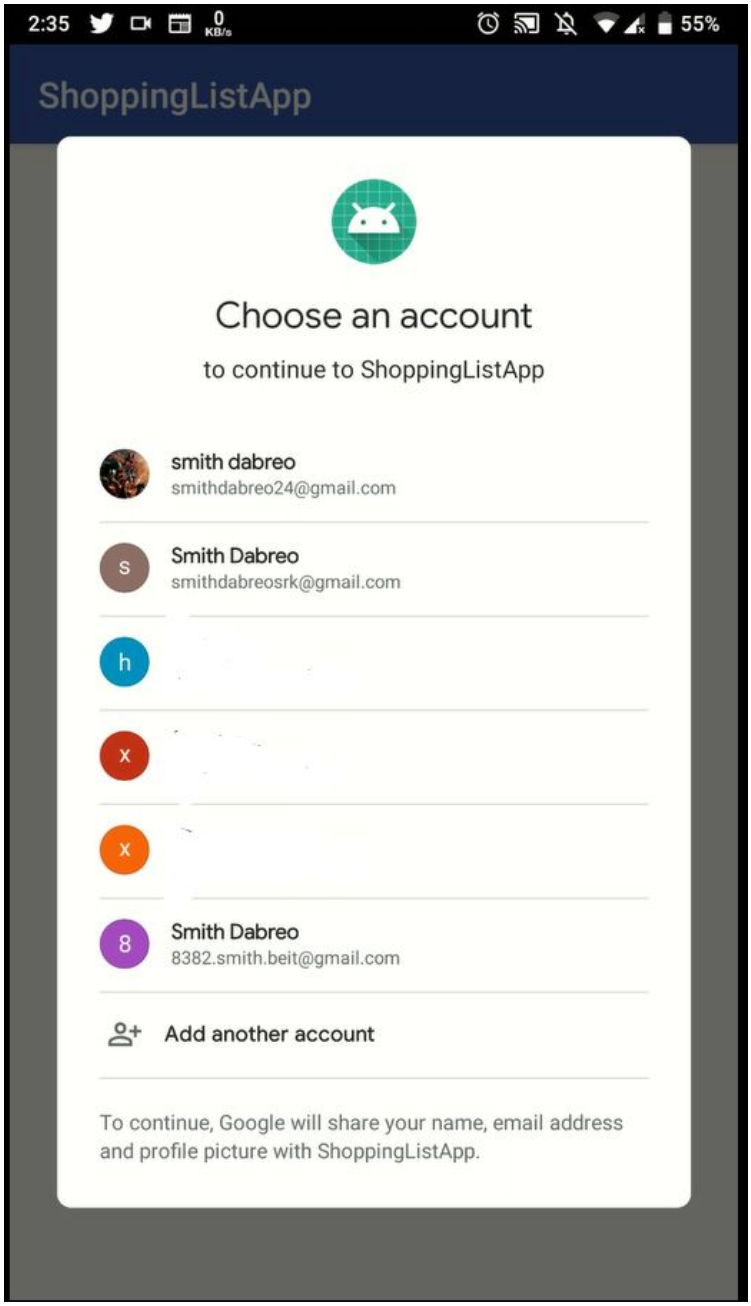

Figure 6.1

Chapter 7

Database Connectivity:

To implement database we have used Google Firebase.

Firebase is a mobile- and web application development platform, backed by Google, to help developers deliver richer app experiences. Firebase manages its own infrastructure with a nice set of tools to simplify the workflow of the developer by providing them with development kits and an online dashboard. These toolkits are interconnected, scalable and integrable with third party software to overcome complex challenges with standard building blocks. The platform consists of a great set of development tools. The Realtime Database and Cloud Firestore can stock document-structured data and synchronize the corresponding apps in milliseconds whenever a data transformation occurs. This means that both the app and its database listen to each other, providing the user with reactive app experiences. And Firebase Cloud Functions can even extend this functionality. These functions allow the developer to write backend code to respond to events happening in the Firebase platform without having to deal with any servers.

For connecting the database when we build the project a custom made JSON file is provided and awe have to give SHA-1 signature certificate for security purposes. This helps the App to function properly with any error and with an extremely secure database

we also used Google Firebase for authenticating the user so each and every user can have their unique account with complete security. This was implemented with google auth API.

The data is stored securely in Firestore which provides complete control over the database.

Here is the code

JSON code(googlr-services.json) :

```
{
  "project_info": {
    "project_number": "76143155752",
    "firebase_url": "https://my-shopping-list-93137.firebaseio.com",
    "project_id": "my-shopping-list-93137",
    "storage_bucket": "my-shopping-list-93137.appspot.com"
  },
  "client": [
    {
```

```
    "client_info": {
      "mobilesdk_app_id": "1:76143155752:android:42699d28379e2d35f0f31d",
      "android_client_info": {
        "package_name": "example.com.shoppinglistapp"
      }
    },
    "oauth_client": [
      {
                                                                    "client_id":
"76143155752-3509th11a68be0v29a2mhrpfp1mmo867.apps.googleusercontent.com",
        "client_type": 1,
        "android_info": {
          "package_name": "example.com.shoppinglistapp",
          "certificate_hash": "e33e0fc77fb94b247fbe58fd90632ec6b9b308e4"
        }
      },
      {
                                                                    "client_id":
"76143155752-39r49mvok520s42qsul84s1k2vp5e8nt.apps.googleusercontent.com",
        "client_type": 3
      }
    ],
    "api_key": [
      {
        "current_key": "AIzaSyAw37M3Xlm2MK7wS9L6mSo7qvFNvfz8IpQ"
      }
    ],
    "services": {
      "appinvite_service": {
```

```
        "other_platform_oauth_client": [
          {
                                                                    "client_id":
"76143155752-39r49mvok520s42qsul84s1k2vp5e8nt.apps.googleusercontent.com",
            "client_type": 3
          }
        ]
      }
    }
  }
 ],
 "configuration_version": "1"
}
```

build.gradle changes :

```
//Firebase
  compile 'com.google.firebase:firebase-auth:11.8.0'
  compile 'com.google.firebase:firebase-firestore:11.8.0'
  compile 'com.google.firebase:firebase-messaging:11.8.0'

  //FirebaseUI
  compile 'com.firebaseui:firebase-ui-auth:3.1.3'
  compile 'com.firebaseui:firebase-ui-firestore:3.1.3'

  //Google Play Services
  compile 'com.google.android.gms:play-services-auth:11.8.0'
}
apply plugin: 'com.google.gms.google-services'
```

Project level build.gradle

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {

  repositories {
    google()
    jcenter()
  }
  dependencies {
    classpath 'com.android.tools.build:gradle:3.0.1'
    classpath 'com.google.gms:google-services:3.1.0'


    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
  }
}

allprojects {
  repositories {
    google()
    jcenter()
  }
}

task clean(type: Delete) {
  delete rootProject.buildDir
```

}

Here are some snapshots of the Database Structure :

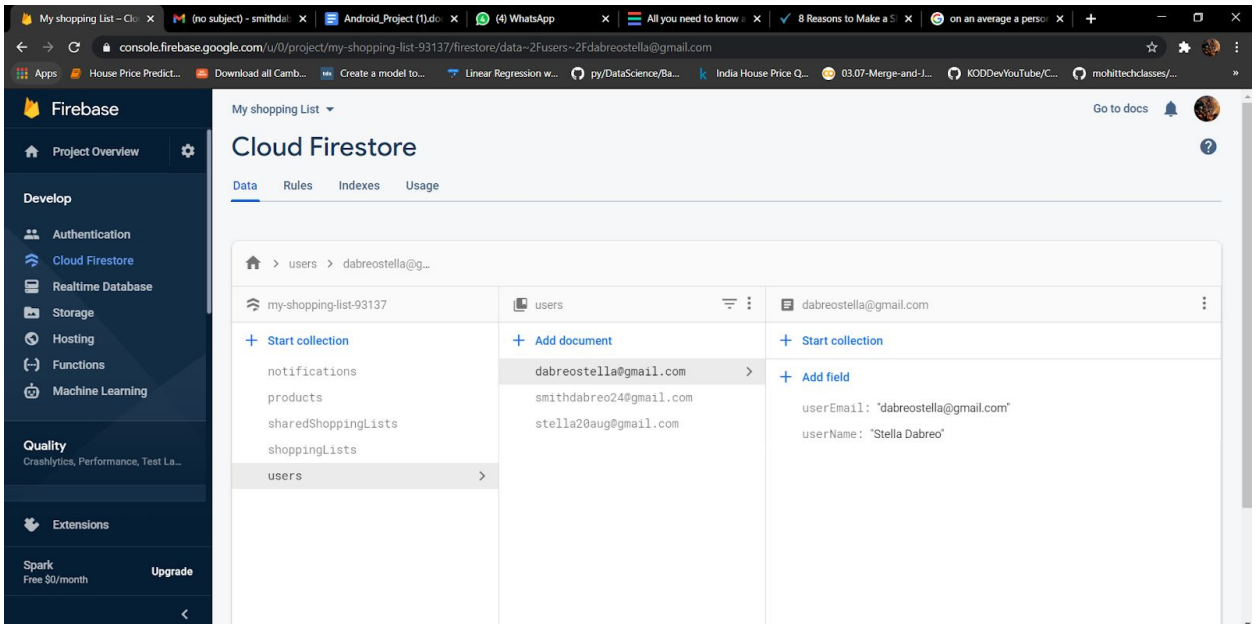

Figure 7.1

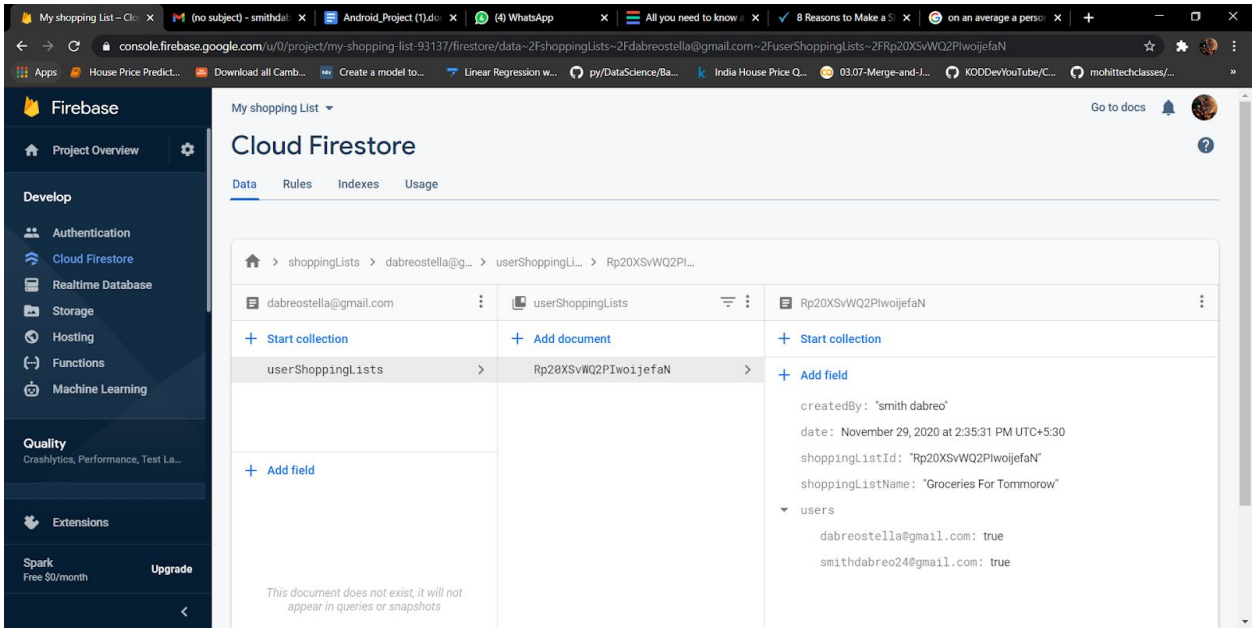

Figure 7.2

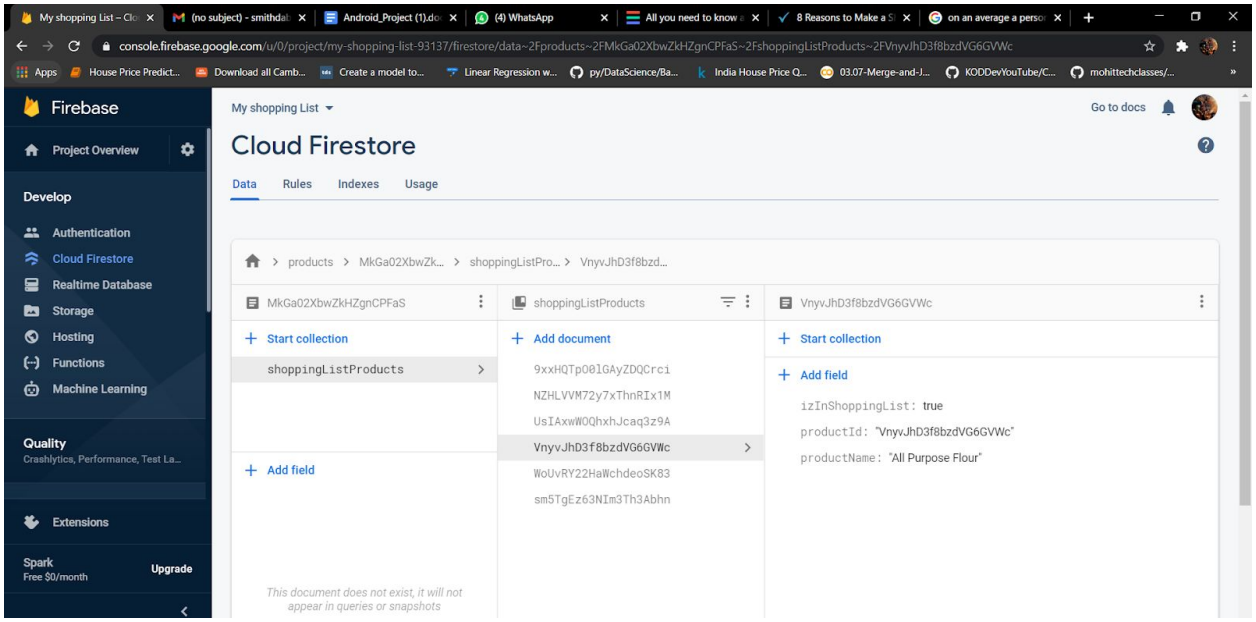

Figure 7.3

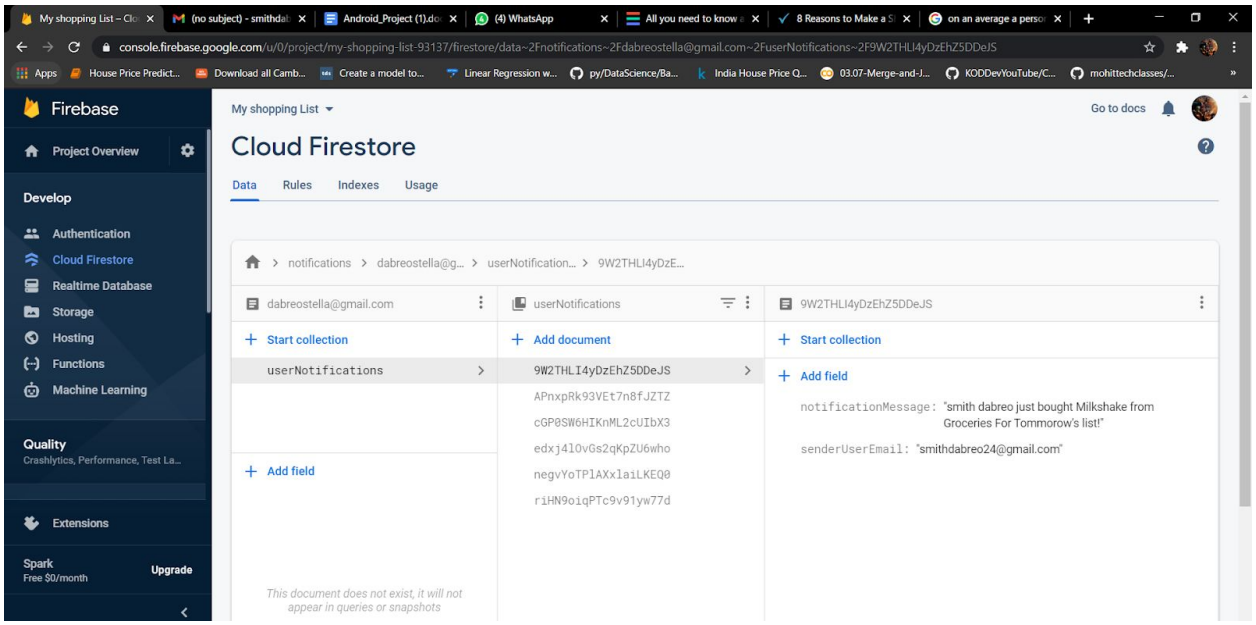

Figure 7.4

Firestore Indexes:

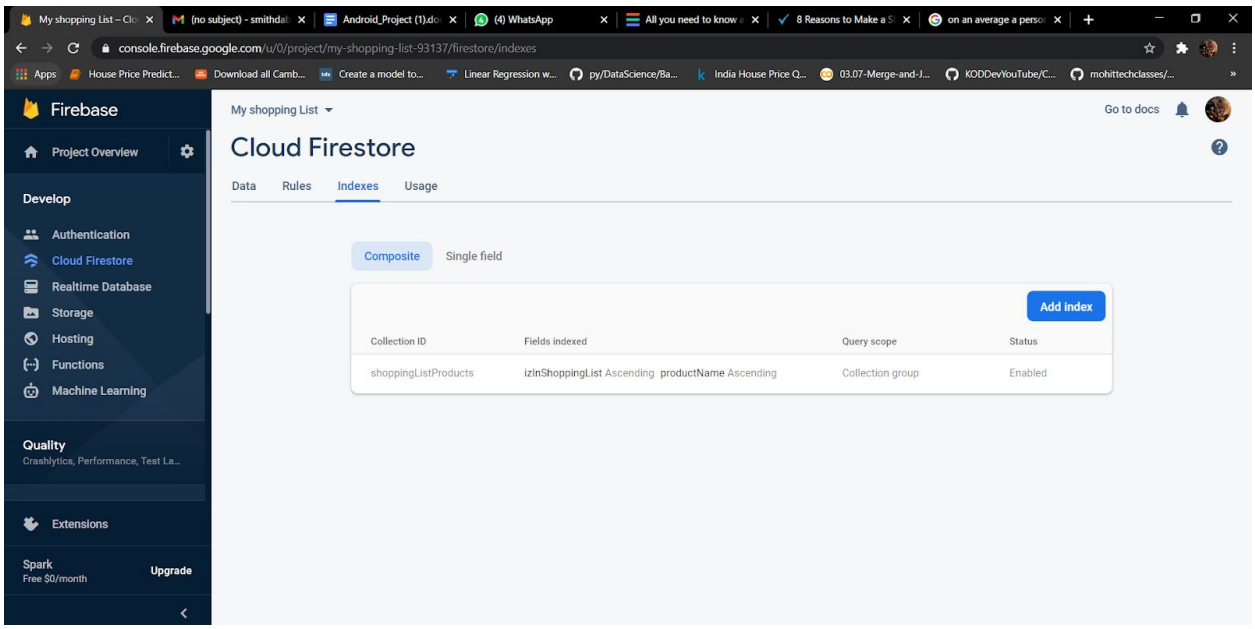

Figure 7.5

Google Authentication :

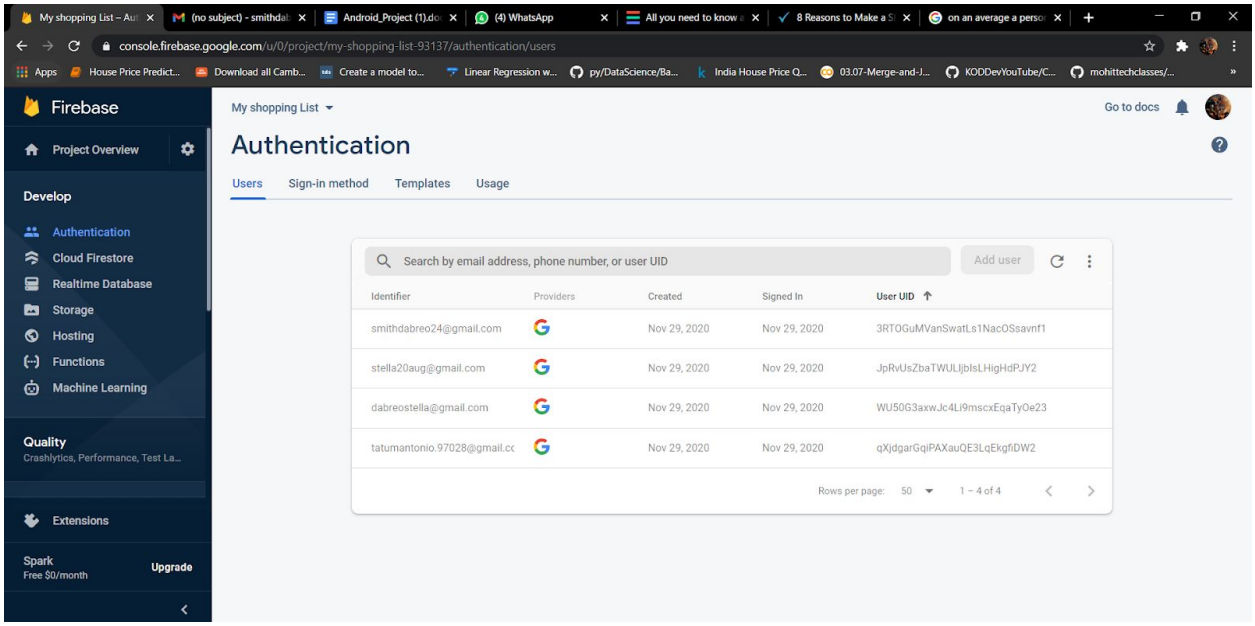

Figure 7.6

Chapter 8

Results

Login Page(Google Authentication)

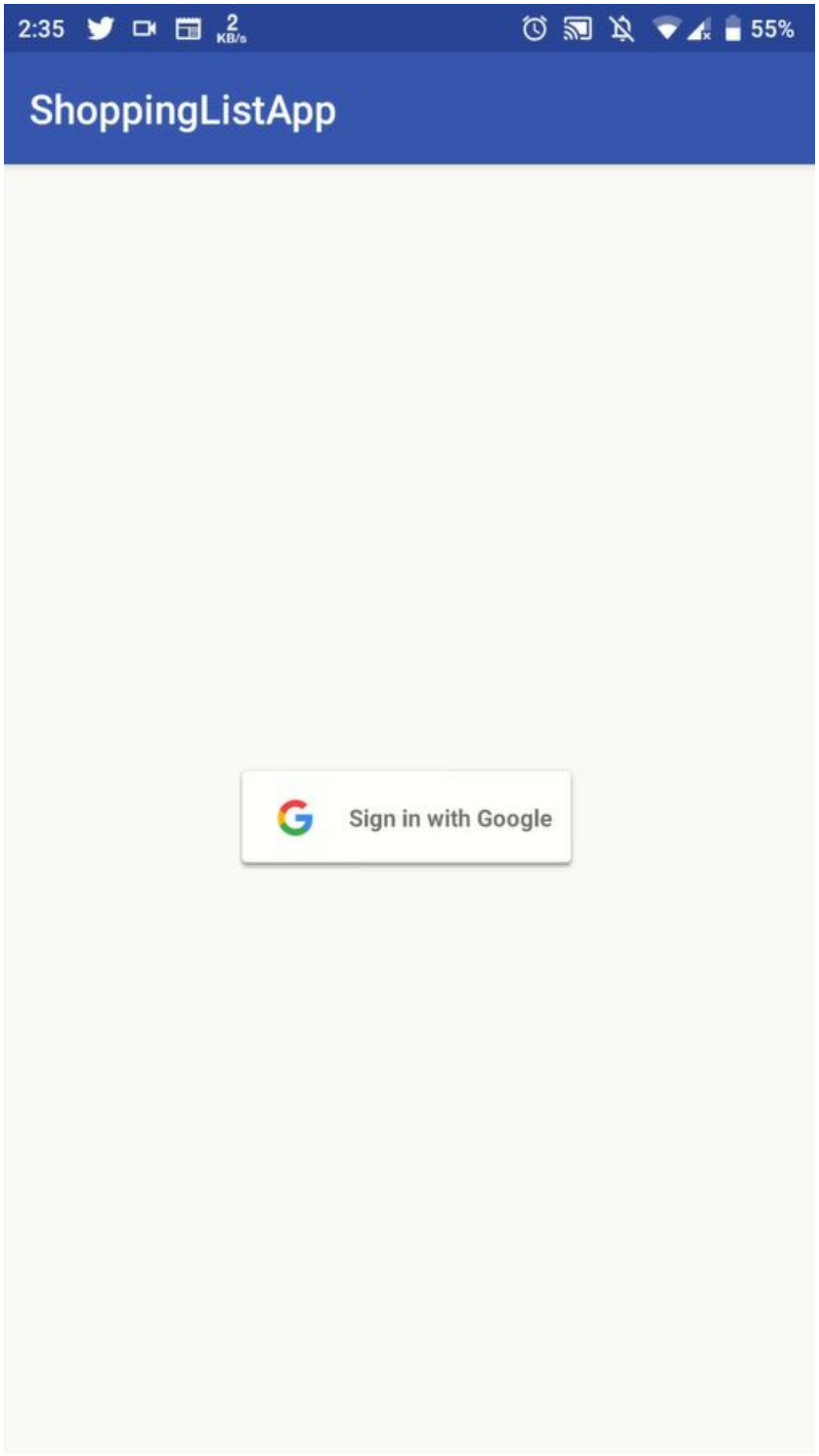

Figure 8.1

Main Page:

Figure 8.2

Creating A Shopping List:

1) Giving a Name

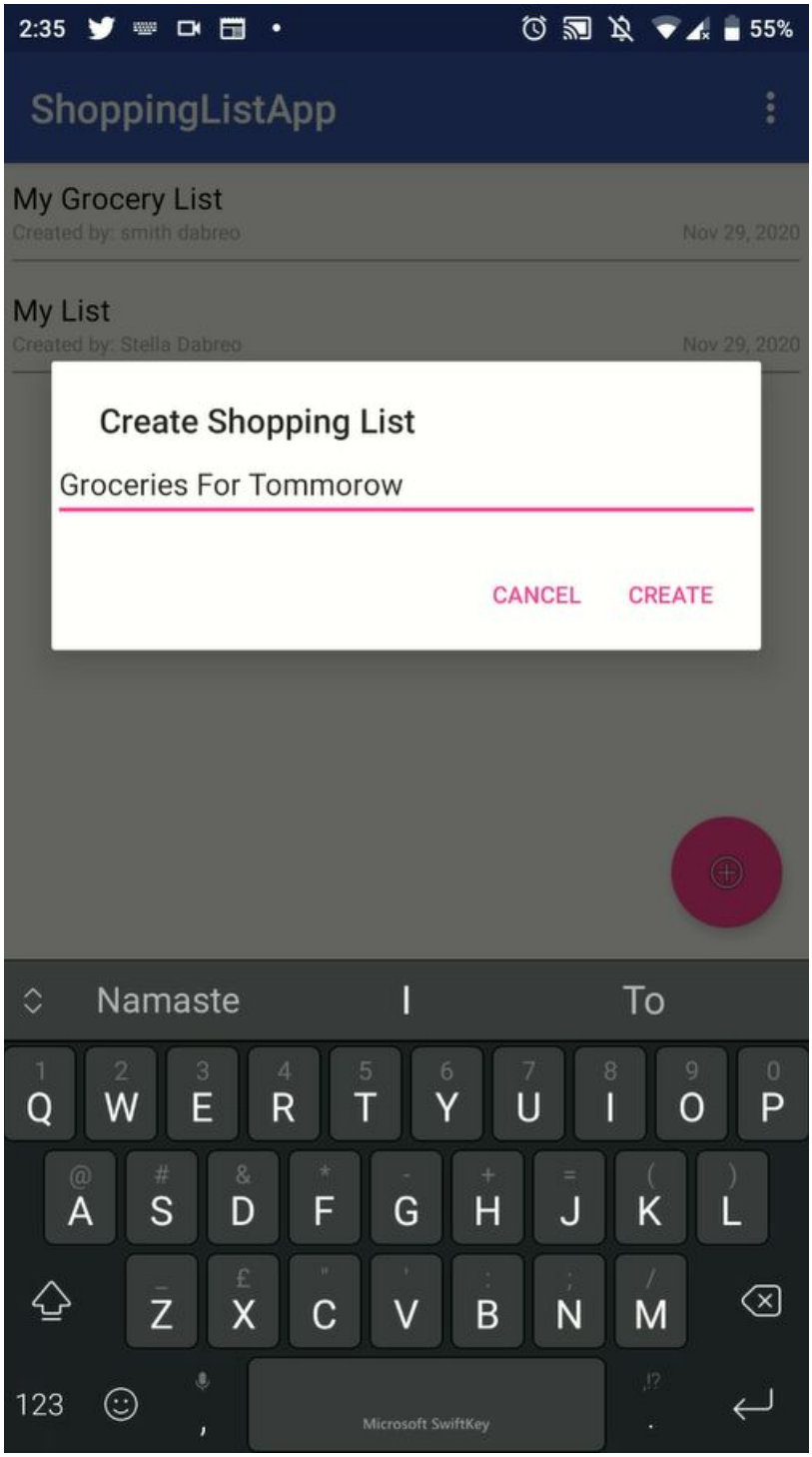

Figure 8.3

2) Creating Products you want to buy

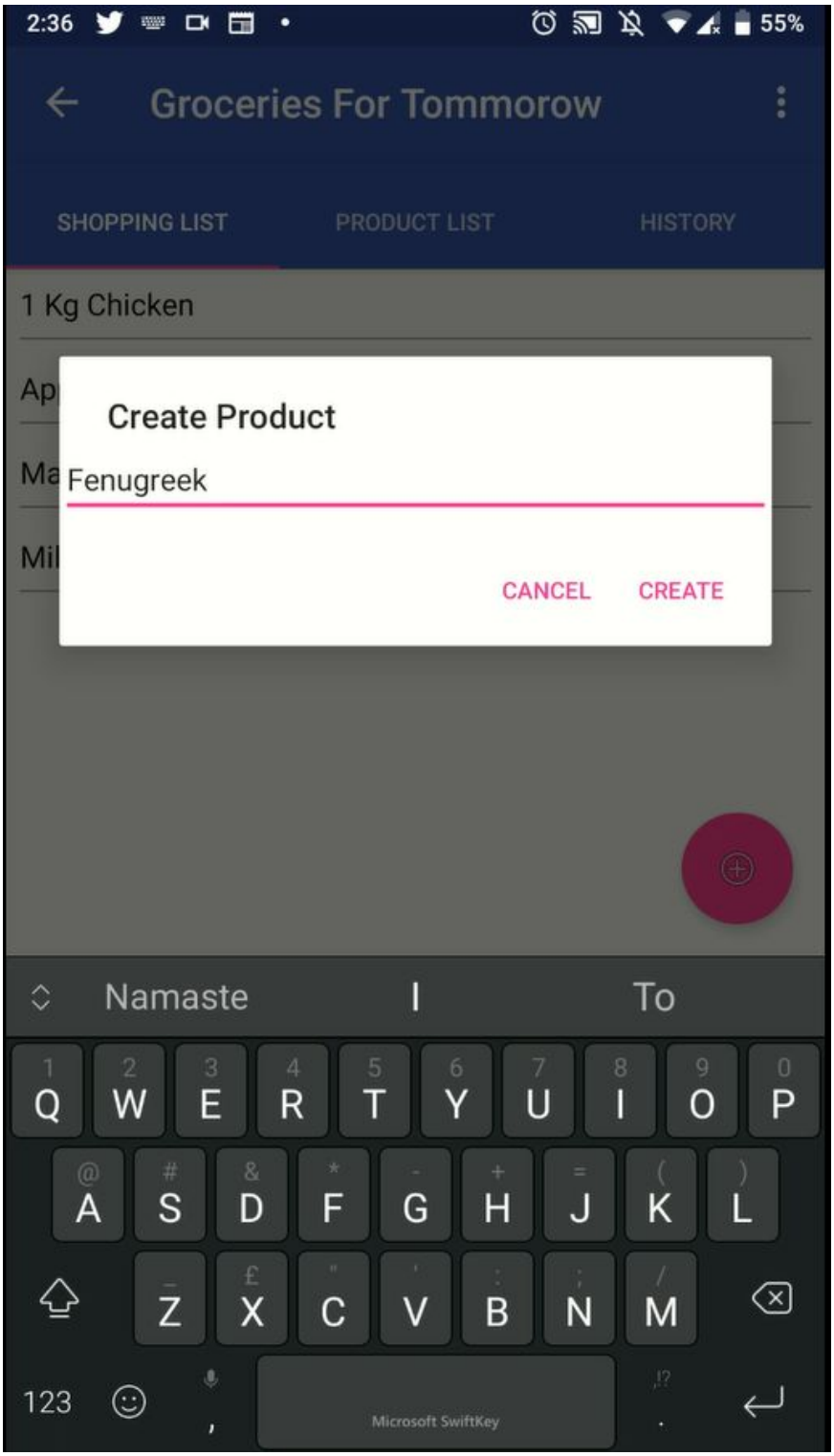

Figure 8.4

Sharing Shopping List WIth Friends :

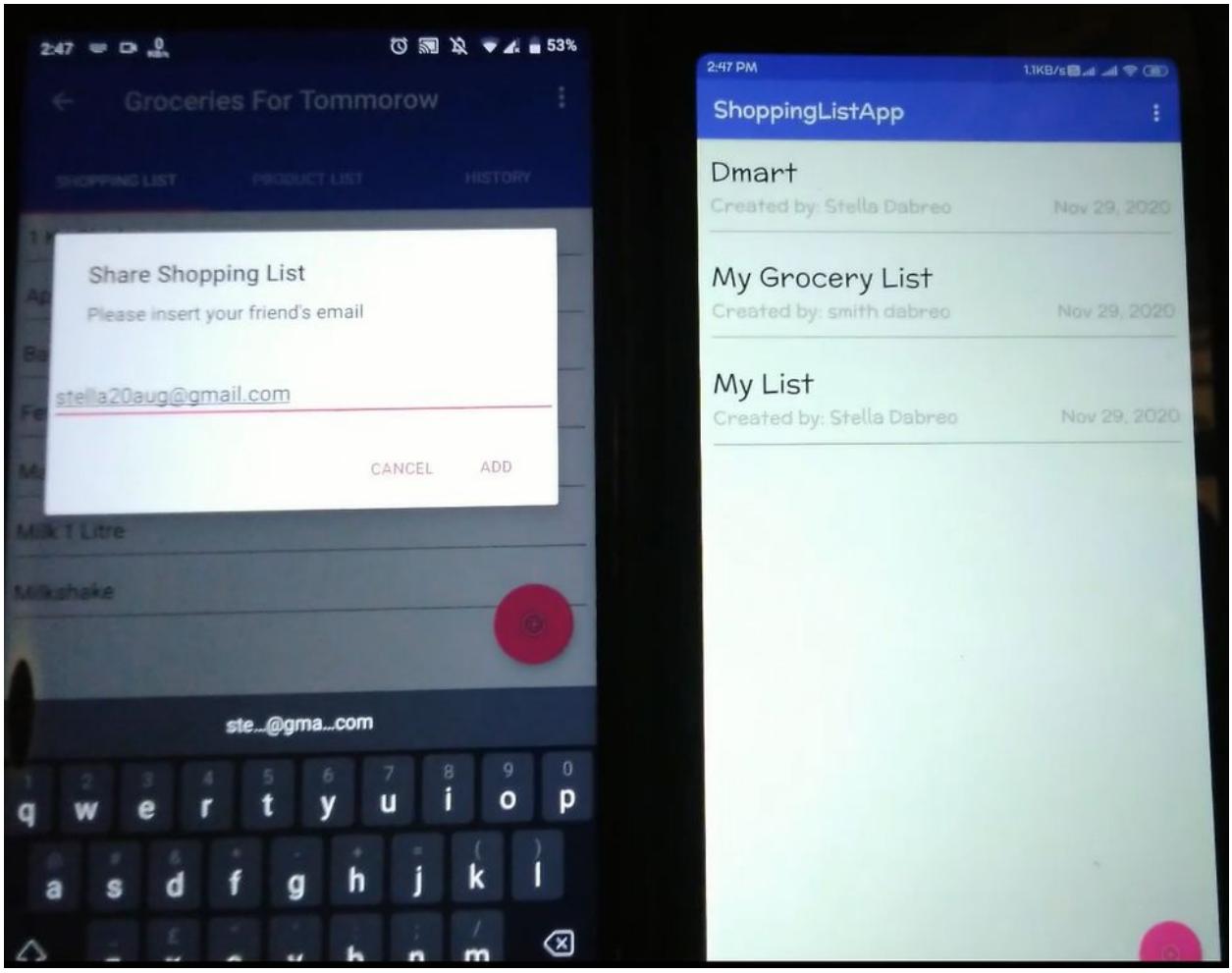

Figure 8.5

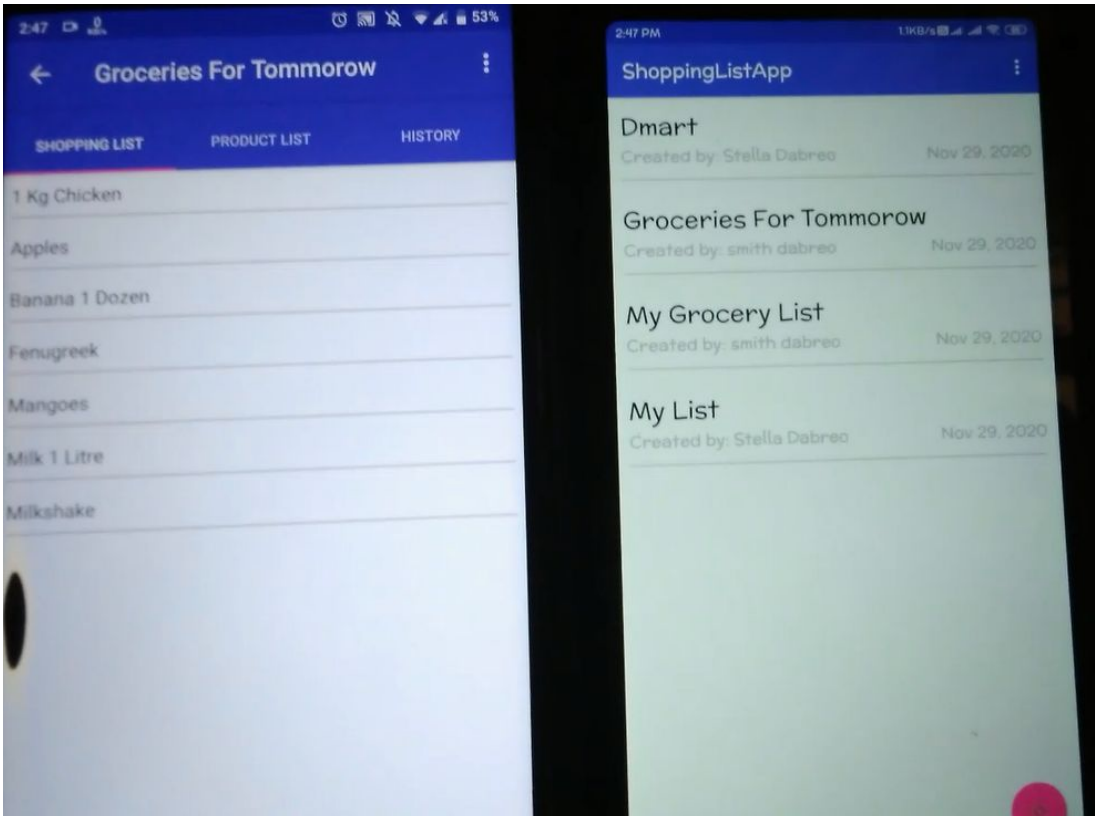

Figure 8.6

Buying Products:(Bought products get moved to Product list From the Shopping List)

Realtime Updates in shared shopping lists

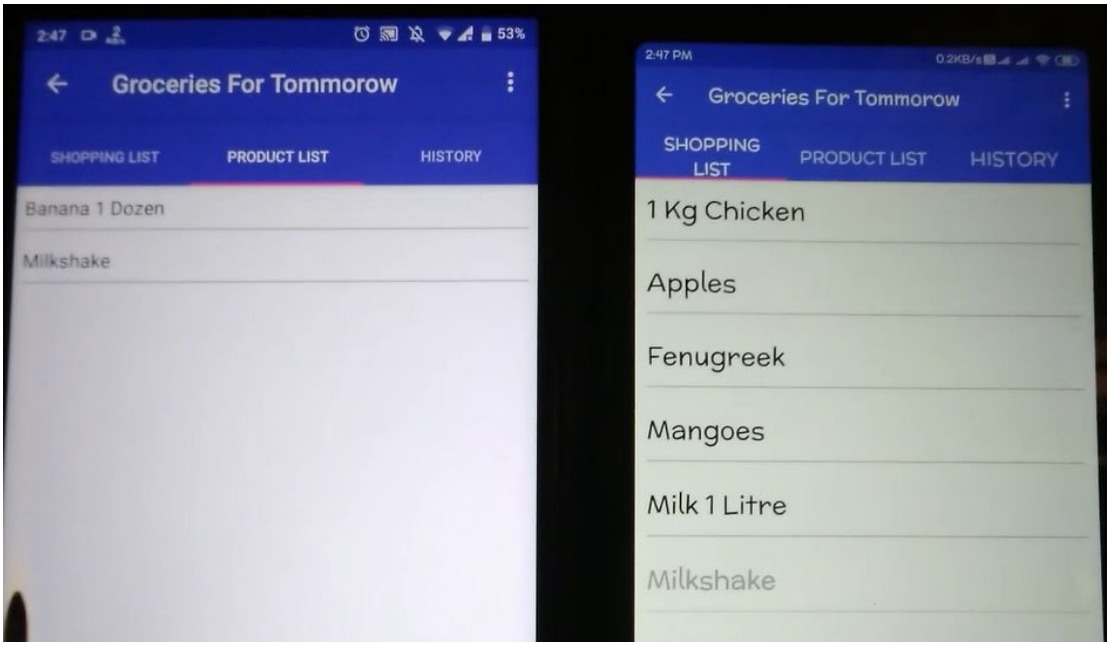

Figure 8.7

Editing Shopping List Name:

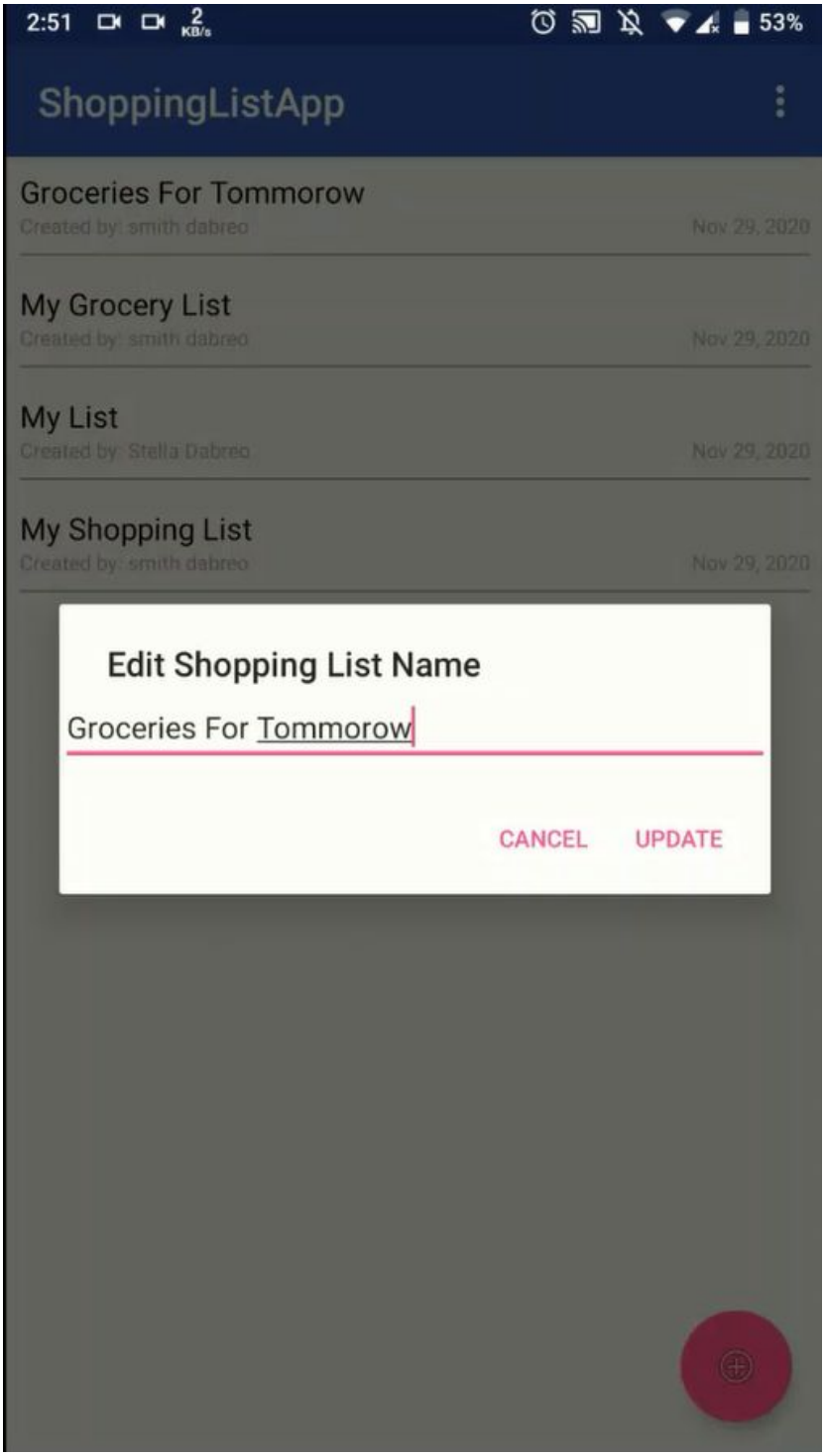

Figure 8.8

Editing/Deleting Products:

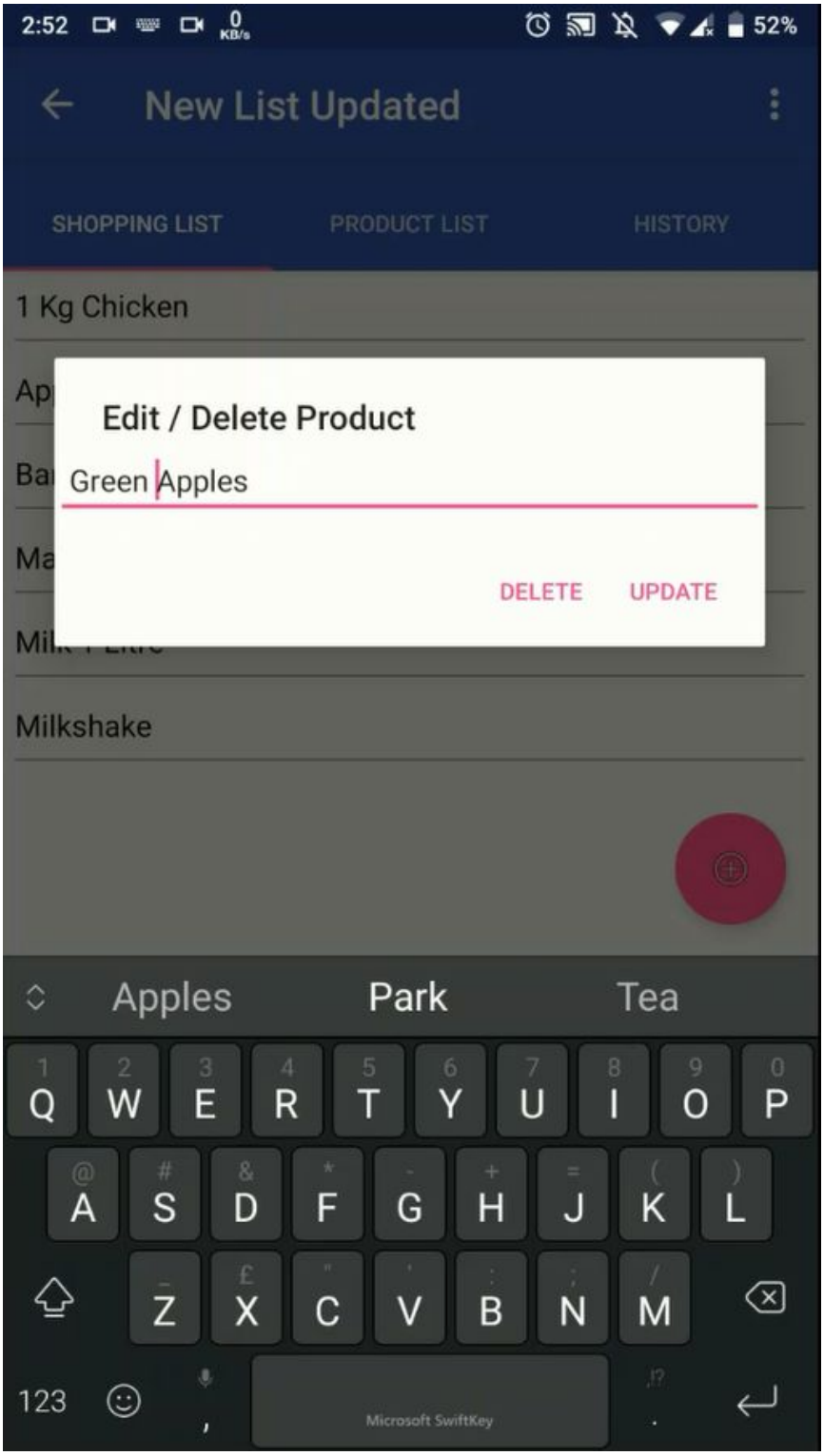
2:52
52%
New List Updated
SHOPPING LIST
PRODUCT LIST
HISTORY
1 Kg Chicken
Edit / Delete Product
Green Apples
DELETE
UPDATE
Milkshake
Apples
Park
Tea
Microsoft SwiftKey

Figure 8.9

Database Results:

Firebase Dashboard:

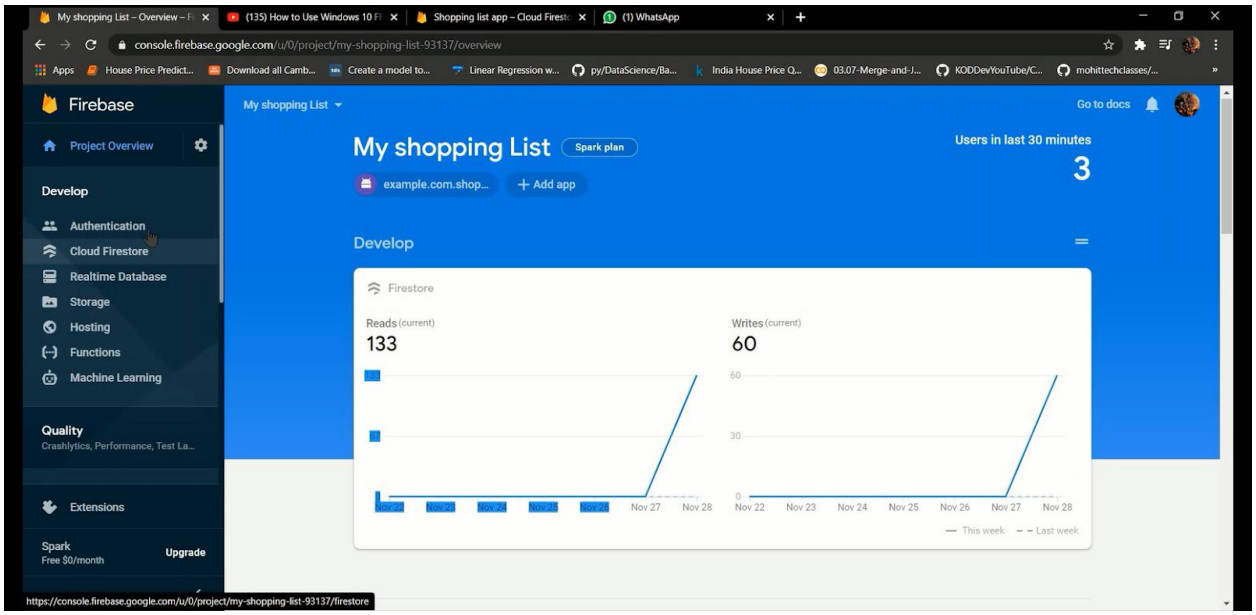

Figure 8.10

Chapter 9

Generating APK

Android Package (APK) is the package file format used by the Android operating system for distribution and installation of mobile apps and middleware. To make an APK file, a program for Android is first compiled, and then all of its parts are packaged into one container file. An APK file contains all of a program's code (such as .dex files), resources, assets, certificates, and manifest file. As is the case with many file formats, APK files can have any name needed, provided that the file name ends in the file extension ".apk". APK files are a type of archive file, specifically in zip format-type packages, based on the JAR file format, with .apk as the filename extension.

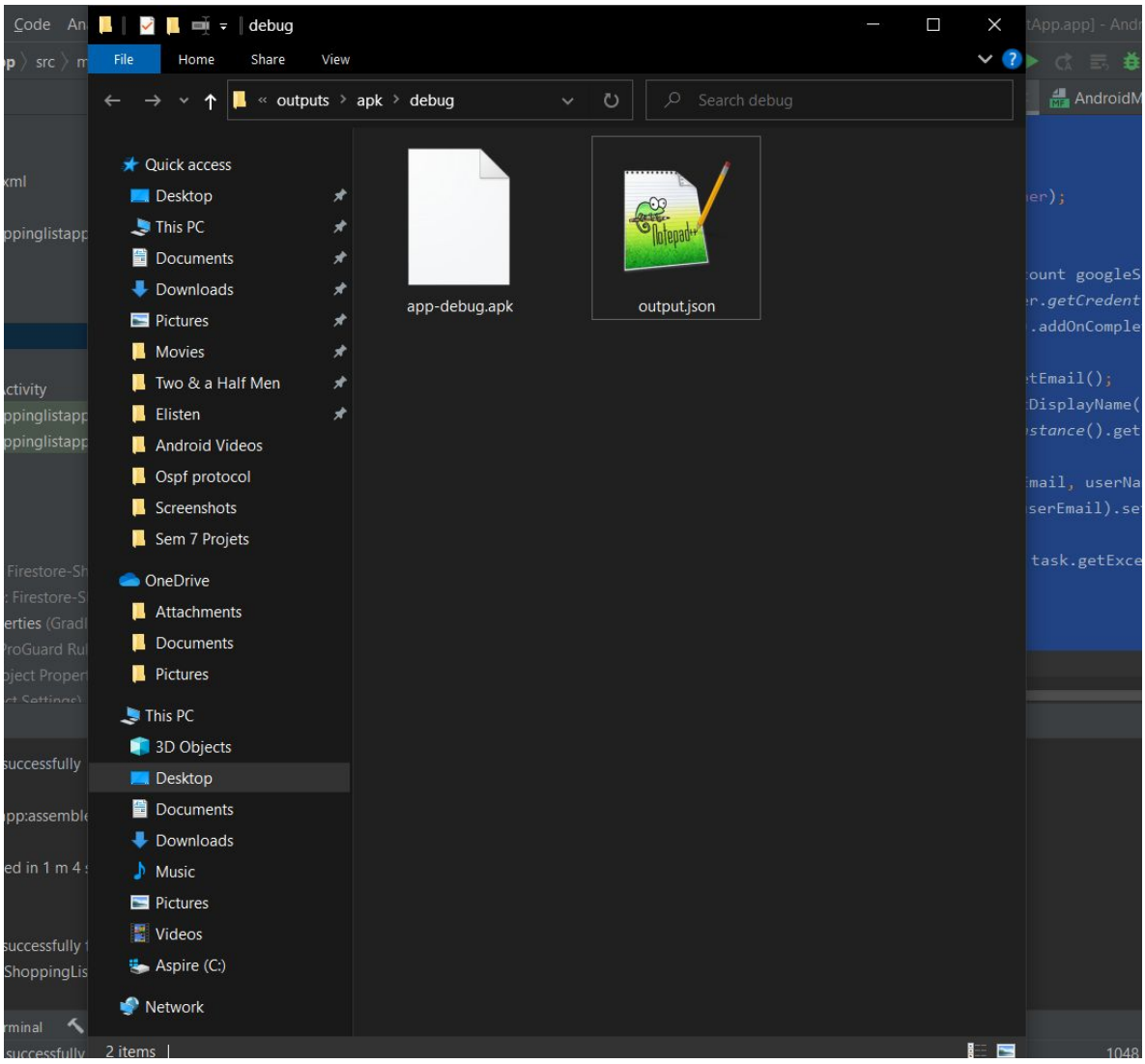

Figure 9.1

After installing the Shopping List Apk on an android phone, the app will appear as an icon on the home screen.

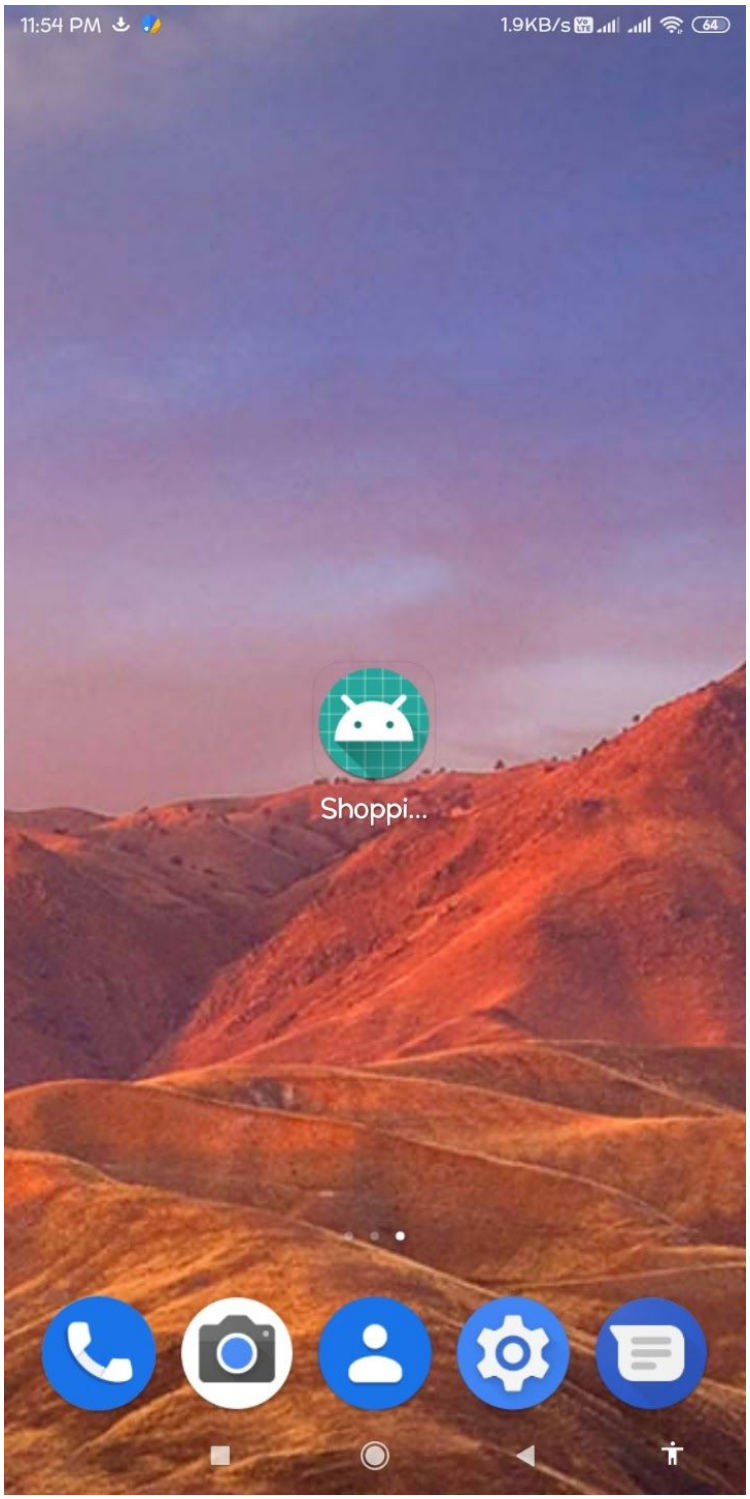

Figure 9.2

Conclusion

Digital Shopping Lists are considered a more reliable and accurate alternative to the conventional lists which waste paper and other resources for just a simple reason. This simple yet effective solution is needed to help people and their lifestyle. Everything should go digital like shopping lists as this world is evolving into a digital world in this internet era. This can help even help people with Alzheimer and other memory related disease, someone can just make a list for them and then they don't have to worry about forgetting anything as everything is safe and secure in the application. This can be used 24x7 without facing any issues with just a click of a finger. The interesting fact about this app is it i highly scalable and versatile as it can be used for purposes other then shopping list. For instance, it can be used as a app of saving recipes and sharing these recipes with you friends and family. Moreover, it can be converted into a To-Do list where you can manage your tasks and move them from done of to be done, etc. All these functionalities already exist in the app and its upto the user and the creativity of his/her mind how they chose to use this application.

.

References

1. https://firebase.google.com/support
2. https://stackoverflow.com/
3. https://developer.android.com/studio
4. https://www.w3schools.com/js/default.asp
5. https://www.udemy.com/
6. https://scholar.google.co.in/scholar?q=android+research+paper&hl=en&as_sdt=0&as_vis=1&oi=scholart