GROCERY APP

TITLE : To build an grocery app by using android.
SOFTWARE REQUIRED : android studio.2021.2.0.0 and some other tools.

INTRODUCTION : An online grocer is either a brick-and-mortar supermarket or grocerystore that allows online ordering, or a standalone e-commerce service that includes grocery items.There is usually a delivery charge for this service. Brick-and-mortar supermarkets that have built internet channels to better service their clients are knownas online grocers. Online grocery delivery services are available throughout Europe, Asiaand North America, mostly in urban c enters. The online ordering is done through e- commerce websites or mobile apps. The COVID-19 pandemic greatly accelerated the growth of online grocers, and in the first few months of the pandemics online groceryshopping increased by 300%. In addition, firsttime online grocery shoppers accounted for 41 percent of online grocery shoppers.The epidemic of COVID-19 has hastened the uptake of online grocery shopping. Pre-COVID-19 food shopping activity accounted for 9percent of the market, but 63 percent of consumers worldwide purchased more groceries online after the outbreak than they did before they were socially isolated.
Most local online grocers have their own drivers. The most common type of personal delivery involves storing grocery inventory in a warehouse to deliver to customers once orders are placed. Another type of personal delivery which is less common is based onjust-in-time business in which there is no warehouse or inventory. In this type of delivery,customers place orders for next-day delivery. The online grocer shops for the groceries on the morning of the delivery day. Some grocery fulfilment c enters are set up as dark stores. Online-only grocers typically have warehouses or distribution c enters nearby, to allow local shipping of refrigerated items. Online grocers with a large regional or national delivery area may ship groceries using courier services. If the order contains cold or frozen items, this involves "flash freezing" the goods and pack them into special shipping containers. Companies have experimented with automated delivery modes including drone delivery and robots. For instance, in Fall 2016 Washington, D.C. approved a trial run of rolling delivery drones produced by Star ship Technologies. The earthbound robots are similar to wheeled coolers and carry around 40 pounds of groceries. Online grocery stores may allow facilitating local food which may reduce the environmental impact of food transport. Small-scale farmers have been embracing digital technologies as a way to sell produce directly, and community-supported agriculture and direct-sell delivery systems are on the rise during the coronavirus pandemic.Furthermore, weekly grocery deliveries can be a better choice than individualtrips to a store.

Step by Step Implementation

Step 1: Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select **Kotlin** as the programming language.

Step 2: Before going to the coding section first you have to do some pre-task

Before going to the coding part first add these libraries in your gradle file and also apply the plugin as 'kotlin-kapt'. To add these library go to **Gradle Scripts > build.gradle(Module:app).**

```
def room_version = "2.2.1"
    def lifecycle_version = "2.0.0"
    // Room and Architectural Components
    implementation "androidx.room:room-runtime:$room_version"
    implementation "androidx.legacy:legacy-support-v4:1.0.0"
    implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.1.0'
    implementation "androidx.room:room-ktx:2.2.1"
    kapt  "androidx.room:room-compiler:$room_version"
    // Coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0'
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.0"
    // New Material Design
    implementation  "com.google.android.material:material:1.0.0"
    // ViewModel
```

implementation “androidx.lifecycle:lifecycle-extensions:$lifecycle_version”

implementation “androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version”

kapt “androidx.lifecycle:lifecycle-compiler:$lifecycle_version”

Below is the complete code for the **build.gradle(:app)** file.

KOTLIN:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'

android {
    compileSdkVersion 29
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.grocerylist"
        minSdkVersion 16
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles  getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }
    kotlinOptions {
```

```
        jvmTarget = "1.8"
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-
jdk7:$kotlin_version"
    implementation  'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.core:core-ktx:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation  'androidx.test.espresso:espresso-
core:3.2.0'

    def room_version = "2.2.1"
    def lifecycle_version = "2.0.0"

    // Room and Architectural Components
    implementation "androidx.room:room-runtime:$room_version"
    implementation "androidx.legacy:legacy-support-v4:1.0.0"
    implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.1.0'
    implementation "androidx.room:room-ktx:2.2.1"
    kapt  "androidx.room:room-compiler:$room_version"

    // Coroutines
    implementation  'org.jetbrains.kotlinx:kotlinx-coroutines-
core:1.3.0'
    implementation  "org.jetbrains.kotlinx:kotlinx-coroutines-
android:1.3.0"

    // New Material Design
    implementation  "com.google.android.material:material:1.0.0"

    // ViewModel
    implementation  "androidx.lifecycle:lifecycle-
extensions:$lifecycle_version"
    implementation  "androidx.lifecycle:lifecycle-viewmodel-
```

```
ktx:$lifecycle_version"
    kapt  "androidx.lifecycle:lifecycle-compiler:$lifecycle_version"
```

Below is the code for the **strings.xml** file. Here we have added the necessary strings that we are going to use in our project.

XML:

```
<resources>
    <string name="app_name">Fresh Basket</string>

    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank
fragment</string>
    <string name="itemName">Banana</string>
    <string name="itemQuantity">35</string>
    <string name="itemPrice">250Rs</string>
    <string name="totalCost">20</string>
    <string name="totalCostTitle">Total Cost</string>
    <string name="title">Add Items to your cart</string>
    <string name="etItem">Item</string>
    <string name="etQuantity">Quantity</string>
    <string name="etPrice">Price</string>
    <string name="save">Save</string>
    <string name="cancel">Cancel</string>

</resources>
```

Step 3: Implement room database

a) Entities class

The entities class contains all the columns in the database and it should be annotated with @Entity(tablename = “Name of table”). Entity class is a data class. And @Column info annotation is used to enter column variable name and datatype. We will also add Primary Key for auto-increment. Go to **app > java > com.example.application-name.** Right-click on **com.example.application-name** go to new and create Kotlin file/class and name the file as **GroceryEntities**. See the code below to completely

understand and implement.

KOTLIN:

```
package
com.example.grocerylist.Database.Entity

import  androidx.room.ColumnInfo
import androidx.room.Entity
import  androidx.room.PrimaryKey

// This is a data class which store data.
// Entities class create a table in database,
// in our database we will create three column

@Entity(tableName = "grocery_items")

data class GroceryItems(

    // create itemName variable to
    // store grocery items.
    @ColumnInfo(name = "itemName")
    var itemName: String,

    // create itemQuantity variable
    // to store grocery quantity.
    @ColumnInfo(name = "itemQuantity")
    var itemQuantity: Int,

    // create itemPrice variable to
    // store grocery price.
    @ColumnInfo(name = "itemPrice")
    var itemPrice: Int
) {
    // Primary key is a unique key
    // for different database.
    @PrimaryKey(autoGenerate = true)
```

```
    var id: Int? = null
}
```

b) Dao Interface

The Dao is an interface in which we create all the functions that we want to implement on the database. This interface also annotated with @Dao. Now we will create a function using suspend function which is a coroutines function. Here we create three functions, First is the insert function to insert items in the database and annotated with @Insert, Second is for deleting

items from the database annotated with @Delete and Third is for getting all items annotated with @Query. Go to the **app > java > com.example.application-name** . Right-click on **com.example.application-name** go to new and create Kotlin file/class and name the file as **GroceryDao**. See the code below to implement.

KOTLIN:

```
package com.example.grocerylist.Database

import androidx.lifecycle.LiveData
import androidx.room.*
import
com.example.grocerylist.Database.Entity.GroceryItems

// This class is used to create
// function for database.
@Dao
interface GroceryDao {

    // Insert function is used to
    // insert data in database.
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(item: GroceryItems)

    // Delete function is used to
    // delete data in database.
    @Delete
    suspend fun delete(item: GroceryItems)

    // getAllGroceryItems function is used to get
    // all the data of database.
    @Query("SELECT * FROM grocery_items")
    fun getAllGroceryItems():
LiveData<List<GroceryItems>>
}
```

c) Database class

Database class annotated with @Database(entities = [Name of Entity class.class], version = 1) these entities are the entities array list all the data entities associating with the database and version shows the current version of the database. This database class inherits from the Room Database class. In **GroceryDatabase** class we will make an abstract method to get an instance of DAO and further use this method from the DAO instance to interact with the database. See the below code to implement. Go to the **app > java > com.example.application-name**. Right-click on **com.example.application-name** go to new and create Kotlin file/class as **GroceryDatabase**. See the code below to implement.

KOTLIN:

```
package  com.example.grocerylist.Database

import  android.content.Context
import androidx.room.Database
import androidx.room.Room
import  androidx.room.RoomDatabase
import  com.example.grocerylist.Database.Entity.GroceryItems

@Database(entities = [GroceryItems::class], version = 1)
abstract class GroceryDatabase : RoomDatabase() {

    abstract fun getGroceryDao(): GroceryDao

    companion object {
        @Volatile
        private var instance: GroceryDatabase? = null
        private val LOCK = Any()

        operator fun invoke(context: Context) = instance ?: synchr
            instance ?: createDatabase(context).also {
                instance = it
            }
        }

        private fun createDatabase(context: Context) =
            Room.databaseBuilder(context.applicationContext,  Groce
"GroceryDatabase.db").build()
    }
}
```

Step 4: Now we will implement the architectural structure in the app

a) Repository class

The repository is one of the design structures. The repository class gives the data to the ViewModel class and then the ViewModel class uses that data for Views. The repository will choose the appropriate data locally or on the

network. Here in our Grocery Repository class data fetch locally from the Room database. We will add constructor value by creating an instance of the database and stored in the db variable in the Grocery Repository class. Go to the **app > java > com.example.application-name**. Right-click on **com.example.application-name** go to new and create Kotlin file/class as **GroceryRepository**. See the code below to implement.

KOTLIN:

```
package com.example.grocerylist.Database

import com.example.grocerylist.Database.Entity.GroceryItems

class GroceryRepository(private val db: GroceryDatabase) {

    suspend fun insert(item: GroceryItems) =
db.getGroceryDao().insert(item)
    suspend fun delete(item: GroceryItems) =
db.getGroceryDao().delete(item)

    fun allGroceryItems() = db.getGroceryDao().getAllGroceryItems(
}
```

Go to **app > java > com.example.application-name**. Right-click on **com.example.application-name** go to new and create a new Package called **UI** and then right-click on UI package and create a Kotlin file/class. See the code below to implement.

b) ViewModel class

ViewModel class used as an interface between View and Data. Grocery View Model class inherit from View Model class and we will pass constructor value by creating instance variable of Repository class and stored in repository variable. As we pass the constructor in View Model we have to create another class which is a Factory View Model class. Go to **app > java > com.example.application-name > UI**. Right-click on the UI package and create a Kotlin file/class and name the file as **GroceryViewModel**. See the

below code.

KOTLIN:

```
package com.example.grocerylist.UI

import androidx.lifecycle.ViewModel
import com.example.grocerylist.Database.Entity.GroceryItems
import com.example.grocerylist.Database.GroceryRepository
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch

class GroceryViewModel(private val repository: GroceryRepository)
ViewModel() {

    // In coroutines thread insert item in insert function.
    fun insert(item: GroceryItems) = GlobalScope.launch {
        repository.insert(item)
    }

    // In coroutines thread delete item in delete function.
    fun delete(item: GroceryItems) = GlobalScope.launch {
        repository.delete(item)
    }

    //Here we initialized allGroceryItems function with repository
    fun allGroceryItems() = repository.allGroceryItems()

}
```

c) Factory ViewModel class

We will inherit the Grocery ViewModel Factory class from ViewModelProvider.NewInstanceFactory and again pass constructor value by creating instance variable of Grocery Repository and return GroceryViewModel(repository). Go to the **app > java > com.example.application-name > UI**. Right-click on the **UI** package and

create a Kotlin file/class name it **GroceryViewModelFactory**. See the below code to understand.

KOTLIN:

```
package com.example.grocerylist.UI

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.grocerylist.Database.GroceryRepository

class GroceryViewModelFactory(private val repository:
GroceryRepository):ViewModelProvider.NewInstanceFactory() {

    override fun <T : ViewModel?> create(modelClass: Class<T>): T
        return GroceryViewModel(repository) as T
    }
}
```

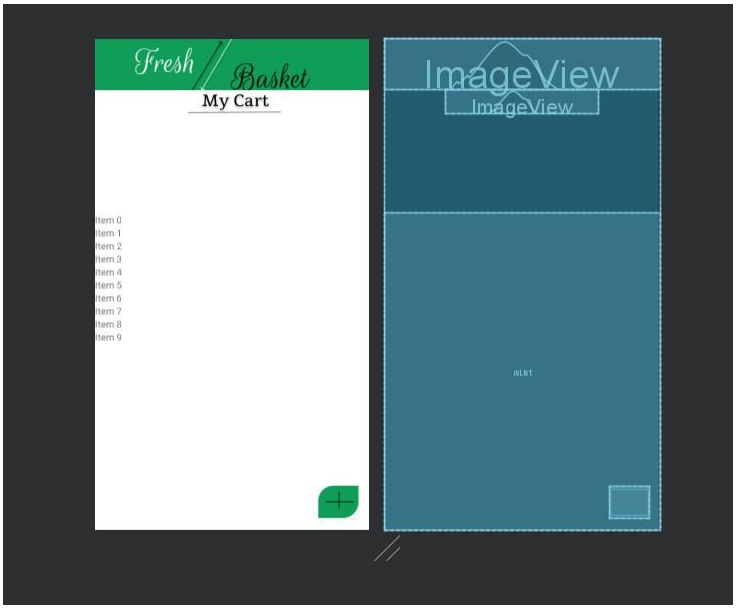
Step 5: Now let's jump into the UI part

In the **activity_main.xml** file, we will add two ImageView, RecyclerView, and Button after clicking this button a **DialogBox** open and in that dialog box user can enter the item name, item quantity, and item price. Refer to the following code.

XML:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical"
    tools:context=".UI.MainActivity">

    <!-- To create a app bar with logo image. -->
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/logo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <!-- In this image view we will add a title image -->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:src="@drawable/title"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/imageView2"
        app:layout_constraintVertical_bias="0.0" />

    <!-- Recycler View to display list -->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvList"
        android:layout_width="match_parent"
```

Output of XML Code:



Step 6: Let's implement **RecyclerView**. Now we will code the UI part of the row in the list. Go to **app > res > layout**. Right-click on layout, go to new, and then add a **Layout Resource File** and name it as **groceryadapter**. See the XML code of the **groceryadapter.xml** file.

XML:

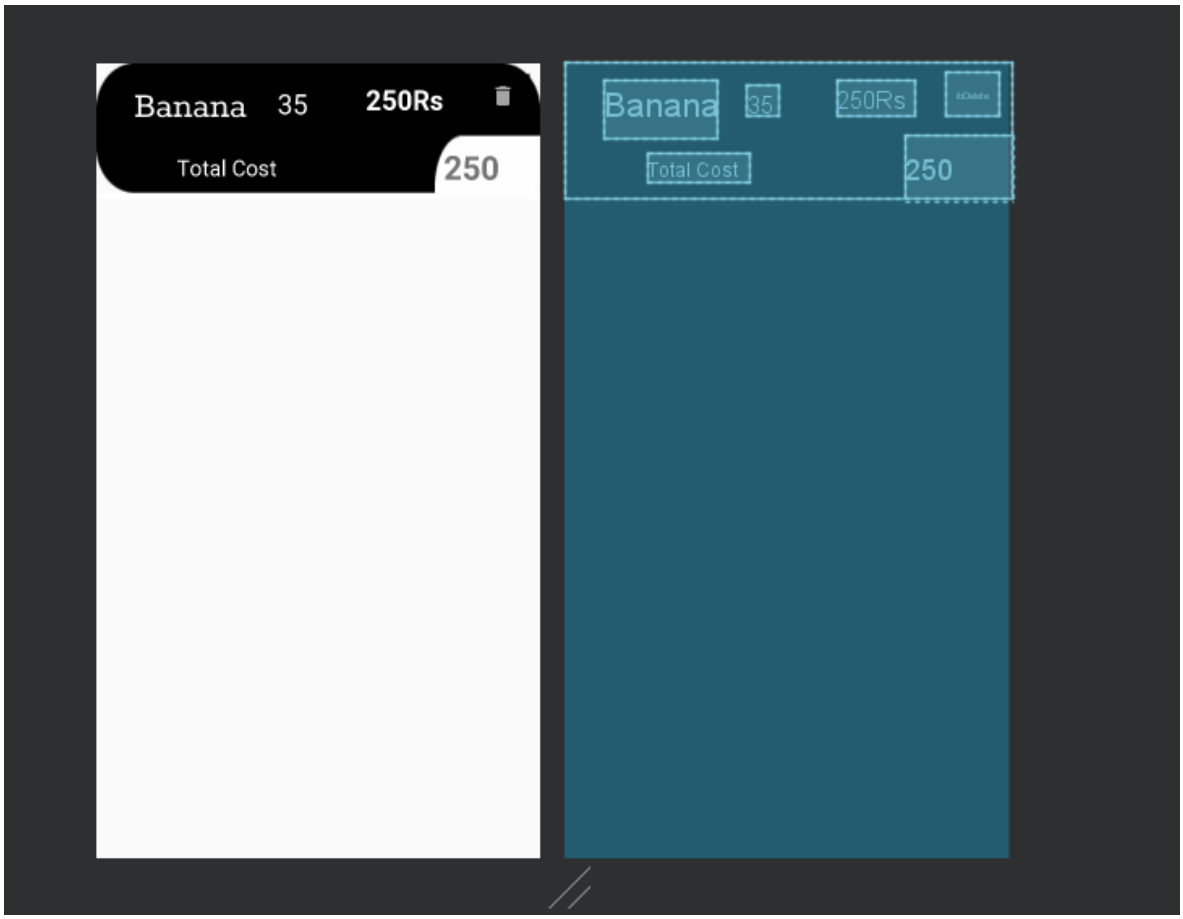
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="125dp"
    android:background="@drawable/adapter1">

    <!-- To display item name in recycler view -->
    <TextView
        android:id="@+id/txtItemName"
        android:layout_width="0dp"
        android:layout_height="53dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="15dp"
        android:layout_marginRight="15dp"
```

```
        android:layout_marginBottom="17dp"
        android:fontFamily="@font/rokkitt"
        android:text="@string/itemName"
        android:textColor="@color/white"
        android:textSize="35sp"
        app:layout_constraintBottom_toTopOf="@+id/txtTotalCostTitle"
        app:layout_constraintEnd_toStartOf="@+id/txtItemQuantity"
        app:layout_constraintStart_toEndOf="@+id/cbItemCheck"
        app:layout_constraintTop_toTopOf="parent" />

    <!-- To display item quantity -->
    <TextView
        android:id="@+id/txtItemQuantity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="34dp"
        android:layout_marginRight="34dp"
        android:layout_marginBottom="9dp"
        android:text="@string/itemQuantity"
        android:textColor="@color/white"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="@+id/txtItemName"
        app:layout_constraintEnd_toStartOf="@+id/txtItemPrice"
        app:layout_constraintStart_toEndOf="@+id/txtItemName"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <!-- To display item price -->
    <TextView
        android:id="@+id/txtItemPrice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="26dp"
        android:layout_marginEnd="22dp"
        android:layout_marginRight="22dp"
        android:layout_marginBottom="26dp"
        android:text="@string/itemPrice"
        android:textColor="@color/white"
        android:textSize="25sp"
        android:textStyle="bold"
```

```
    app:layout_constraintBottom_toTopOf="@+id/txtItemTotalCost"
    app:layout_constraintEnd_toStartOf="@+id/ibDelete"
    app:layout_constraintStart_toEndOf="@+id/txtItemQuantity"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0" />

<CheckBox
    android:id="@+id/cbItemCheck"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="29dp"
    android:layout_marginLeft="29dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:background="@color/white"
    android:shadowColor="@color/black"
    app:layout_constraintBaseline_toBaselineOf="@+id/txtItemName"
    app:layout_constraintEnd_toStartOf="@+id/txtItemName"
    app:layout_constraintStart_toStartOf="parent" />

<!-- This button is used to delete grocery item  -->
<ImageButton
    android:id="@+id/ibDelete"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="26dp"
    android:layout_marginRight="26dp"
    android:background="@color/black"
    android:src="@drawable/ic_action_delete"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/txtItemPrice"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.257" />

<!-- To display total cost of grocery items -->
<TextView
    android:id="@+id/txtItemTotalCost"
    android:layout_width="100dp"
    android:layout_height="60dp"
```

```
        android:background="@drawable/adapter2"
        android:padding="8dp"
        android:paddingLeft="12dp"
        android:text="@string/totalCost"
        android:textSize="30sp"
        android:textStyle="bold"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/txtTotalCostTitle"
        app:layout_constraintTop_toTopOf="@+id/txtTotalCostTitle"
        app:layout_constraintVertical_bias="0.875" />

    <!-- This text view is used to add statement for total cost  -->
    <TextView
        android:id="@+id/txtTotalCostTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="77dp"
        android:layout_marginLeft="77dp"
        android:layout_marginEnd="149dp"
        android:layout_marginRight="149dp"
        android:layout_marginBottom="16dp"
        android:text="@string/totalCostTitle"
        android:textColor="@color/white"
        android:textSize="20dp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/txtItemTotalCost"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

OUTPUT OF XML CODE:



We will code adapter class for recycler view. In the Grocery Adapter class, we will add constructor value by storing entities class as a list in list variable and create an instance of the view model. In Grocery Adapter we will override three functions: onCreateViewHolder, getItemCount, and onbindViewHolder, we will also create an inner class called grocery view holder. Go to the **app > java > com.example.application-name**. Right-click on **com.example.application-name** go to new and create a new Package called **Adapter** and then right-click on Adapter package and create a Kotlin file/class name it **GroceryAdapter**. See the below code.

KOTLIN:

```
package com.example.grocerylist.Adapter
 import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.grocerylist.Database.Entity.GroceryItems
import com.example.grocerylist.R
import com.example.grocerylist.UI.GroceryViewModel
import kotlinx.android.synthetic.main.groceryadapter.view.*

class GroceryAdapter(var list: List<GroceryItems>, val viewModel:
GroceryViewModel) :
    RecyclerView.Adapter<GroceryAdapter.GroceryViewHolder>() {

    // In this function we will add our groceryadapter.xml to kotlin
class
    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): GroceryViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.groceryadapter,
parent, false)
        return GroceryViewHolder(view)
    }

    // This function is used to return total number of size of list.
    override fun getItemCount(): Int {
        return list.size
```

```
    }

    // In onBindViewHolder we will bind our itemViews with adapter
    override fun onBindViewHolder(holder: GroceryViewHolder,
position: Int) {
        var currentPosition = list[position]
        holder.itemView.txtItemName.text = currentPosition.itemName
        holder.itemView.txtItemPrice.text =
"${currentPosition.itemPrice}"
        holder.itemView.txtItemQuantity.text =
"${currentPosition.itemQuantity}"
        holder.itemView.ibDelete.setOnClickListener {
            viewModel.delete(currentPosition)
        }

        // To get total cost
        if (position == list.size - 1) {
            var totalCost = 0
            for (i in 0 until list.size) {
                totalCost += list[i].itemPrice
            }
            holder.itemView.txtItemTotalCost.visibility =
View.VISIBLE
            holder.itemView.txtTotalCostTitle.visibility =
View.VISIBLE
            holder.itemView.txtItemTotalCost.text = "$totalCost"
        }
    }

    // Inner class for viewHolder
    inner class GroceryViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView)
}
```

Step 7: To enter grocery item, quantity, and price from the user we have to create an interface. To implement this interface we will use DialogBox. First create UI of dialog box. In this dialog box we will add three edit text and two text view. Three edit text to enter grocery item name, quantity and price. Two text view one for save and other for cancel. After clicking the save text all data saved into the database and by clicking on the cancel text dialog box closes. Go to the **app > res > layout**. Right-click on **layout**, go to new and then add a **Layout**

Resource File and name it as **grocerydialog**. See xml code of **grocerydialog.xml** file.

XML:

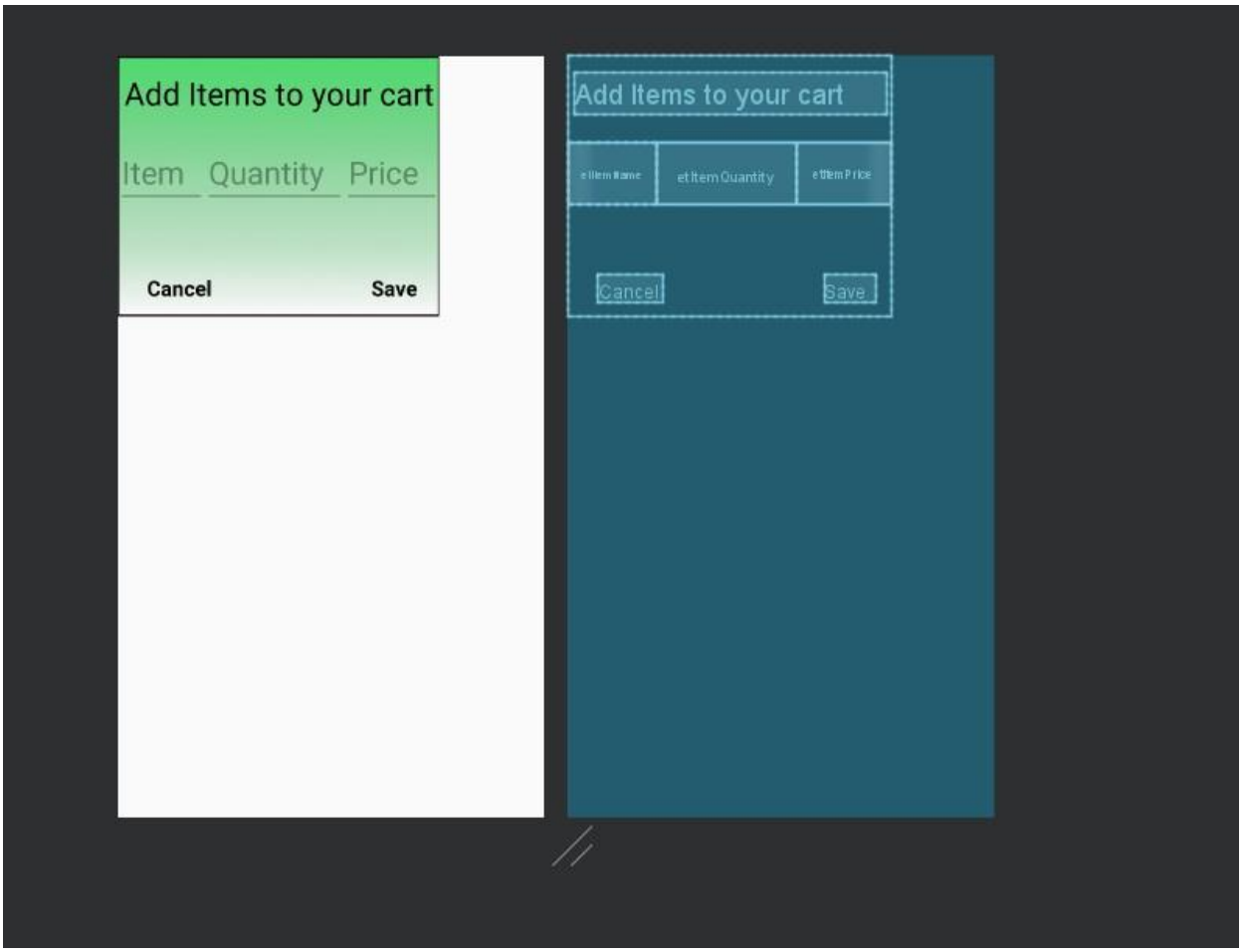
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="310dp"
    android:layout_height="250dp"
    android:background="@drawable/rectangle">

    <!-- To display title-->
    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:layout_marginEnd="5dp"
        android:layout_marginRight="5dp"
        android:layout_marginBottom="26dp"
        android:text="@string/title"
        android:textColor="@color/black"
        android:textSize="30dp"
        app:layout_constraintBottom_toTopOf="@+id/linearLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <!-- Linear Layout is used to give equal
         weight sum to edit text-->
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="66dp"
        android:weightSum="3"
        app:layout_constraintBottom_toTopOf="@+id/tvSave"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvTitle">
```

```
<!-- Edit Text is used to Enter Grocery
     Item Name by user-->
<EditText
    android:id="@+id/etItemName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="0dp"
    android:layout_weight="1"
    android:hint="@string/etItem"
    android:textSize="30dp"
    app:layout_constraintBottom_toTopOf="@+id/tvCancel"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.069"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.661" />

<!-- Edit Text is used to Enter Grocery
     Item Quantity by user-->
<EditText
    android:id="@+id/etItemQuantity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="0dp"
    android:layout_weight="1"
    android:hint="@string/etQuantity"
    android:inputType="number"
    android:textSize="30dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/etItemPrice"
    app:layout_constraintHorizontal_bias="0.461"
    app:layout_constraintStart_toEndOf="@+id/etItemName"
    app:layout_constraintTop_toBottomOf="@+id/tvTitle"
    app:layout_constraintVertical_bias="0.276" />

<!-- Edit Text is used to Enter Grocery
     Item Price by user-->
<EditText
    android:id="@+id/etItemPrice"
    android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:layout_weight="1"
            android:hint="@string/etPrice"
            android:inputType="number"
            android:textSize="30dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.861"
            app:layout_constraintStart_toEndOf="@+id/etItemName"
            app:layout_constraintTop_toTopOf="parent" />

    </LinearLayout>

    <!-- Text view is used as save button to save
         all details in database by user-->
    <TextView
        android:id="@+id/tvSave"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="14dp"
        android:text="@string/save"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tvCancel"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout"  />

    <!-- Text View is used to close dialog box-->
    <TextView
        android:id="@+id/tvCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginEnd="154dp"
```

```
        android:layout_marginRight="154dp"
        android:layout_marginBottom="14dp"
        android:text="@string/cancel"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/tvSave"
        app:layout_constraintStart_toStartOf="parent"  />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Output of XML Code:



To add a clicklistener on save text we have to create an interface first in which we create a function. Go to the **app > java > com.example.application-name > UI.** Right-click on the **UI** package and create a Kotlin file/class and create an **interface** name it as **DialogListener**. See the code of the **DialogListener.kt**

file.

KOTLIN:

```
package com.example.grocerylist.UI
 import com.example.grocerylist.Database.Entity.GroceryItems
 interface DialogListener {
    // Create a function to add items
      // in GroceryItems on clicking
    fun onAddButtonClicked(item:GroceryItems)
}
```

Now we will create grocery dialog class in which we save all input in different variable and then insert in database. Go to the **app > java > com.example.application-name > UI.** Right-click on **UI** package and create a Kotlin file/class and create an class name it **GroceryItemDialog**. See the below code.

KOTLIN:

```
package com.example.grocerylist.UI
 import android.content.Context
import android.os.Bundle
import android.view.Window
import android.widget.Toast
import androidx.appcompat.app.AppCompatDialog
import com.example.grocerylist.Database.Entity.GroceryItems
import com.example.grocerylist.R
import kotlinx.android.synthetic.main.grocerydialog.*
 class GroceryItemDialog(context: Context, var dialogListener:
DialogListener) : AppCompatDialog(context) {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        supportRequestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.grocerydialog)
  // Click listener on Save button
        // to save all data.
        tvSave.setOnClickListener {
  // Take all three inputs in different variables from user
            // and add it in Grocery Items database
            val name = etItemName.text.toString()
            val quantity = etItemQuantity.text.toString().toInt()
            val price = etItemPrice.text.toString().toInt()
```

```
  // Toast to display enter items in edit text
            if (name.isEmpty()) {
                Toast.makeText(context, "Please Enter Item Name",
Toast.LENGTH_SHORT).show()
            }
   val item = GroceryItems(name, quantity, price)
            dialogListener.onAddButtonClicked(item)
            dismiss()
        }

 // On click listener on cancel text to close dialog box
             tvCancel.setOnClickListener {
            cancel()
        }
    }
}
```

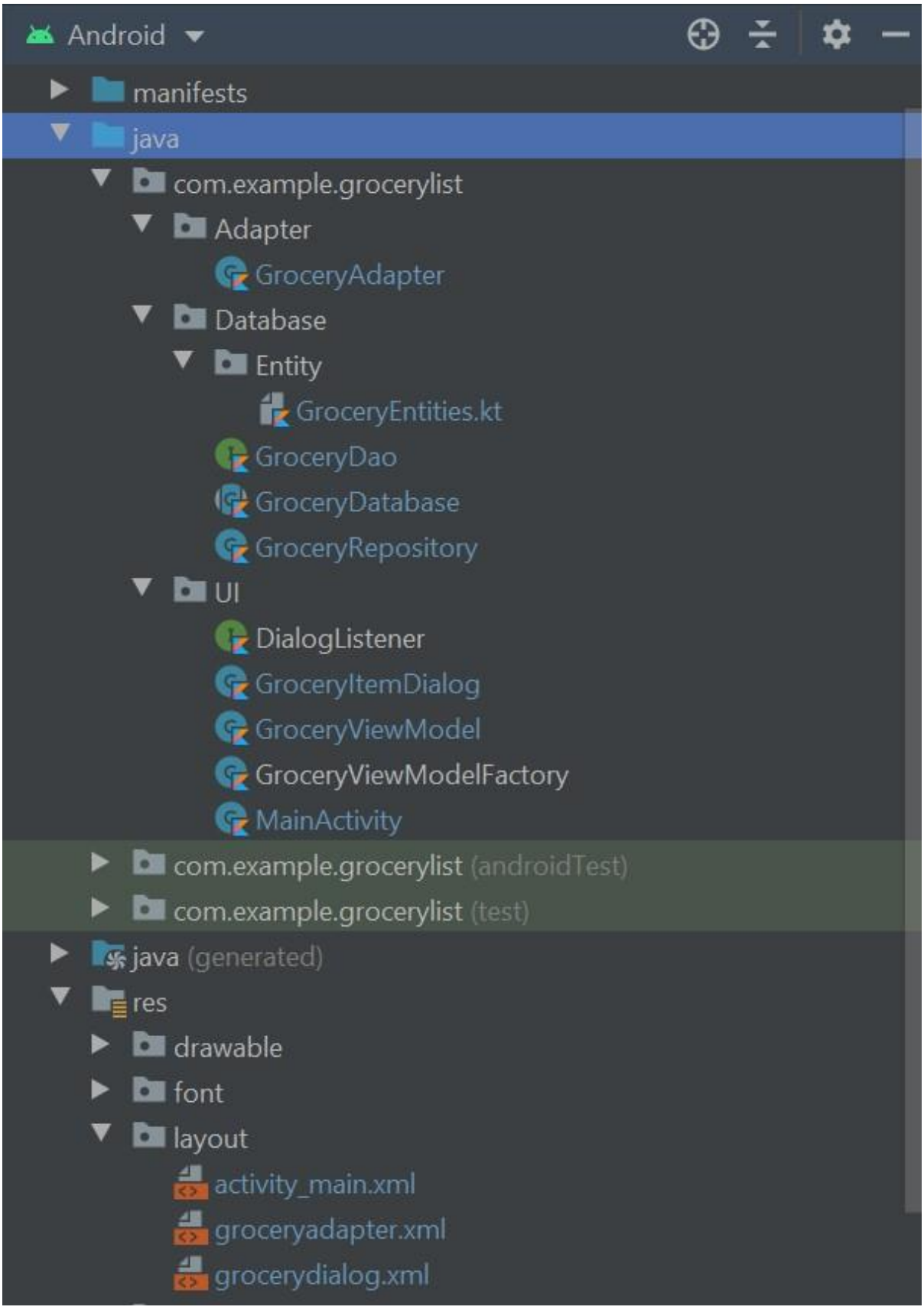
Step 8: In this step finally we will code in our MainActivity. In our main activity, we have to set up the recycler view and add click listener on add button to open the dialog box. Go to the **MainActivity.kt** file and refer to the following code. Below is the code for the **MainActivity.kt** file. Comments are added inside the code to understand the code in more detail.

KOTLIN:

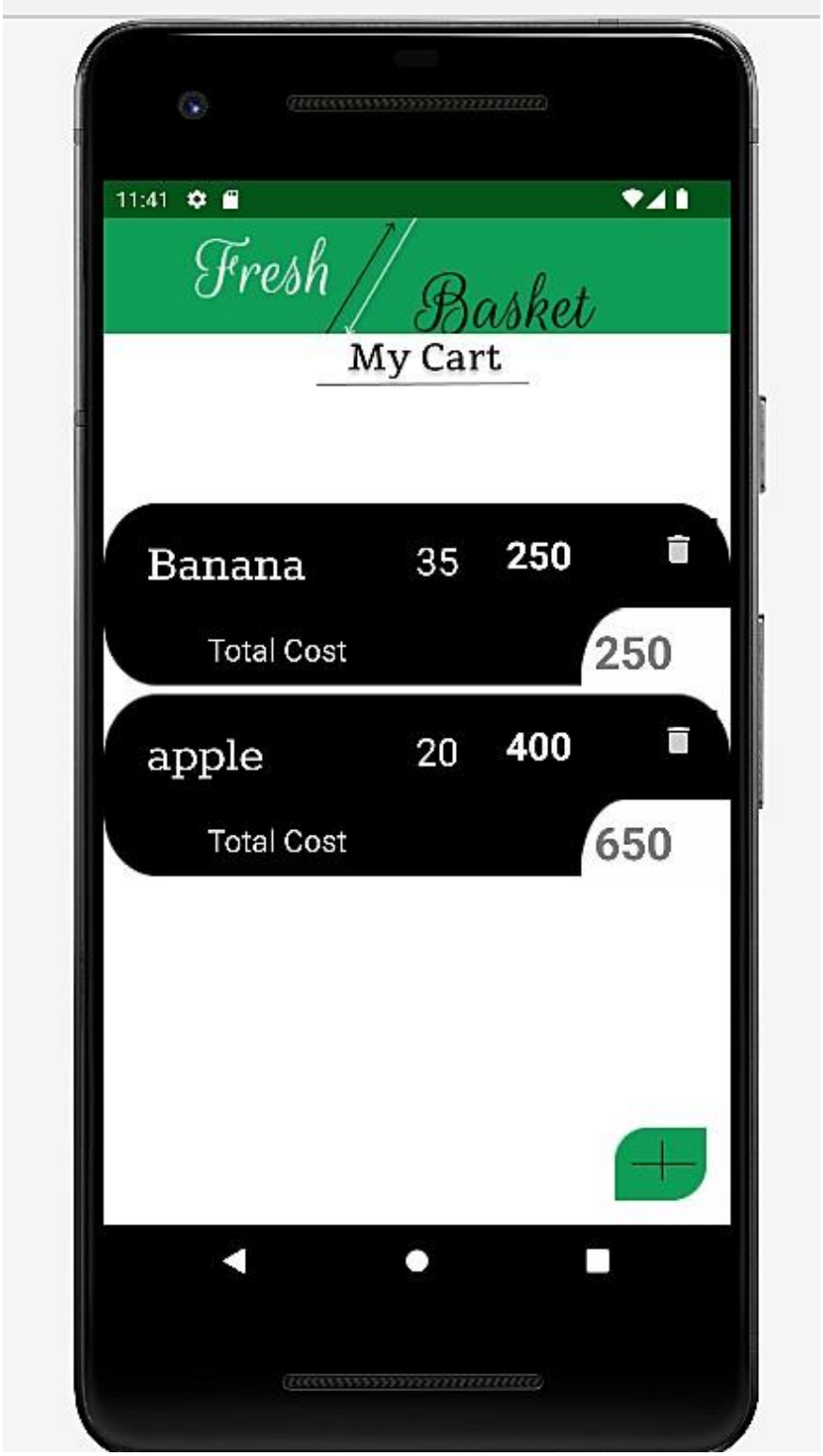
```
package com.example.grocerylist.UI

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.grocerylist.Adapter.GroceryAdapter
import com.example.grocerylist.Database.Entity.GroceryItems
import com.example.grocerylist.Database.GroceryDatabase
import com.example.grocerylist.Database.GroceryRepository
import com.example.grocerylist.R
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
```

```
    lateinit var ViewModel: GroceryViewModel
    lateinit var list: List<GroceryItems>

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

        val groceryRepository =
GroceryRepository(GroceryDatabase(this))
        val factory = GroceryViewModelFactory(groceryRepository)

        // Initialised View Model
        ViewModel = ViewModelProvider(this,
factory).get(GroceryViewModel::class.java)
        val groceryAdapter = GroceryAdapter(listOf(), ViewModel)
        rvList.layoutManager = LinearLayoutManager(this)
        rvList.adapter = groceryAdapter

        // To display all items in recycler view
        ViewModel.allGroceryItems().observe(this, Observer {
            groceryAdapter.list = it
            groceryAdapter.notifyDataSetChanged()
        })

        // on ClickListener on button to open dialog box
        btnAdd.setOnClickListener {
            GroceryItemDialog(this, object : DialogListener {
                override fun onAddButtonClicked(item: GroceryItems) {
                    ViewModel.insert(item)
                }
            }).show()
        }
    }
}
```

This is how the complete project structure looks like.


Android
manifests
java
com.example.grocerylist
Adapter
GroceryAdapter
Database
Entity
GroceryEntities.kt
GroceryDao
GroceryDatabase
GroceryRepository
UI
DialogListener
GroceryItemDialog
GroceryViewModel
GroceryViewModelFactory
MainActivity
com.example.grocerylist (androidTest)
com.example.grocerylist (test)
java (generated)
res
drawable
font
layout
activity_main.xml
groceryadapter.xml
grocerydialog.xml

OUTPUT:


11:41
Fresh Basket
My Cart
Banana 35 250
Total Cost 250
apple 20 400
Total Cost 650