Smart Internz

My Grocery (App Report)

1. Introduction

1.1 OverView

My Grocery is an android app supported on API 19 or above which help you to store list of grocery items with their price and quantity in an organized manner.

1.2 Purpose

By the use of this app the user can store their day to day grocery items that need to be purchased from shops in our app database. so, that they don't forget any item in future and have rough idea of overall budget required.

2. Literature Survey

2.1 Existing problem

We buy groceries more often going to market and if the no of items is big then we can not remember all of items that need to be purchased with their respective quantity. So, for that what we try to do is to make list in form of some physical paper. But it leads to wastage of papers unnecessary.Also, we need to calculate manually the overall cost of each item by multiplying quantity x Rate which can be time consuming and also error prone.

2.2 Proposed Solution-

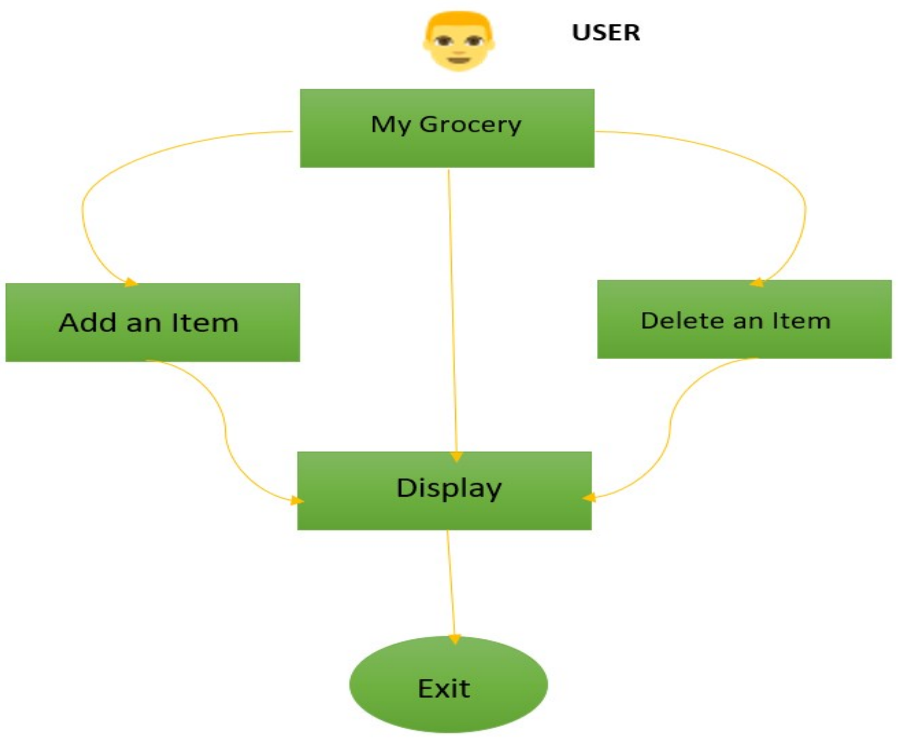
Now, an ultimate solution to this problem is to have an app which can store all our items that need to be buy ed that where My Grocery comes in place. My Grocery is simple to use, reliable, User friendly listing app which keep records of your Grocery item with their Quantity, Rate and total cost made. So, you can have an esteemed budget before you go to shopping.

Functions:-

- Add an item(Name, Quantity, Rate)
- Delete an item
- See total cost

3. *Theoretical Analysis*

3.1 Block Diagram



3.2 Hardware/Software Requirements-

In order to run My Grocery app you must have an android device with good quality of touch sensitivity and have Android Version KitKat(API 19) or above with play store installed.

4. Experiment Investigation

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

LiveData: A data holder class that can be observed. Always holds/caches the latest version of data, and notifies its observers when data has changed. LiveData is lifecycle aware. UI components just observe relevant data and don't stop or resume observation. LiveData automatically manages all of this since it's aware of the relevant lifecycle status changes while observing.

ViewModel: Acts as a communication center between the Repository (data) and the UI. The UI no longer needs to worry about the origin of the data. ViewModel instances survive Activity/Fragment recreation.

Repository: A class that you create that is primarily used to manage multiple data sources.

Entity: Annotated class that describes a database table when working with Room. Room database: Simplifies database work and serves as an access point to the underlying SQLite database (hides SQLiteOpenHelper). The Room database uses the DAO to issue queries to the SQLite database.

SQLite database: On device storage. The Room persistence library creates and maintains this database for you.

DAO: Data access object. A mapping of SQL queries to functions. When you use a DAO, you call the methods, and Room takes care of the rest.

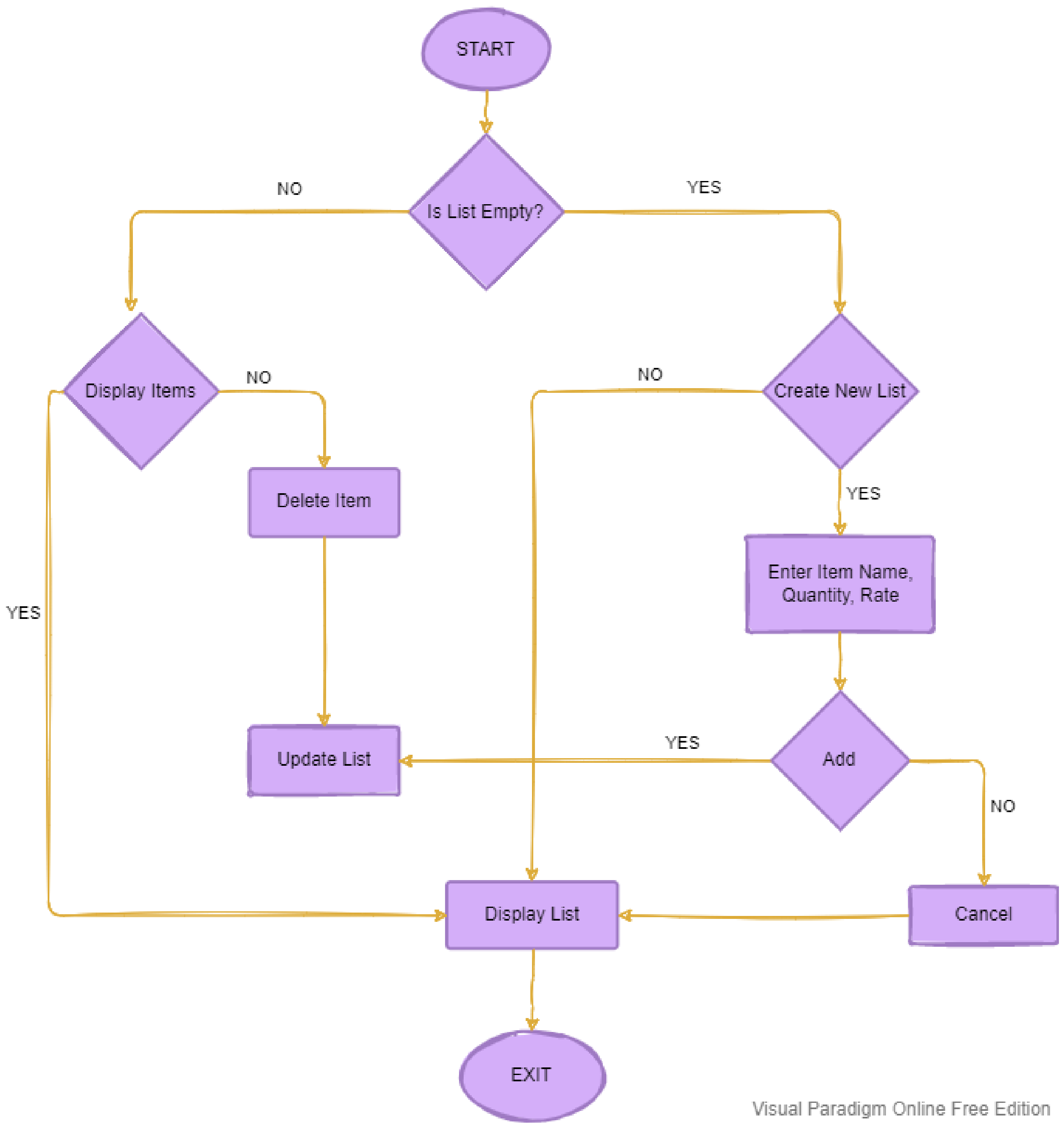
RecyclerView: It is a container and is used to display the collection of data in a large amount of dataset that can be scrolled very effectively by maintaining a limited number of views.

Coroutines: Coroutines are lightweight thread, we use a coroutine to perform an operation on other threads, by this our main thread doesn't block and our app doesn't crash.

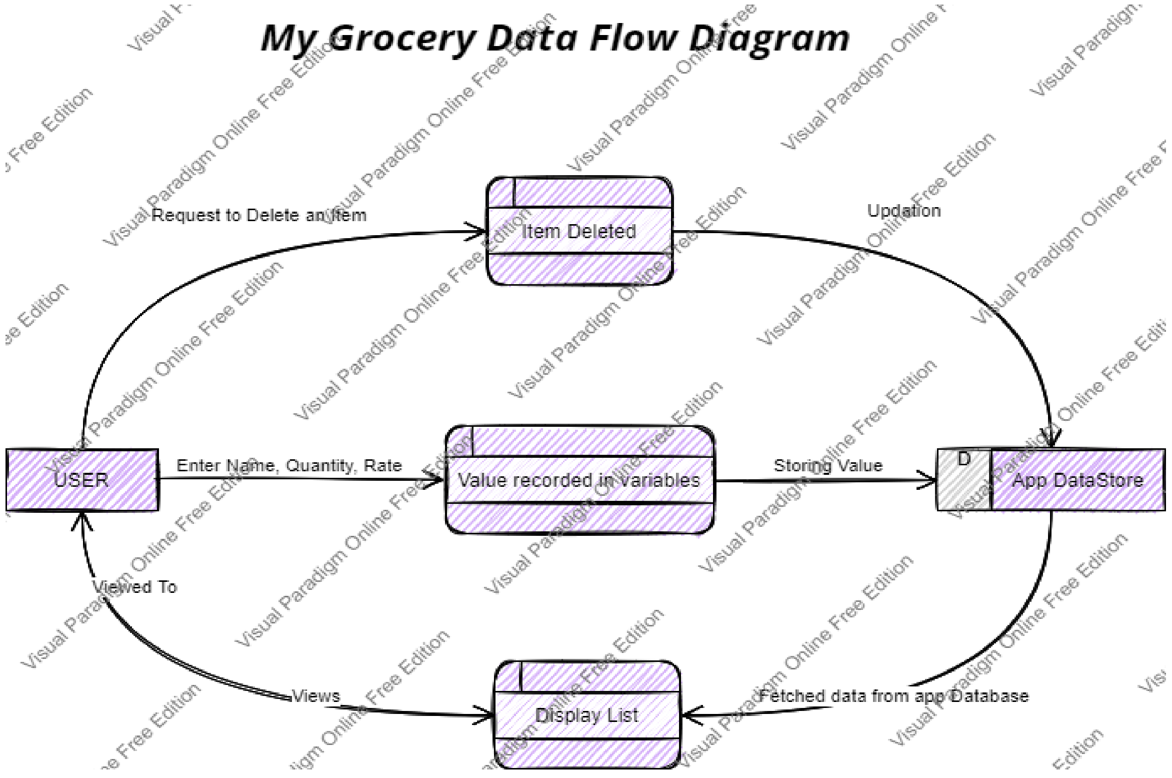
5. Flow Chart

a)

Flow Chart of App Use Case



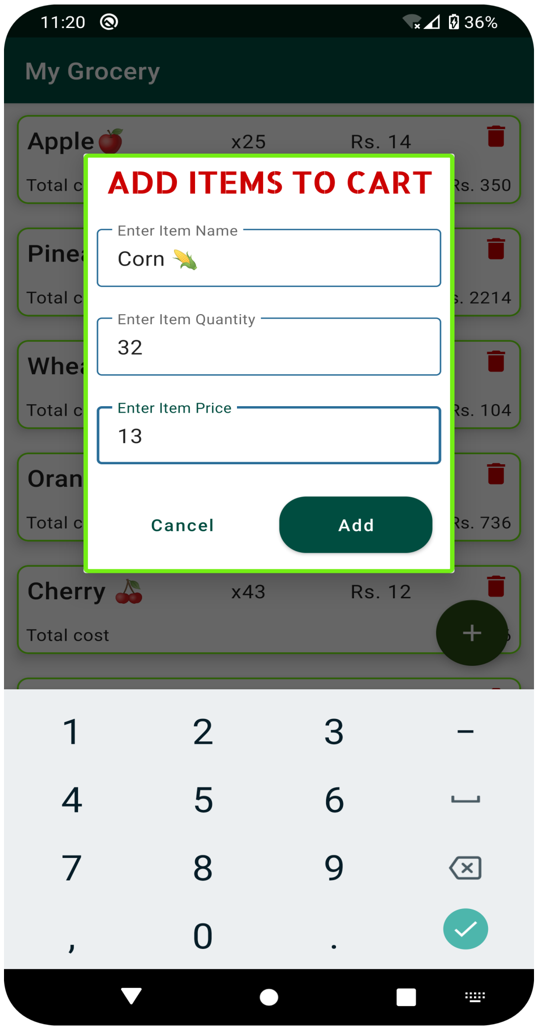
b)



My Grocery Data Flow Diagram
Request to Delete an Item
Item Deleted
Updation
USER
Enter Name, Quantity, Rate
Value recorded in variables
Storing Value
D
App DataStore
Viewed To
Views
Display List
Fetched data from app Database

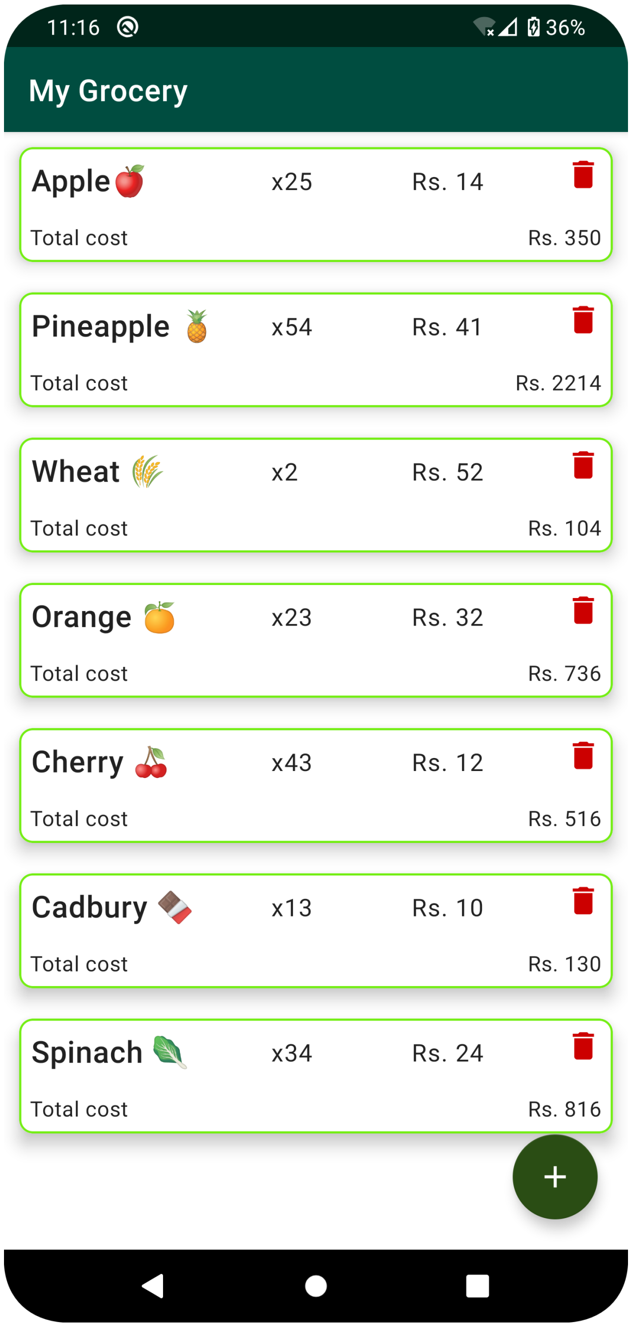
6. Result

Add Item Screen



Smart Internz

Light Theme



11:16 36%

My Grocery

Apple🍎 x25 Rs. 14
Total cost Rs. 350

Pineapple 🍍 x54 Rs. 41
Total cost Rs. 2214

Wheat 🌾 x2 Rs. 52
Total cost Rs. 104

Orange 🍊 x23 Rs. 32
Total cost Rs. 736

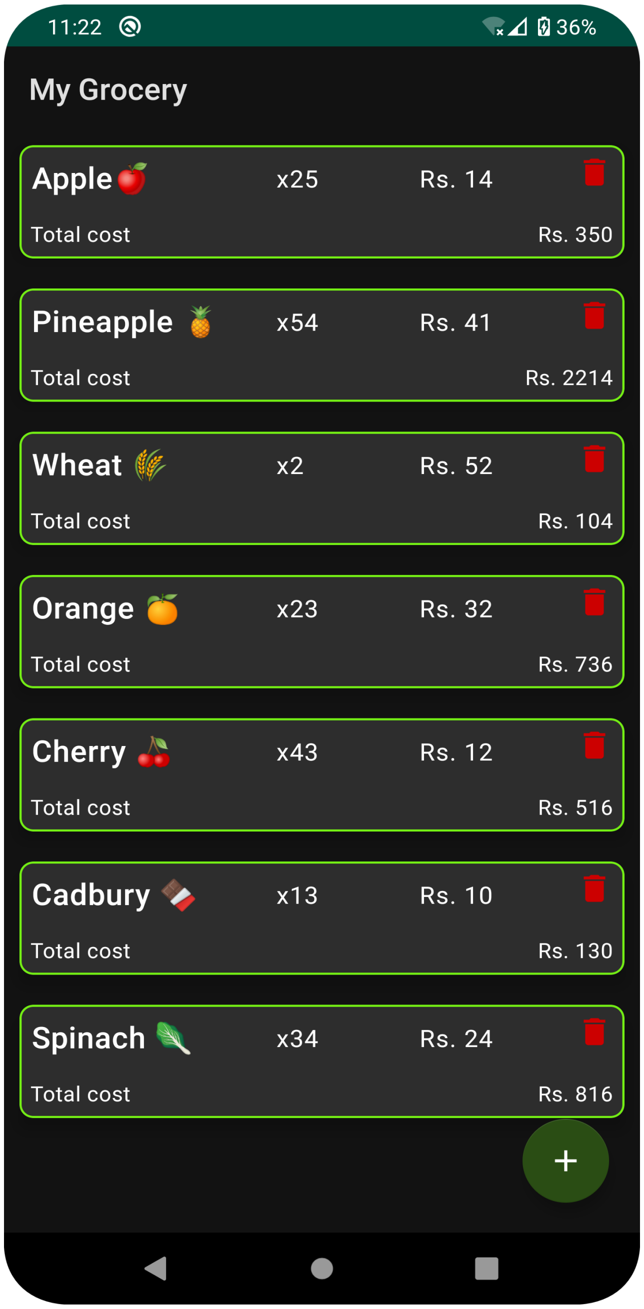
Cherry 🍒 x43 Rs. 12
Total cost Rs. 516

Cadbury 🍫 x13 Rs. 10
Total cost Rs. 130

Spinach 🥬 x34 Rs. 24
Total cost Rs. 816

+

Dark Theme



11:22 36%

My Grocery

Apple🍎 x25 Rs. 14
Total cost Rs. 350

Pineapple 🍍 x54 Rs. 41
Total cost Rs. 2214

Wheat 🌾 x2 Rs. 52
Total cost Rs. 104

Orange 🍊 x23 Rs. 32
Total cost Rs. 736

Cherry 🍒 x43 Rs. 12
Total cost Rs. 516

Cadbury 🍫 x13 Rs. 10
Total cost Rs. 130

Spinach 🥬 x34 Rs. 24
Total cost Rs. 816

+

7. Advantages & Disadvantages

Adavantages-

- Simple, convenient, clean UI so, that any group of user can use it.
- See overall budget
- Any no. of item can be added

Diadvantages-

- We can't specify item quantity type(kg, l, m).
- Item with long name appear to be break apart.
- Redundancy also comes.

9. Conclusion

My Grocery is the simple data store app which uses MVVM architecture and Room library to store list of Grocery Items which helps the user to keep track of needed.

10. Future Scope

- More functionalities can be added such that from same app user can place orders of items from nearest Grocery Center.
- Ui can be made more attractive.
- Option to Increase or decrease quantity of item or update an existing item.

11. BiBilography-

- **Google:** https://www.google.co.in/
- **(Data Persistence) :** https://developer.android.com/courses/android-basics-kotlin/unit-5
- **Stack Overflow:** https://stackoverflow.com/
- **You Tube(G.F.G)**: https://www.youtube.com/embed/vdcLb_Y71Ic
- **Smart Inernz:** https://smartinternz.com/Student/guided_project_info/54155#

12. Appendix

Source Code

a) MainActivity.kt

```
package com.example.mygrocery

import android.annotation.SuppressLint
import android.app.Dialog
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.mygrocery.databinding.ActivityMainBinding
import com.example.mygrocery.databinding.GroceryAddDialogBinding
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity(),
GroceryRVAdapter.GroceryItemClickInterface {

    lateinit var itemsRV:RecyclerView
    lateinit var addFAB:FloatingActionButton
    lateinit var list: List<GroceryItems>
    lateinit var groceryRVAdapter: GroceryRVAdapter
    lateinit var groceryViewModel: GroceryViewModel
    lateinit var binding:GroceryAddDialogBinding

    @SuppressLint("NotifyDataSetChanged")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = GroceryAddDialogBinding.inflate(layoutInflater)
        setContentView(R.layout.activity_main)
        itemsRV=findViewById(R.id.idRVItems)
        addFAB=findViewById(R.id.FABAdd)
        list = ArrayList<GroceryItems>()
        groceryRVAdapter = GroceryRVAdapter(list, this)
        itemsRV.layoutManager= LinearLayoutManager(this)
        itemsRV.adapter=groceryRVAdapter
        val groceryRepository= GroceryRepository(GroceryDatabase(this))
        val factory= GroceryViewModelFactory(groceryRepository)
        groceryViewModel=
ViewModelProvider(this,factory).get(GroceryViewModel::class.java)
        groceryViewModel.getAllGroceryItems().observe(this, Observer{
            groceryRVAdapter.list= it
            groceryRVAdapter.notifyDataSetChanged()
        })




        addFAB.setOnClickListener{
            addFAB.show()
            openDialog()
        }

    }


    @SuppressLint("NotifyDataSetChanged")
    fun openDialog(){
        val dialog= Dialog(this)
        dialog.setContentView(R.layout.grocery_add_dialog)
        val cancelBtn = dialog.findViewById<Button>(R.id.idBtnCancel)
        val addBtn = dialog.findViewById<Button>(R.id.idBtnAdd)
        val itmEdt = dialog.findViewById<EditText>(R.id.idEdItemName)
        val itmPriceEdt = dialog.findViewById<EditText>(R.id.idEdItemPrice)
        val itmQuantityEdt =
dialog.findViewById<EditText>(R.id.idEdItemQuantity)

        cancelBtn.setOnClickListener {
            dialog.dismiss()
        }

        addBtn.setOnClickListener {
            val itemName: String= itmEdt.text.toString()
            val itemPrice: String= itmPriceEdt.text.toString()
            val itemQuantity: String= itmQuantityEdt.text.toString()

            val qty: Int= itemQuantity.toInt()
            val pr: Int= itemPrice.toInt()


            if(itemName.isNotEmpty() && itemPrice.isNotEmpty() &&
itemQuantity.isNotEmpty()) {
                val items = GroceryItems(itemName, qty, pr)
                groceryViewModel.insert(items)
                Toast.makeText(applicationContext, "Item Inserted...",
Toast.LENGTH_SHORT).show()
                groceryRVAdapter.notifyDataSetChanged()
                dialog.dismiss()
            }


            else{
                    Toast.makeText(applicationContext,"Please Enter all
data..", Toast.LENGTH_SHORT).show()

            }



        }

        dialog.show()

    }

    @SuppressLint("NotifyDataSetChanged")
    override fun onItemClick(groceryItems: GroceryItems) {

        groceryViewModel.delete(groceryItems)
        groceryRVAdapter.notifyDataSetChanged()
        Toast.makeText(applicationContext,"Item Deleted...",
Toast.LENGTH_SHORT).show()

    }
}
```

b) GroceryItems.kt

```
package com.example.mygrocery
```

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

@Entity(tableName = "grocery_items")
data class GroceryItems (
    @ColumnInfo(name="itemName")
    var itemName: String,

    @ColumnInfo(name = "itemQuantity")
    var itemQuantity: Int,

     @ColumnInfo(name = "itemPrice")
     var itemPrice: Int
)


{
    @PrimaryKey(autoGenerate = true)
    var id: Int?= null
}
```

c) GroceryDatabase.kt

```
package com.example.mygrocery

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

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

    abstract  fun getGroceryDao(): MyGroceryDao

    companion object{
        @Volatile
        private var instance: GroceryDatabase?= null
        private val LOCK= Any()
        operator fun invoke(context: Context)= instance ?: synchronized(LOCK){
            instance ?:createDatabase(context).also {
                instance= it
            }

        }


        private fun createDatabase(context: Context)=

Room.databaseBuilder(context.applicationContext,GroceryDatabase::class.java,
"Grocery.db").build()
    }


}
```

d) GroceryRvAdapter.kt

```
package com.example.mygrocery

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.view.menu.MenuView
import androidx.recyclerview.widget.RecyclerView

class GroceryRVAdapter(var list: List<GroceryItems>, val
groceryItemClickInterface: GroceryItemClickInterface):
    RecyclerView.Adapter<GroceryRVAdapter.GroceryViewHolder>() {
    var sum: Int=0

    inner class GroceryViewHolder(itemView: View):
RecyclerView.ViewHolder(itemView){
        val nameTV =itemView.findViewById<TextView>(R.id.idTVItemName)
        val quantityTV =itemView.findViewById<TextView>(R.id.idTVQuantity)
        val rateTV =itemView.findViewById<TextView>(R.id.idTVRate)
        val amountTV =itemView.findViewById<TextView>(R.id.idTVTotalAmt)
        val deleteTV =itemView.findViewById<ImageView>(R.id.idTVDelete)
    }
    interface  GroceryItemClickInterface{
        fun onItemClick(groceryItems: GroceryItems)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
GroceryViewHolder {
        val view=
LayoutInflater.from(parent.context).inflate(R.layout.grocery_rv_item,parent,fals
e)
        return GroceryViewHolder(view)
    }

    override fun onBindViewHolder(holder: GroceryViewHolder, position: Int) {
        holder.nameTV.text= list.get(position).itemName
        holder.quantityTV.text= "x"+ list.get(position).itemQuantity.toString()
        holder.rateTV.text= "Rs. "+ list.get(position).itemPrice.toString()
        val itemTotal: Int= list.get(position).itemPrice
*list.get(position).itemQuantity
        holder.amountTV.text= "Rs. " + itemTotal.toString()

        holder.deleteTV.setOnClickListener{
            groceryItemClickInterface.onItemClick(list.get(position))
        }

    }

    override fun getItemCount(): Int {
        return list.size
    }
}
```

Note:- As the page limitation has been reached so I couldn't mention more of it.Kindly refer to Github link given below to see all codes.

Github: https://github.com/smartinternz02/SPSGP-54155-Virtual-Internship---Android-Application-Development-Using-Kotlin/tree/master

Made with love by Ansh Vikalp