GROCERY ANDROID APP

1. INTRODUCTION

Many times we forget to purchase things that we want to buy, after all, we can’t remember all the items. However, with the assistance of this app, you can make a list of the groceries you intend to buy so that you don't forget anything. The project is implemented using the Kotlin language.

1.1 OVERVIEW

In this project, we are using MVVM (Model View ViewModel) for architectural patterns, Room for database, Co-routines, and RecyclerView to display the list of items.

MVVM (Model View ViewModel)
MVVM architecture in android is used to structure the project’s code and is also easy to understand. MVVM is an architectural design pattern in android. MVVM treats Activity classes and XML files as View. This design pattern completely separates UI from its logic.

Room for database
Room persistence library is a database management library and is used to store the data of apps like grocery item name, grocery item quantity, and grocery item price. Room is a cover layer on SQLite, which helps perform the database operation.

Co-routines
Co-routines are lightweight threads. It is used to operate on other threads. By using co-routines the main thread doesn’t block and also app doesn’t crash.

RecyclerView
RecyclerView is a container used to display the collection of data in a large amount of data set that can be scrolled very effectively by maintaining a limited number of views.

1.2 PURPOSE

The purpose of this project is to create a Grocery Android App using Kotlin. Many times we forget to purchase things that we want to buy, after all, we can’t remember all the items. However, with the assistance of this app, you can make a list of the groceries you intend to buy so that you don't forget anything. The Item name, Item quantity, and price can be added and removed according to the user.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Some of the existing approaches only allow users to add item names and quantities. In some apps, only the item name can be added and removed. Sometimes you might have to order your products by weight, then it can create a problem if the app doesn’t support this cause. In other cases, the UI is complicated and the user finds it hard to use the app.

2.2 PROPOSED SOLUTION

The proposed solution is a simple and user-friendly grocery list app. It takes the mundane task of listing your groceries and adds a little flair. The app lets you quickly add and delete items and gives the total amount for the respective items. The proposed app takes in item names with their respective quantity and prices. The user can easily navigate between options and is likely to use the app again.

In this project, we used MVVM (Model View ViewModel) for architectural patterns, Room for database, Co-routines, and RecyclerView to display the list of items.

The proposed app uses MVVM. It has several benefits, including the separation of concerns. Typically, there is a connection between the user interface and application logic, resulting in change-resistant, brittle code. MVVM cleanly separates the user interface from the application logic.

Room is now considered a better approach for data persistence than SQLiteDatabase. It makes it easier to work with SQLiteDatabase objects in your app, decreasing the amount of boilerplate code and verifying SQL queries at compile time. It has Compile-time verification of SQL queries. Each @Query and @Entity is checked at the compile time, which preserves your app from crash issues at runtime not only it checks the only syntax, but also missing tables and is easily integrated with other Architecture components (like LiveData)

Coroutines provide us with an easy way to do synchronous and asynchronous programming. Coroutines allow execution to be suspended and resumed later at some point in the future which is best suited for performing non-blocking operations in the case of multithreading.

The major advantage of the usage of RecyclerView is the performance when loading list items in the list to the view. RecyclerView prepares the view behind and ahead beyond the visible entries. It gives significant performance when you need to fetch the bitmap image in your list from a background task.

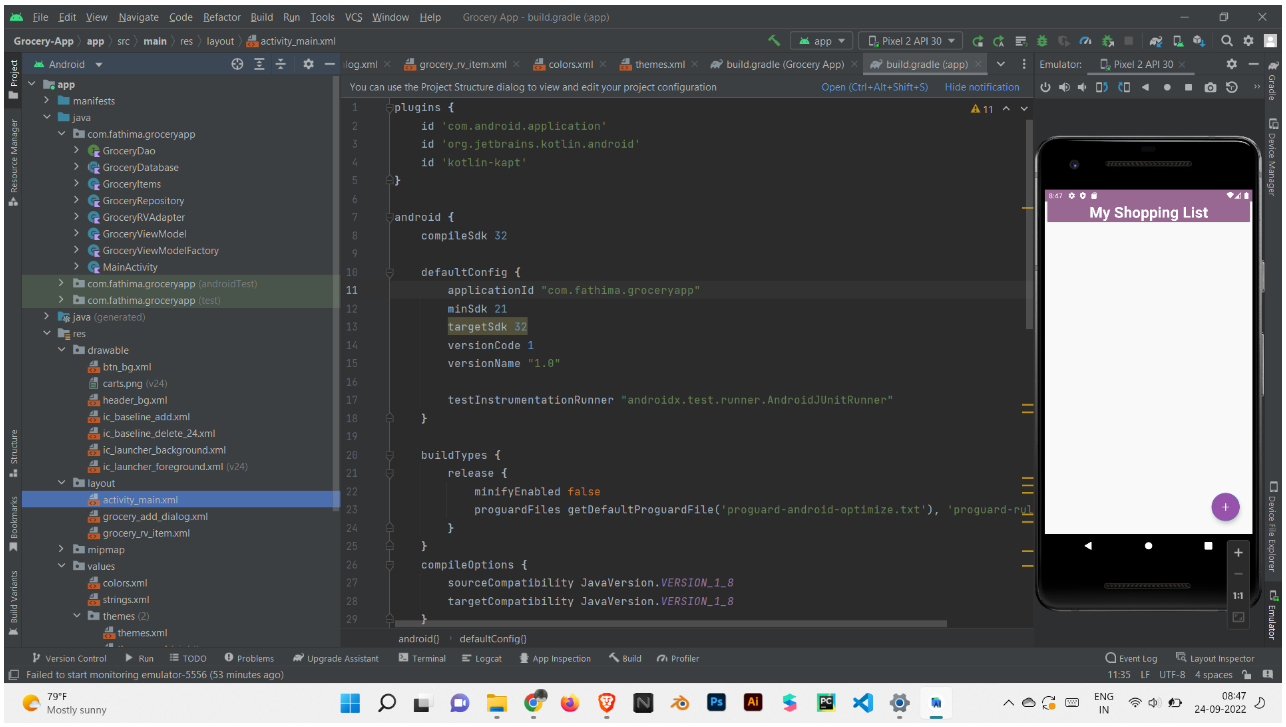
3. THEORETICAL ANALYSIS

The proposed app uses MVVM. It has several benefits, including the separation of concerns. Typically, there is a connection between the user interface and application logic, resulting in change-resistant, brittle code. MVVM cleanly separates the user interface from the application logic.

Room is now considered a better approach for data persistence than SQLiteDatabase. It makes it easier to work with SQLiteDatabase objects in your app, decreasing the amount of boilerplate code and verifying SQL queries at compile time.

Coroutines provide us with an easy way to do synchronous and asynchronous programming. Coroutines allow execution to be suspended and resumed later which is best suited for performing non-blocking operations in the case of multithreading.

RecyclerView prepares the view behind and ahead beyond the visible entries. It gives significant performance when you need to fetch the bitmap image in your list from a background task.



```
app
  manifests
  java
    com.fathima.groceryapp
      GroceryDao
      GroceryDatabase
      GroceryItems
      GroceryRepository
      GroceryRVAdapter
      GroceryViewModel
      GroceryViewModelFactory
      MainActivity
    com.fathima.groceryapp (androidTest)
    com.fathima.groceryapp (test)
  java (generated)
  res
    drawable
      btn_bg.xml
      carts.png (v24)
      header_bg.xml
      ic_baseline_add.xml
      ic_baseline_delete_24.xml
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
      activity_main.xml
      grocery_add_dialog.xml
      grocery_rv_item.xml
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
        themes.xml
```

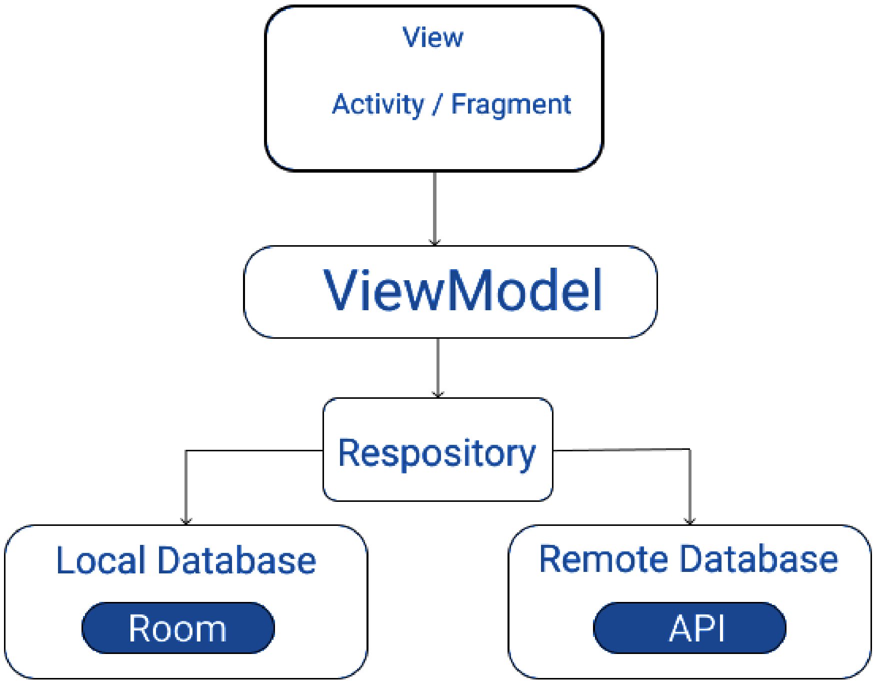
```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-kapt'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.fathima.groceryapp"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rul
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

Failed to start monitoring emulator-5556 (53 minutes ago)

Grocery App - build.gradle (:app)
You can use the Project Structure dialog to view and edit your project configuration
My Shopping List

The project file structure is as of above

3.1 BLOCK DIAGRAM



3.2 SOFTWARE/HARDWARE DESIGNING

3.2.1 SOFTWARE REQUIREMENTS

- Android Studio

3.2.1 HARDWARE REQUIREMENTS

- 64-bit Microsoft® Windows® 8/10.
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor.
- 8 GB RAM or more.
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution.

4. EXPERIMENTAL INVESTIGATION

- Entities class

The entities class contains all the columns in the database and it should be annotated with @Entity(tablename = "Name of table"). An 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.

- Dao Interface

The Dao is an interface in which we create all the functions that we want to implement on the database. This interface is 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 them with @Insert, the Second is for deleting items from the database annotated with @Delete and the Third is for getting all items annotated with @Query.

- 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 associated with the database, and the 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.

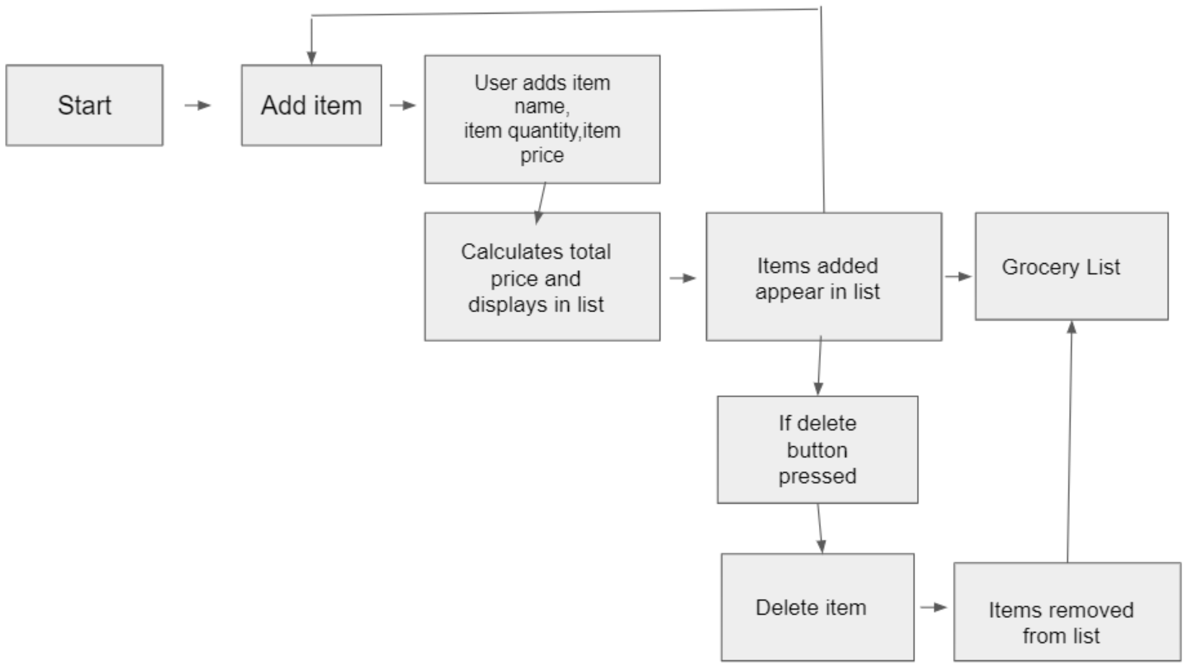
- Repository class

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.

- ViewModel class

ViewModel class is used as an interface between View and Data. Grocery View Model class inherits 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.

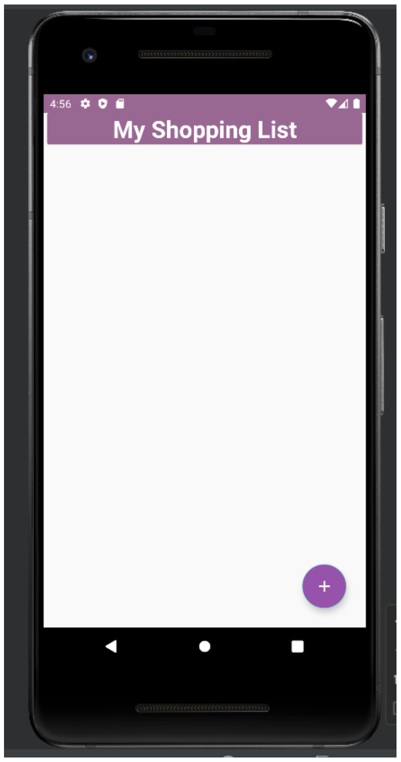
5. FLOWCHART

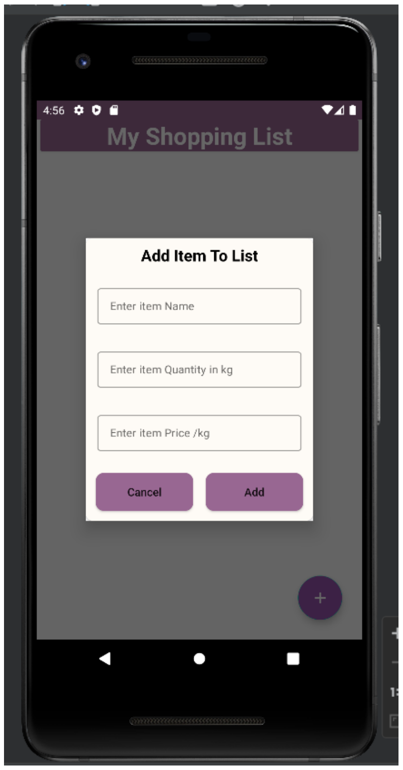

Start
Add item
User adds item name, item quantity,item price
Calculates total price and displays in list
Items added appear in list
Grocery List
If delete button pressed
Delete item
Items removed from list

6. RESULT

A Grocery List Android App using MVVM and Room Database in Kotlin is successfully developed. The Grocery list App is simple and user-friendly. In this project, we used MVVM (Model View ViewModel) for architectural patterns, Room for database, Co-routines, and RecyclerView to display the list of items.

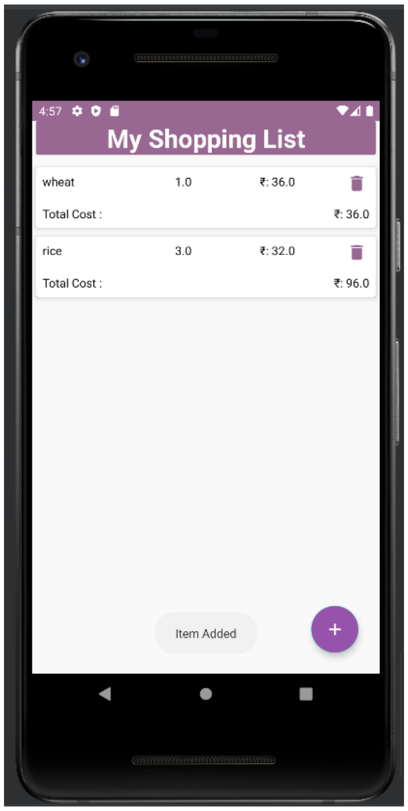
The screenshots of the Grocery List App project :





Grocery list app

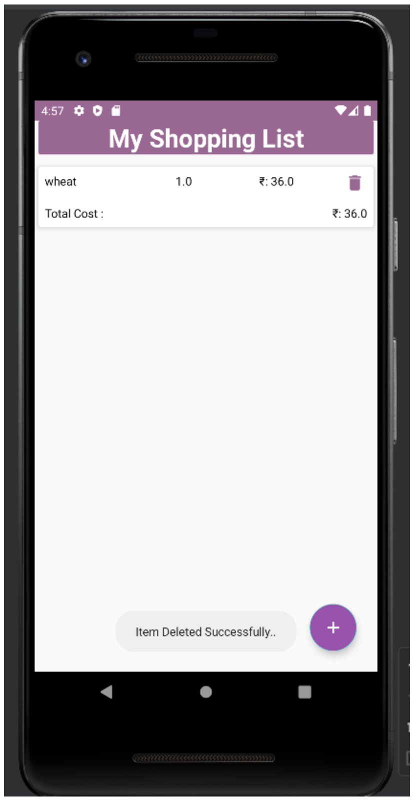
adding item to the list


4:57

My Shopping List

wheat	1.0	₹: 36.0	
Total Cost :			₹: 36.0
rice	3.0	₹: 32.0	
Total Cost :			₹: 96.0

Item Added


4:57

My Shopping List

wheat	1.0	₹: 36.0	
Total Cost :			₹: 36.0

Item Deleted Successfully..

Added items shown item deleted

7. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Enables Users To Track Spending
- Easy to use
- User Friendly
- Customized Grocery List

DISADVANTAGES

- Custom Categories not available
- Smart Suggestions are not available

8. APPLICATIONS

- A Grocery List app where multiple people can edit the shopping list at the same time. These lists can also be accessed online on a computer in real-time
- Reminders and notifications can be added to the Grocery List proposed
- Recipe Application where the items are added and a suitable recipe with the items is suggested

9. CONCLUSION

The proposed solution for the Grocery list App is simple and user-friendly. It takes the mundane task of listing your groceries and adds a little flair. In this project, we used MVVM (Model View ViewModel) for architectural patterns, Room for database, Co-routines, and RecyclerView to display the list of items. The proposed app takes in item names with their respective quantity and prices and gives the Total amount for the respective products. The user can easily navigate between options and is likely to use the app again.

10. FUTURE SCOPE

The following features can be added:

- Voice Input

  A grocery list app with voice functionality that can allow you to add, remove, or make a whole list using just your voice input.

- Custom Categories & Items

  Categorizing similar products and items can add value to the application

- Smart Suggestions

  This feature lets you welcome suggestions on the food items you add to your grocery shopping list. It will speed up your job of creating a grocery shopping list

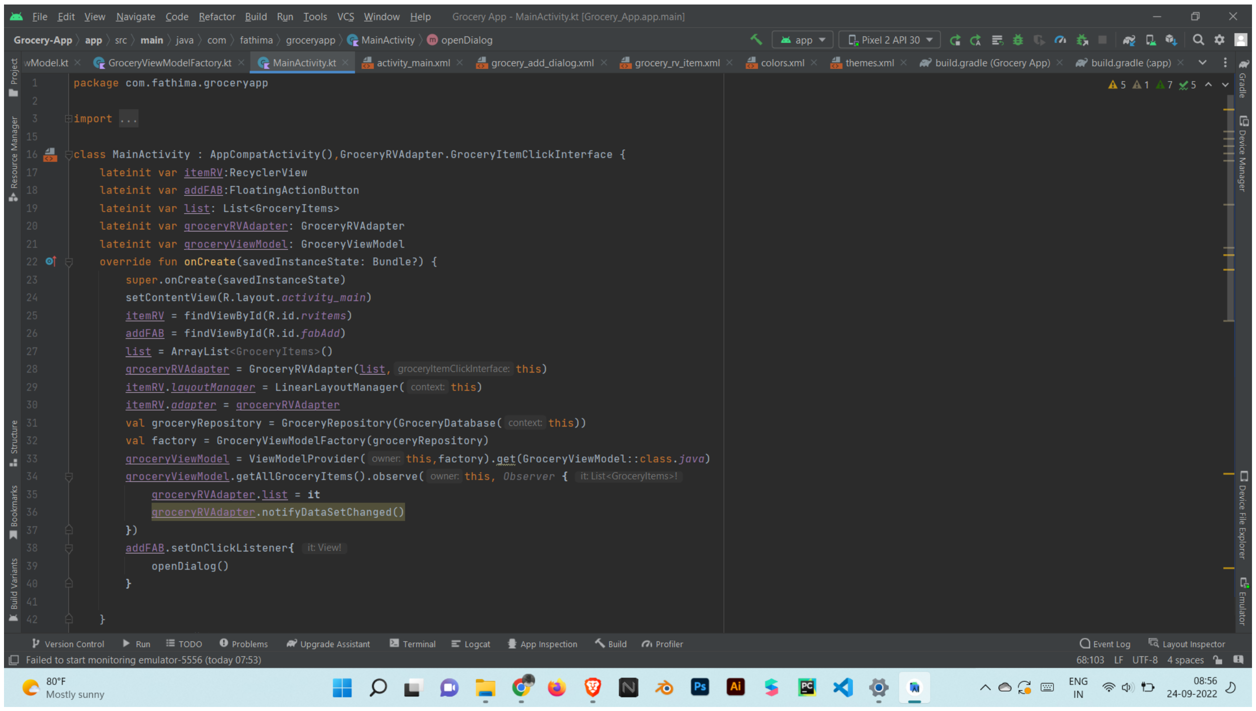
11. BIBLIOGRAPHY

- https://www.geeksforgeeks.org/how-to-build-a-grocery-android-app-using-mvvm-and-room-database/
- https://www.youtube.com/watch?v=vdcLb_Y71Ic
- https://levelup.gitconnected.com/android-recycler-view-b1bc493cac53
- https://www.xenonstack.com/insights/coroutines
- https://proandroiddev.com/kotlin-coroutines-in-andriod-ff0b3b399fa0
- https://medium.com/mindorks/using-room-database-android-jetpack-675a89a0e942
- https://www.sagitec.com/blog/the-5ws-of-mvvm#:~:text=Migrating%20to%20MVVM%20offers%20several,interface%20from%20the%20application%20logic.
- https://www.tutorialspoint.com/mvvm/mvvm_first_application.html
- https://www.emizentech.com/blog/how-to-make-a-grocery-list-app.html

APPENDIX

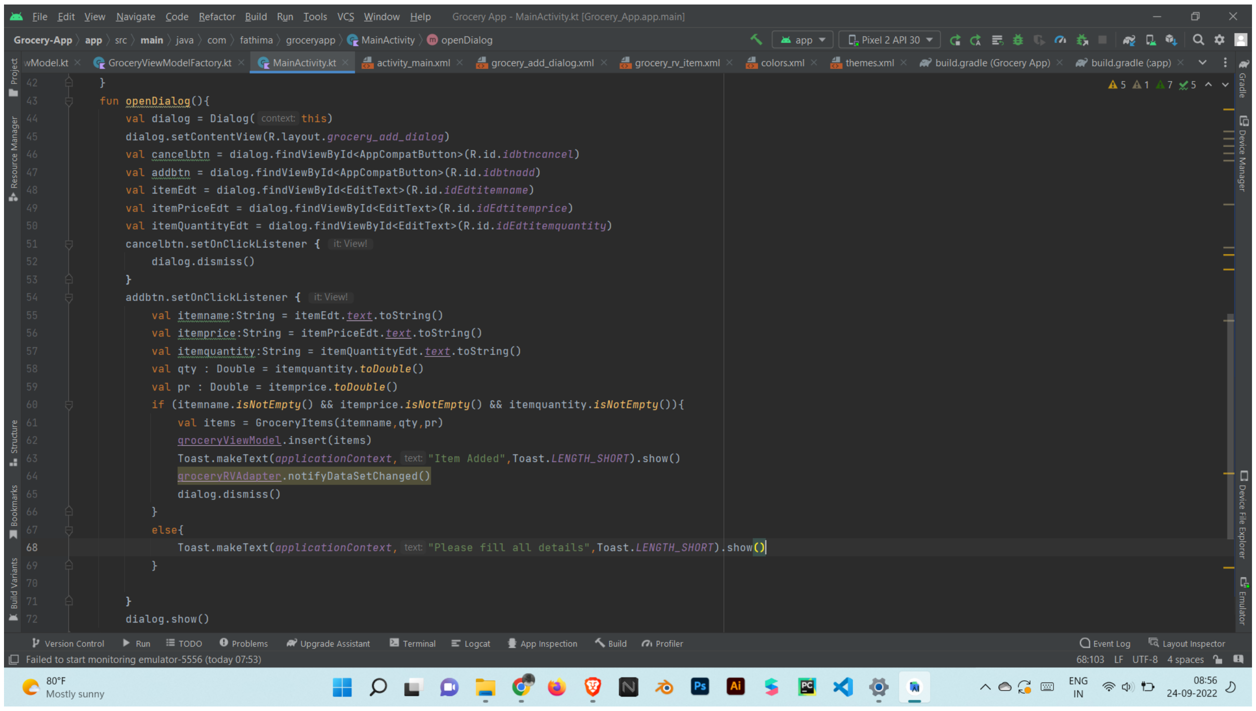
A. SOURCE CODE

- MainActivity



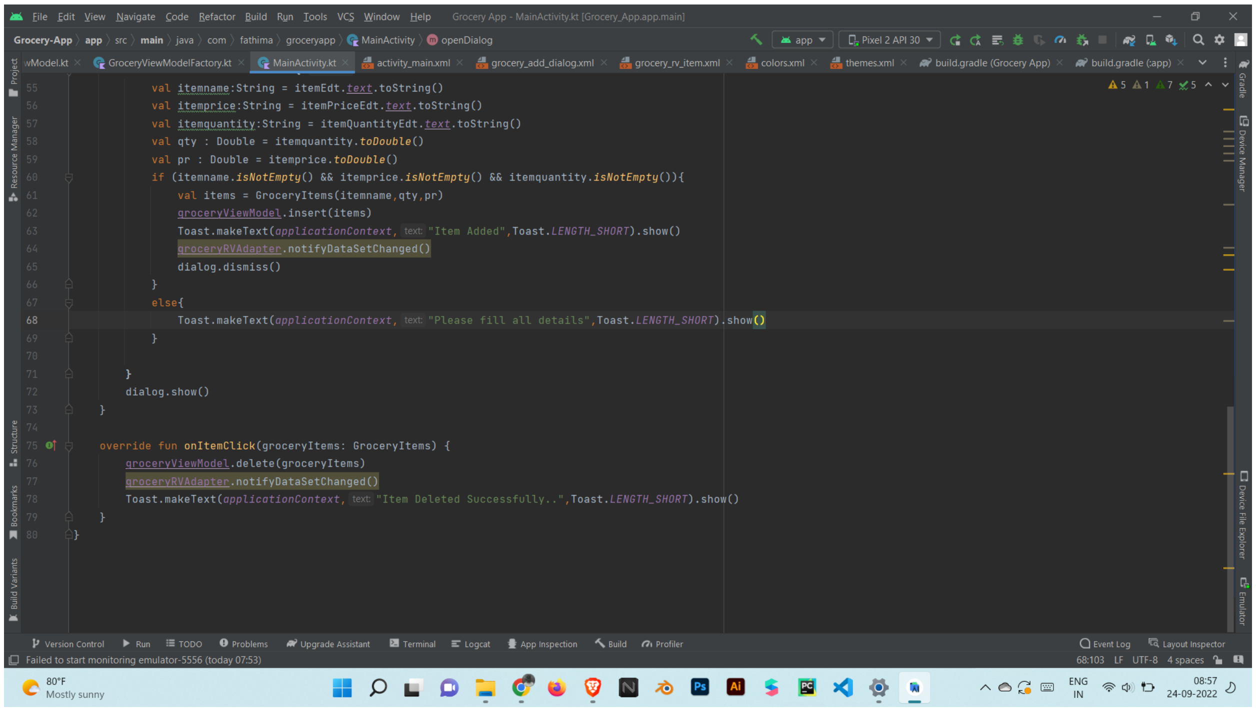
```
package com.fathima.groceryapp

import ...

class MainActivity : AppCompatActivity(),GroceryRVAdapter.GroceryItemClickInterface {
    lateinit var itemRV:RecyclerView
    lateinit var addFAB:FloatingActionButton
    lateinit var list: List<GroceryItems>
    lateinit var groceryRVAdapter: GroceryRVAdapter
    lateinit var groceryViewModel: GroceryViewModel
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        itemRV = findViewById(R.id.rvitems)
        addFAB = findViewById(R.id.fabAdd)
        list = ArrayList<GroceryItems>()
        groceryRVAdapter = GroceryRVAdapter(list, groceryItemClickInterface: this)
        itemRV.layoutManager = LinearLayoutManager( context: this)
        itemRV.adapter = groceryRVAdapter
        val groceryRepository = GroceryRepository(GroceryDatabase( context: this))
        val factory = GroceryViewModelFactory(groceryRepository)
        groceryViewModel = ViewModelProvider( owner: this,factory).get(GroceryViewModel::class.java)
        groceryViewModel.getAllGroceryItems().observe( owner: this, Observer {  it: List<GroceryItems>!
            groceryRVAdapter.list = it
            groceryRVAdapter.notifyDataSetChanged()
        })
        addFAB.setOnClickListener{  it: View!
            openDialog()
        }

    }
```

Failed to start monitoring emulator-5556 (today 07:53)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Grocery App - MainActivity.kt [Grocery_App.app.main]
Grocery-App 〉 app 〉 src 〉 main 〉 java 〉 com 〉 fathima 〉 groceryapp 〉 MainActivity 〉 openDialog
wModel.kt GroceryViewModelFactory.kt MainActivity.kt activity_main.xml grocery_add_dialog.xml grocery_rv_item.xml colors.xml themes.xml build.gradle (Grocery App) build.gradle (:app)
Version Control Run TODO Problems Upgrade Assistant Terminal Logcat App Inspection Build Profiler Event Log Layout Inspector
68:103 LF UTF-8 4 spaces
80°F Mostly sunny
ENG IN 08:56 24-09-2022

```kotlin
    }
    fun openDialog(){
        val dialog = Dialog( context: this)
        dialog.setContentView(R.layout.grocery_add_dialog)
        val cancelbtn = dialog.findViewById<AppCompatButton>(R.id.idbtncancel)
        val addbtn = dialog.findViewById<AppCompatButton>(R.id.idbtnadd)
        val itemEdt = dialog.findViewById<EditText>(R.id.idEdtitemname)
        val itemPriceEdt = dialog.findViewById<EditText>(R.id.idEdtitemprice)
        val itemQuantityEdt = dialog.findViewById<EditText>(R.id.idEdtitemquantity)
        cancelbtn.setOnClickListener {  it: View!
            dialog.dismiss()
        }
        addbtn.setOnClickListener {  it: View!
            val itemname:String = itemEdt.text.toString()
            val itemprice:String = itemPriceEdt.text.toString()
            val itemquantity:String = itemQuantityEdt.text.toString()
            val qty : Double = itemquantity.toDouble()
            val pr : Double = itemprice.toDouble()
            if (itemname.isNotEmpty() && itemprice.isNotEmpty() && itemquantity.isNotEmpty()){
                val items = GroceryItems(itemname,qty,pr)
                groceryViewModel.insert(items)
                Toast.makeText(applicationContext,  text: "Item Added",Toast.LENGTH_SHORT).show()
                groceryRVAdapter.notifyDataSetChanged()
                dialog.dismiss()
            }
            else{
                Toast.makeText(applicationContext,  text: "Please fill all details",Toast.LENGTH_SHORT).show()
            }

        }
        dialog.show()
```

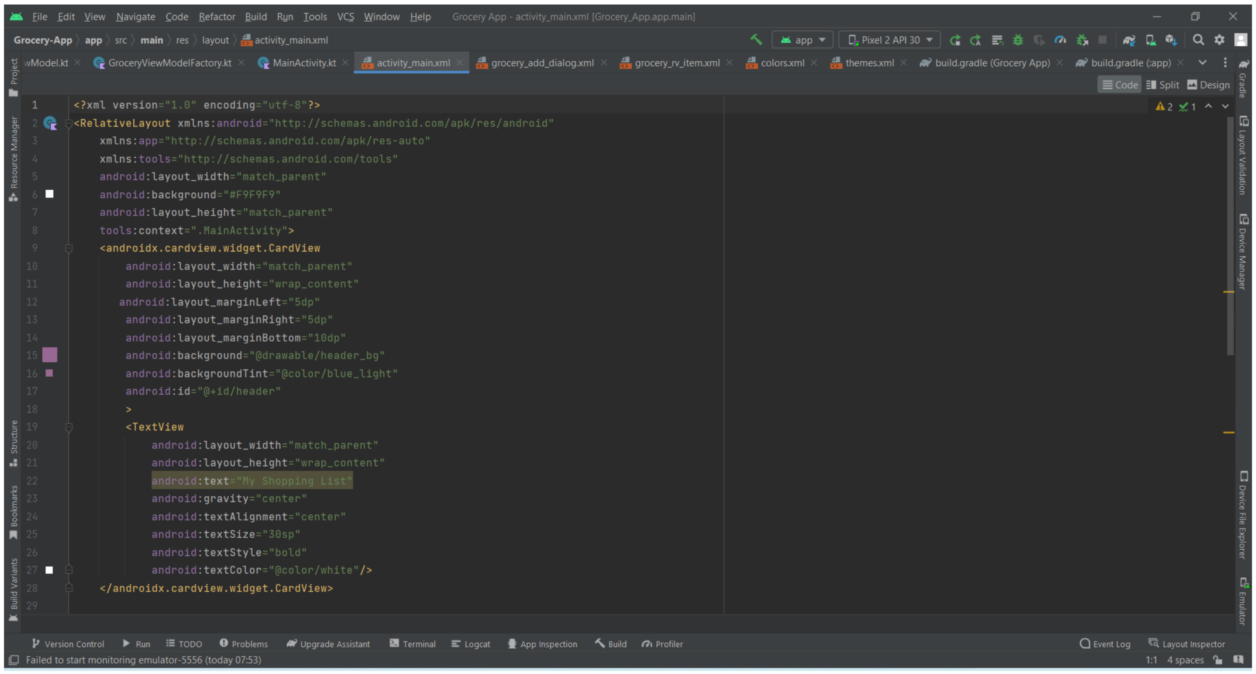
```
Failed to start monitoring emulator-5556 (today 07:53)
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Grocery App - MainActivity.kt [Grocery_App.app.main]
Grocery-App 〉 app 〉 src 〉 main 〉 java 〉 com 〉 fathima 〉 groceryapp 〉 MainActivity 〉 openDialog
wModel.kt GroceryViewModelFactory.kt MainActivity.kt activity_main.xml grocery_add_dialog.xml grocery_rv_item.xml colors.xml themes.xml build.gradle (Grocery App) build.gradle (:app)
Version Control Run TODO Problems Upgrade Assistant Terminal Logcat App Inspection Build Profiler Event Log Layout Inspector
68:103 LF UTF-8 4 spaces
80°F Mostly sunny
ENG IN 08:57 24-09-2022

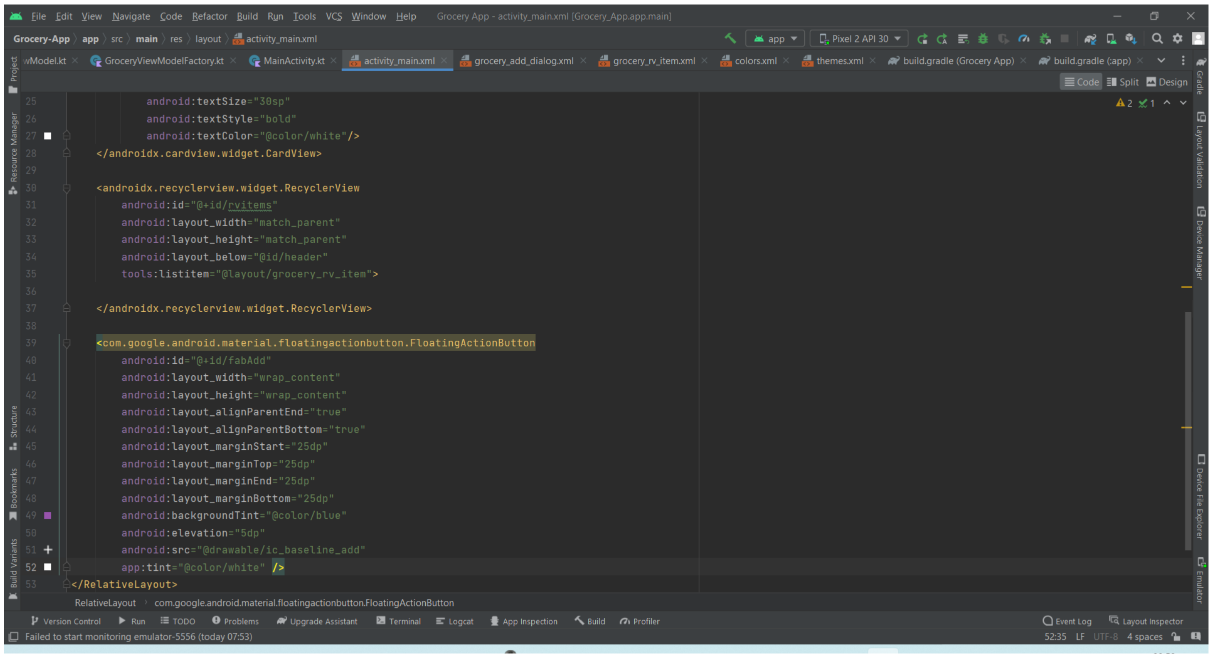
```kotlin
            val itemname:String = itemEdt.text.toString()
            val itemprice:String = itemPriceEdt.text.toString()
            val itemquantity:String = itemQuantityEdt.text.toString()
            val qty : Double = itemquantity.toDouble()
            val pr : Double = itemprice.toDouble()
            if (itemname.isNotEmpty() && itemprice.isNotEmpty() && itemquantity.isNotEmpty()){
                val items = GroceryItems(itemname,qty,pr)
                groceryViewModel.insert(items)
                Toast.makeText(applicationContext,  text: "Item Added",Toast.LENGTH_SHORT).show()
                groceryRVAdapter.notifyDataSetChanged()
                dialog.dismiss()
            }
            else{
                Toast.makeText(applicationContext,  text: "Please fill all details",Toast.LENGTH_SHORT).show()
            }

        }
        dialog.show()
    }

    override fun onItemClick(groceryItems: GroceryItems) {
        groceryViewModel.delete(groceryItems)
        groceryRVAdapter.notifyDataSetChanged()
        Toast.makeText(applicationContext,  text: "Item Deleted Successfully..",Toast.LENGTH_SHORT).show()
    }
}
```

```
Failed to start monitoring emulator-5556 (today 07:53)
```

- activity_main.xml

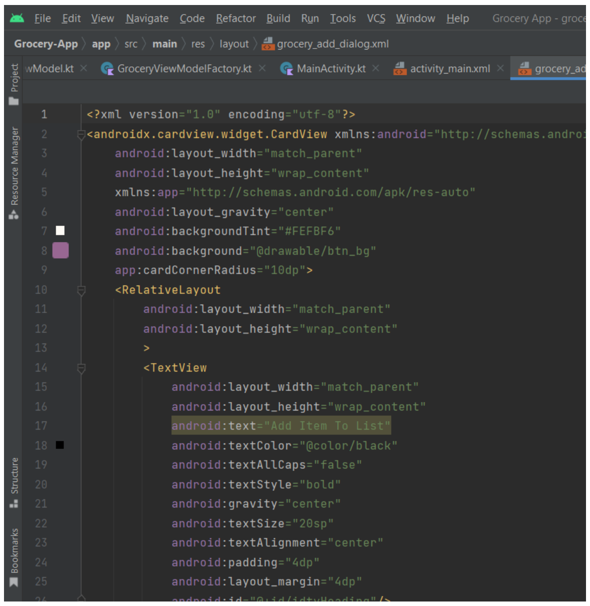


```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="#F9F9F9"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
       android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:layout_marginBottom="10dp"
        android:background="@drawable/header_bg"
        android:backgroundTint="@color/blue_light"
        android:id="@+id/header"
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="My Shopping List"
            android:gravity="center"
            android:textAlignment="center"
            android:textSize="30sp"
            android:textStyle="bold"
            android:textColor="@color/white"/>
    </androidx.cardview.widget.CardView>

```

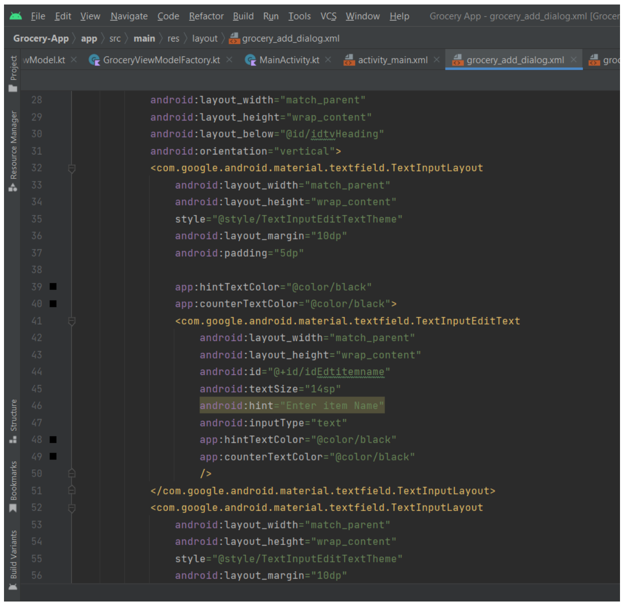


```xml
            android:textSize="30sp"
            android:textStyle="bold"
            android:textColor="@color/white"/>
    </androidx.cardview.widget.CardView>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvitems"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/header"
        tools:listitem="@layout/grocery_rv_item">

    </androidx.recyclerview.widget.RecyclerView>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:backgroundTint="@color/blue"
        android:elevation="5dp"
        android:src="@drawable/ic_baseline_add"
        app:tint="@color/white" />
</RelativeLayout>
```

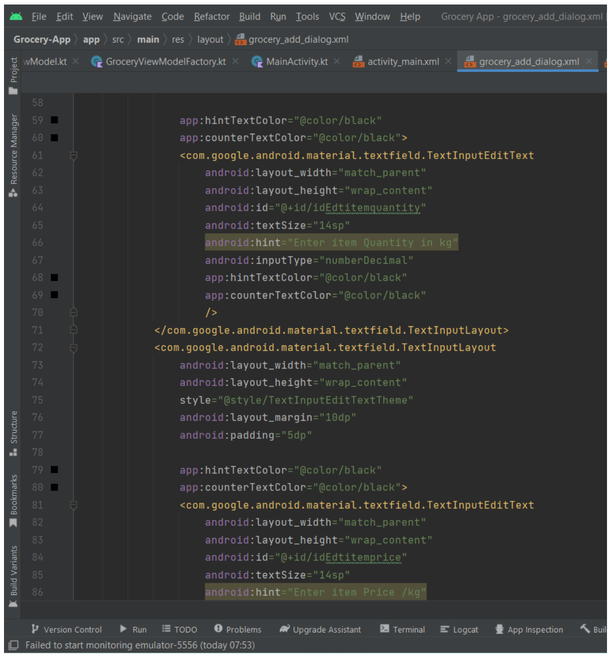
- grocery_add_dialog.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.andro
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_gravity="center"
    android:backgroundTint="#FEFBF6"
    android:background="@drawable/btn_bg"
    app:cardCornerRadius="10dp">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Add Item To List"
            android:textColor="@color/black"
            android:textAllCaps="false"
            android:textStyle="bold"
            android:gravity="center"
            android:textSize="20sp"
            android:textAlignment="center"
            android:padding="4dp"
            android:layout_margin="4dp"
            android:id="@+id/idtvHeading"/>
```

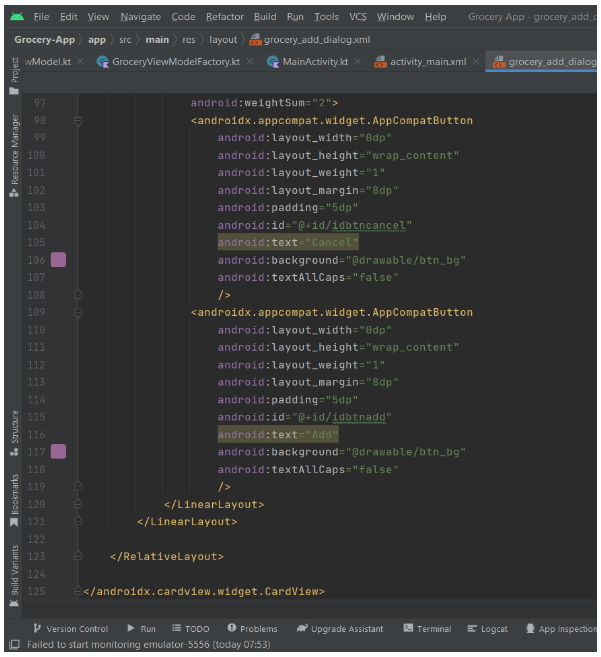


```xml
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/idtvHeading"
            android:orientation="vertical">
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                style="@style/TextInputEditTextTheme"
                android:layout_margin="10dp"
                android:padding="5dp"

                app:hintTextColor="@color/black"
                app:counterTextColor="@color/black">
                <com.google.android.material.textfield.TextInputEditText
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/idEdtitemname"
                    android:textSize="14sp"
                    android:hint="Enter item Name"
                    android:inputType="text"
                    app:hintTextColor="@color/black"
                    app:counterTextColor="@color/black"
                    />
            </com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                style="@style/TextInputEditTextTheme"
                android:layout_margin="10dp"
```



```xml

                app:hintTextColor="@color/black"
                app:counterTextColor="@color/black">
                <com.google.android.material.textfield.TextInputEditText
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/idEdtitemquantity"
                    android:textSize="14sp"
                    android:hint="Enter item Quantity in kg"
                    android:inputType="numberDecimal"
                    app:hintTextColor="@color/black"
                    app:counterTextColor="@color/black"
                    />
            </com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                style="@style/TextInputEditTextTheme"
                android:layout_margin="10dp"
                android:padding="5dp"

                app:hintTextColor="@color/black"
                app:counterTextColor="@color/black">
                <com.google.android.material.textfield.TextInputEditText
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:id="@+id/idEdtitemprice"
                    android:textSize="14sp"
                    android:hint="Enter item Price /kg"
```

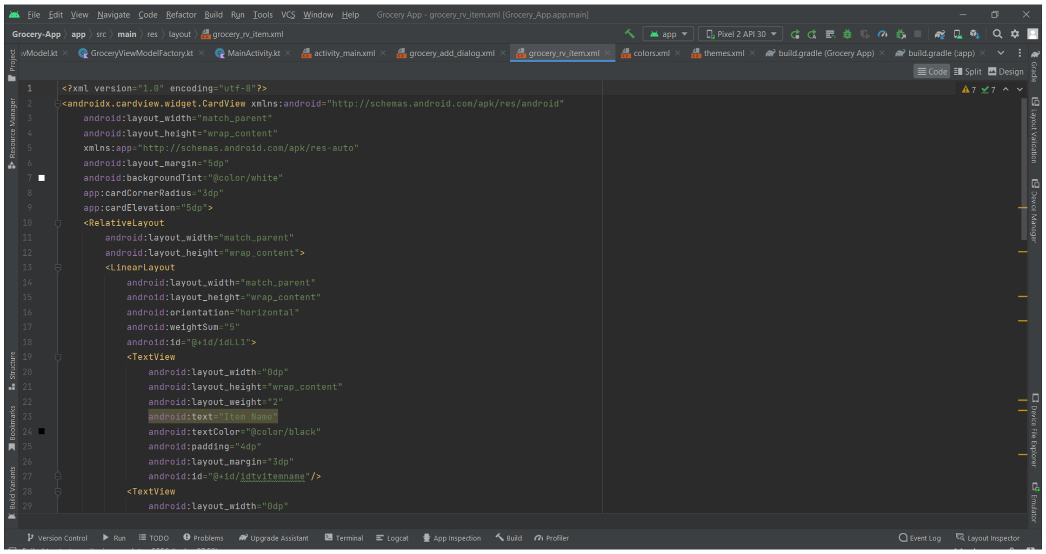
Failed to start monitoring emulator-5556 (today 07:53)



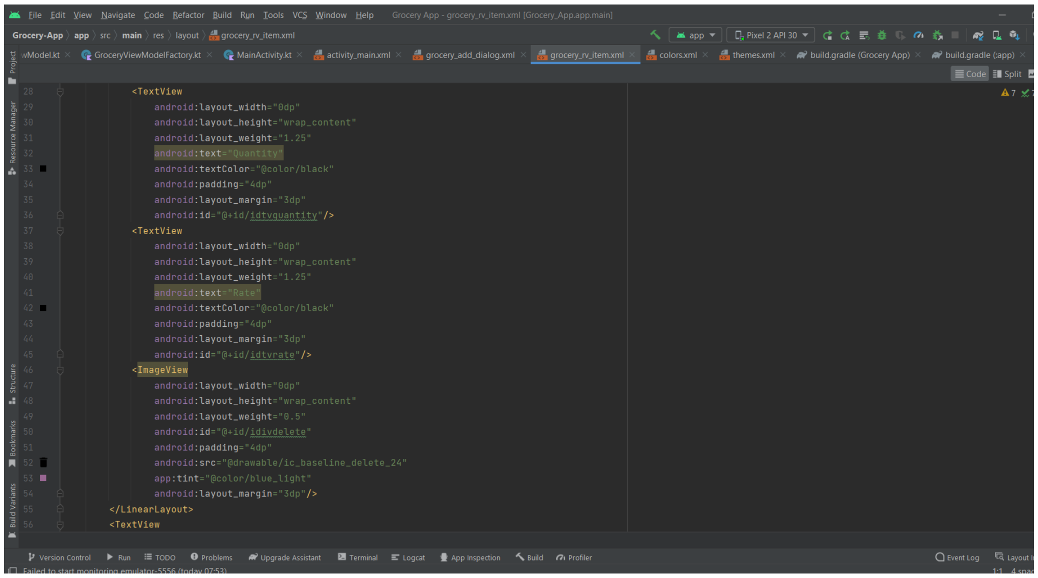
```xml
                android:weightSum="2">
                <androidx.appcompat.widget.AppCompatButton
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:layout_margin="8dp"
                    android:padding="5dp"
                    android:id="@+id/idbtncancel"
                    android:text="Cancel"
                    android:background="@drawable/btn_bg"
                    android:textAllCaps="false"
                    />
                <androidx.appcompat.widget.AppCompatButton
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:layout_margin="8dp"
                    android:padding="5dp"
                    android:id="@+id/idbtnadd"
                    android:text="Add"
                    android:background="@drawable/btn_bg"
                    android:textAllCaps="false"
                    />
            </LinearLayout>
        </LinearLayout>

    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

Failed to start monitoring emulator-5556 (today 07:53)

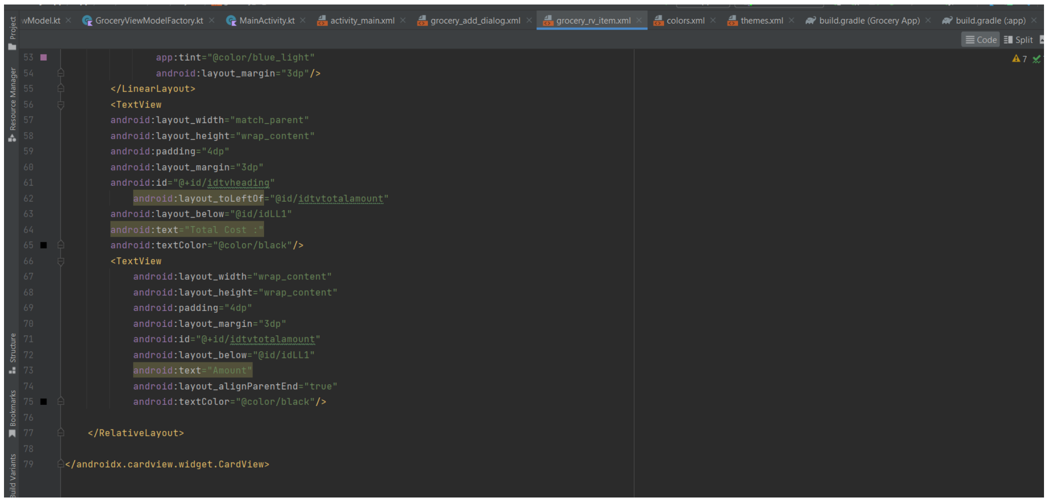
- grocery_rv_item.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="5dp"
    android:backgroundTint="@color/white"
    app:cardCornerRadius="3dp"
    app:cardElevation="5dp">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:weightSum="5"
            android:id="@+id/idLL1">
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="2"
                android:text="Item Name"
                android:textColor="@color/black"
                android:padding="4dp"
                android:layout_margin="3dp"
                android:id="@+id/idtvitemname"/>
            <TextView
                android:layout_width="0dp"
```

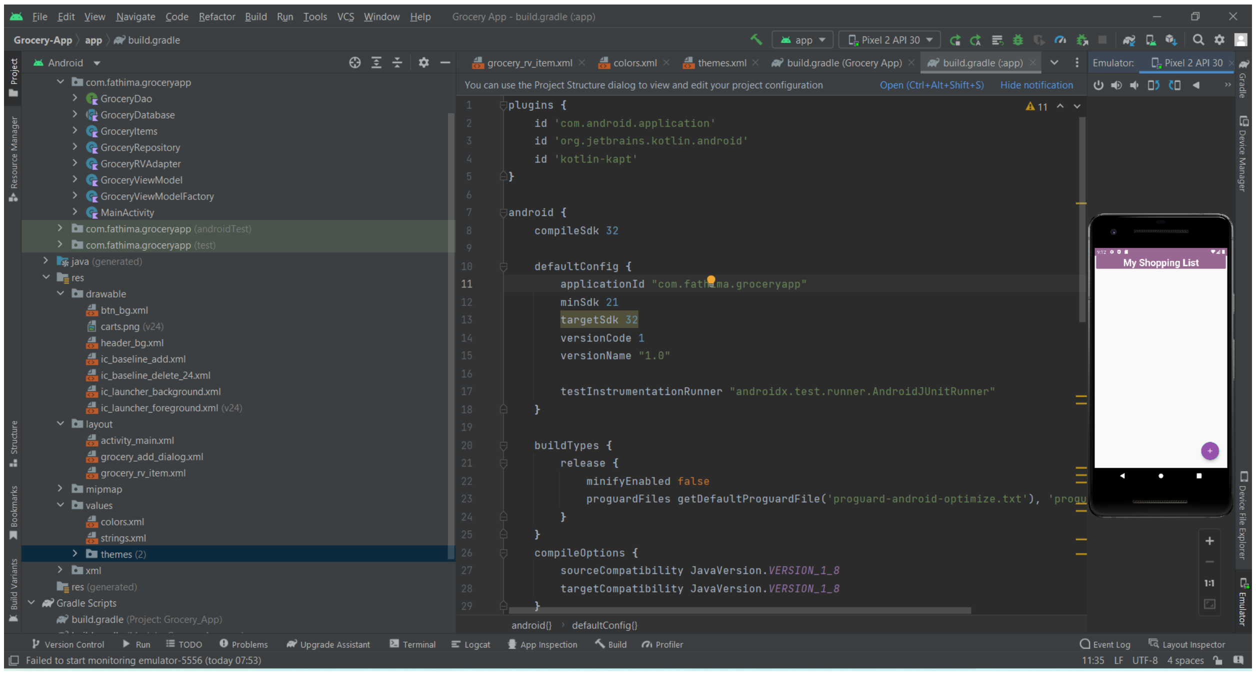


```xml
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1.25"
                android:text="Quantity"
                android:textColor="@color/black"
                android:padding="4dp"
                android:layout_margin="3dp"
                android:id="@+id/idtvquantity"/>
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1.25"
                android:text="Rate"
                android:textColor="@color/black"
                android:padding="4dp"
                android:layout_margin="3dp"
                android:id="@+id/idtvrate"/>
            <ImageView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="0.5"
                android:id="@+id/idivdelete"
                android:padding="4dp"
                android:src="@drawable/ic_baseline_delete_24"
                app:tint="@color/blue_light"
                android:layout_margin="3dp"/>
        </LinearLayout>
        <TextView
```



```xml
                app:tint="@color/blue_light"
                android:layout_margin="3dp"/>
        </LinearLayout>
        <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="4dp"
        android:layout_margin="3dp"
        android:id="@+id/idtvheading"
            android:layout_toLeftOf="@id/idtvtotalamount"
        android:layout_below="@id/idLL1"
        android:text="Total Cost :"
        android:textColor="@color/black"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="4dp"
            android:layout_margin="3dp"
            android:id="@+id/idtvtotalamount"
            android:layout_below="@id/idLL1"
            android:text="Amount"
            android:layout_alignParentEnd="true"
            android:textColor="@color/black"/>

    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

File structure:



Grocery App - build.gradle (:app)

Grocery-App 〉 app 〉 build.gradle

```
com.fathima.groceryapp
    GroceryDao
    GroceryDatabase
    GroceryItems
    GroceryRepository
    GroceryRVAdapter
    GroceryViewModel
    GroceryViewModelFactory
    MainActivity
com.fathima.groceryapp (androidTest)
com.fathima.groceryapp (test)
java (generated)
res
    drawable
        btn_bg.xml
        carts.png (v24)
        header_bg.xml
        ic_baseline_add.xml
        ic_baseline_delete_24.xml
        ic_launcher_background.xml
        ic_launcher_foreground.xml (v24)
    layout
        activity_main.xml
        grocery_add_dialog.xml
        grocery_rv_item.xml
    mipmap
    values
        colors.xml
        strings.xml
        themes (2)
    xml
res (generated)
Gradle Scripts
    build.gradle (Project: Grocery_App)
```

You can use the Project Structure dialog to view and edit your project configuration

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-kapt'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.fathima.groceryapp"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'progu
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

Failed to start monitoring emulator-5556 (today 07:53)