Grocery Android App using MVVM and Room Database in Kotlin

1.Introduction

a. Overview

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

This app helps you to make a list of groceryitems alongwith its price and quantity.

b. Purpose

People forget the things they need to because of busy schedule in their life. Nowadays people are more sophisticated with technology and they tend to make remainder in their phone apps. This app helps people to make a list of grocery items they tend to buy and they can track their expenditure.

2. Literature Survey

a. Existing Problem

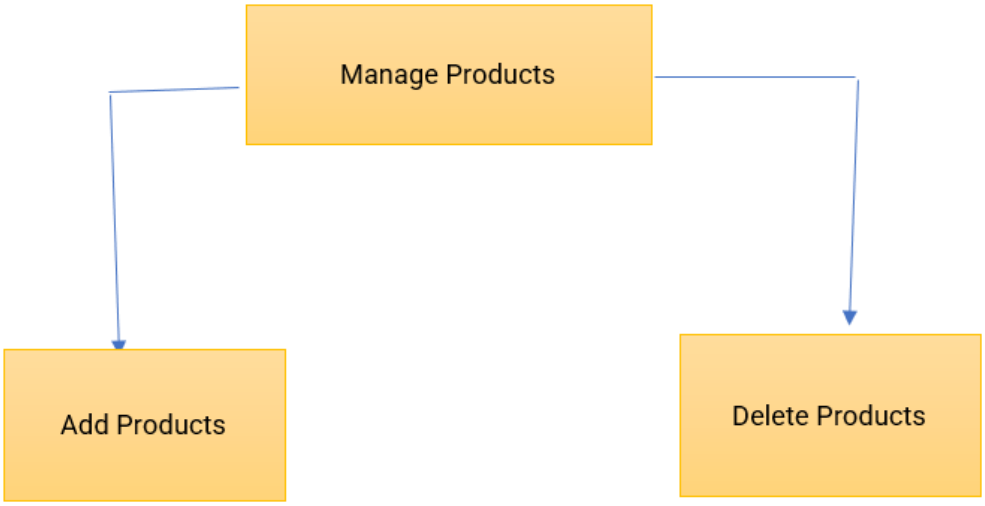
Users may forget the things to buy because of that they need to visit the shop again. This leads to frustration for customers as well for shop owners. Sometimes people failed to estimate the total amount needed to buy things and they carry some money. Because of this they need to drop some items.

b. ProposedSolution

To overcome this problem Grocery app was build to help the customers to estimate the cost of the products and act as remainder to purchase the items.

3. Theoritical Analysis

c. Block Diagram



a. **Hardware/ Software Designing**

- Android Studio
- Windows 11 OS
- Ram 8GB
- 20 GB Memory

4. Experimental Investigations

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.

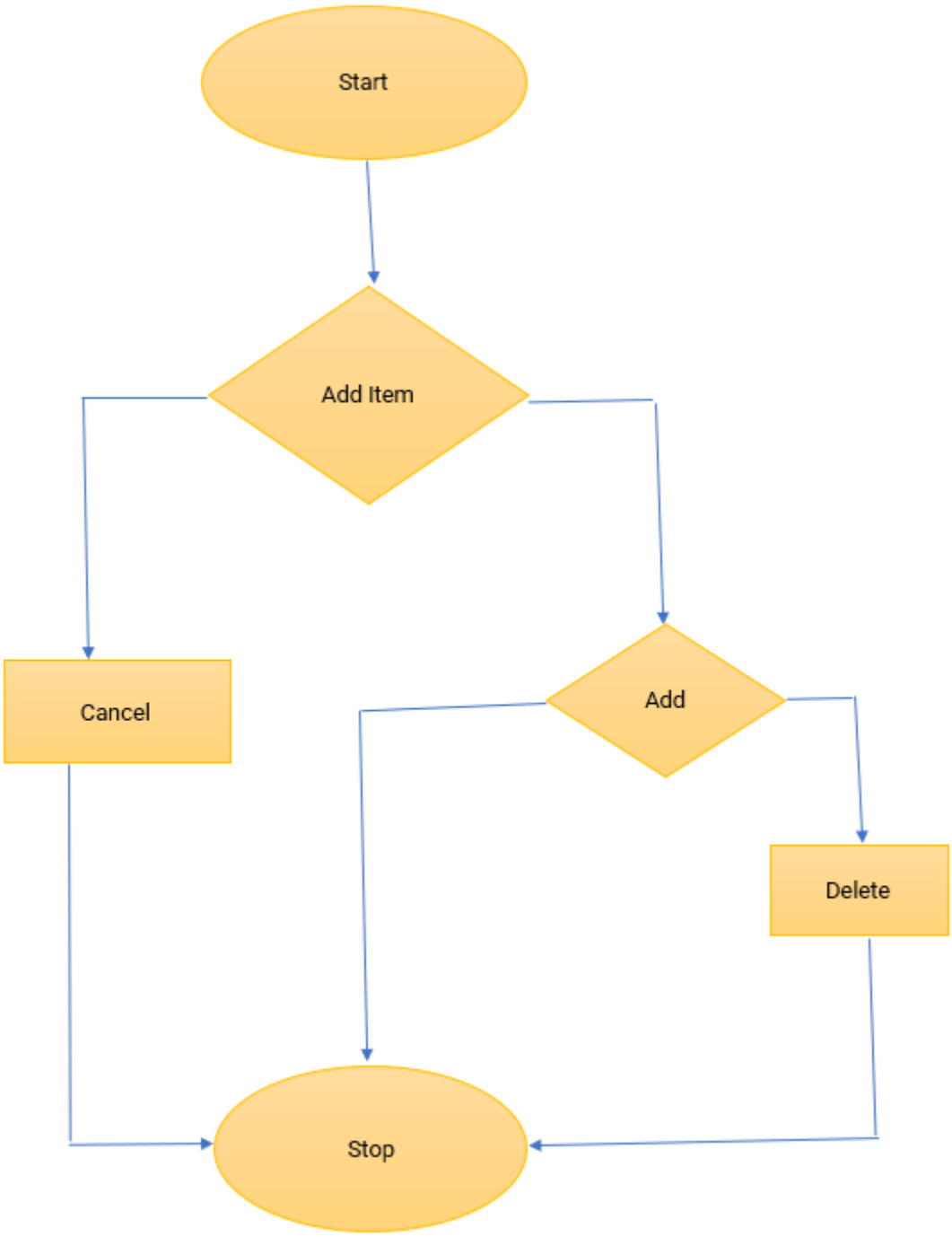
RecyclerView:

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.

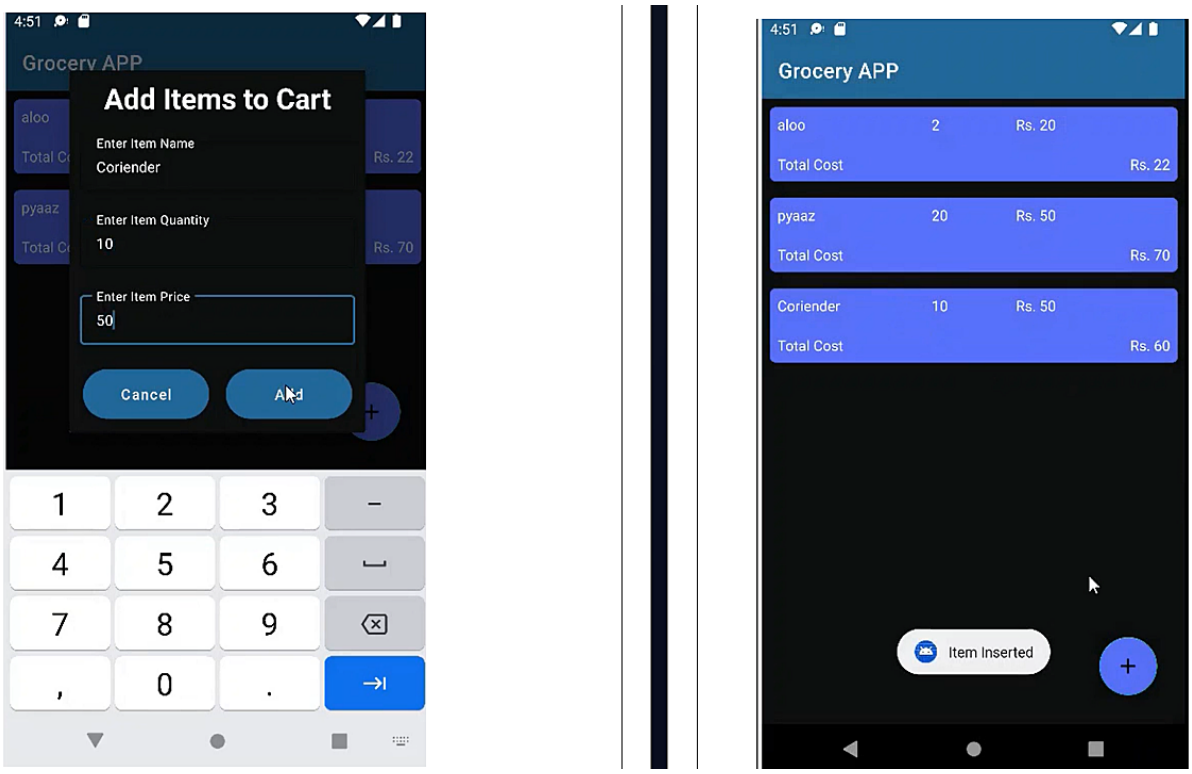
Coroutines:

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



6. Result



7. Advantages and Disadvantages

Advantages:

- Estimate the budget the budget of the product
- App act as a remainder

Disadvantages:

- This app makes the people more lazy to remember things
- People should be be careful while entering price for estimation

8. Applications:

- This is widely used by old people to remember the items need to be purchased
- Grocery shopkeeper calculate the amount by using this app
- Some busy schedule working people use this app for estimate the budget for purchase
- Some people use this app as remainder to buy things

9. Conclusion

This Android app development project helped me to learn concepts like Room Database, Coroutines, MVVM, etc. Working on this project made me confidentenough to apply my knowledge on android app development and to create an app for existing problem. I have used Kotlin to build this application and used android studio as a medium. All the functionality is coded in the classesand interfaces createdand the layout is designedusing xml.

10.Future Scope

This app can be connect to cloud for storage and tracking of information. Further this app can be made to work on online with login credentials.

11. Bibilography

- https://www.geeksforgeeks.org/how-tobuild-a-grocery- androidapp-using-mvvm-and-roomdatabase/
- https://www.youtube.com/watch?v=vdcLb_Y71Ic
- https://smartinternz.com/Student/externships_workspace_info/93784

- https://developer.android.com/guide

Appendix

MainActivity.kt

```
package com.rohan.groceryapp

    import android.app.Dialog
    import androidx.appcompat.app.AppCompatActivity
    import android.os.Bundle
    import android.widget.Button
    import android.widget.EditText
    import android.widget.Toast
    import androidx.lifecycle.ViewModelProvider
    import androidx.recyclerview.widget.LinearLayoutManager
    import androidx.recyclerview.widget.RecyclerView
    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 groceryViewModal: GroceryViewModal
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)
            itemsRv = findViewById(R.id.idRVItems)
            addFAB = findViewById(R.id.idFABAdd)
            list = ArrayList<GroceryItems>()
            groceryRVAdapter = GroceryRVAdapter(list, this)
            itemsRv.layoutManager = LinearLayoutManager(this)
```

```
        itemsRv.adapter = groceryRVAdapter
        val grocreryRepository = GroceryRepository(GroceryDatabase(this))
        val factory = GroceryViewModalFactory(grocreryRepository)
        groceryViewModal =
ViewModelProvider(this,factory).get(GroceryViewModal::class.java)
        groceryViewModal.getAllGroceryItems().observe(this, {
            groceryRVAdapter.list = it
            groceryRVAdapter.notifyDataSetChanged()
        })
        addFAB.setOnClickListener{
            openDialog()
        }
    }

    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 itemEdt = dialog.findViewById<EditText>(R.id.idEditItemName)
        val itemPriceEdt = dialog.findViewById<EditText>(R.id.idEditItemPrice)
        val itemQuantityEdt = dialog.findViewById<EditText>(R.id.idEditItemQuantity)

        cancelBtn.setOnClickListener{
            dialog.dismiss()
        }
        addBtn.setOnClickListener{
            val itemName: String = itemEdt.text.toString()
            val itemPrice: String = itemPriceEdt.text.toString()
            val itemQuantity: String = itemQuantityEdt.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)
                groceryViewModal.insert(items)
                Toast.makeText(applicationContext, "Item Inserted",
```

```
        Toast.LENGTH_SHORT).show()
                    groceryRVAdapter.notifyDataSetChanged()
                    dialog.dismiss()

                }else{
                    Toast.makeText(applicationContext, "Please enter all the data",
        Toast.LENGTH_SHORT).show()
                }
            }
            dialog.show()
        }

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

GroceryDatabase.kt

```
package com.rohan.groceryapp

    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() : GroceryDao

        companion object{
            @Volatile
```

```
        private var instance: GroceryDatabase? =null
        private val LOCK = Any()

        operator fun invoke(context: Context) = instance ?: synchronized(LOCK){
            val groceryDatabase = instance ?: createDatabase(context).also {
                instance = it
            }
            groceryDatabase
        }


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

GroceryItems.kt

```
package com.rohan.groceryapp

        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


}
```

GroceryRspository.kt

```
package com.rohan.groceryapp

class GroceryRepository(private val db: GroceryDatabase) {

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

    fun getAllItems() = db.getGroceryDao().getAllGroceryItems()

}
```

GroceryRVAdapter

```
package com.rohan.groceryapp

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

class GroceryRVAdapter(var list: List<GroceryItems>,
```

```
                val groceryItemClickInterface: GroceryItemClickInterface
                ) : RecyclerView.Adapter<GroceryRVAdapter.GroceryViewHolder>() {


    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_items,parent,false)
        return GroceryViewHolder(view)
    }

    override fun onBindViewHolder(holder: GroceryViewHolder, position: Int) {
        holder.NameTV.text = list.get(position).itemName
        holder.QuantityTV.text = 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
    }
}
```

GroceryViewModel.kt

```
package com.rohan.groceryapp

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch

class GroceryViewModal(private val repository: GroceryRepository) : ViewModel() {
    fun insert(items : GroceryItems) = GlobalScope.launch {
        repository.insert(items)
    }

    fun delete(items: GroceryItems) = GlobalScope.launch {
        repository.delete(items)
    }

    fun getAllGroceryItems() = repository.getAllItems()
}
```

GroceryViewModelFactory.kt

```
package com.rohan.groceryapp

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

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

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

}
```

Here is my GitHub link: https://github.com/smartinternz02/SPSGP-79768-Virtual-Internship---Android-Application-Development-Using-Kotlin

Google Developer Profile: https://g.dev/rohan0702

Demo Drive link: https://drive.google.com/file/d/1q6RwGGZDYOPo7r8Hgis8ODATkMlavHWo/view?usp=sharing