GROCERY APPLICATION USING KOTLIN

Jobin John Abraham

SBID : SB20220184052

1. INTRODUCTION

a. Overview:

An Android app that helps you to make a list of grocery items along with its price and quantity which is easy to add and remove items to buy from store.

b. Purpose:

Every people go to shopping regularly. It has become a very essential part of life. We people often forget what we want to buy and how much we have to buy.
The shopping cart app is the solution for that problem. With the help of this app, we can list all the items that we want to buy and for what price we want to buy.

2. LITERATURE SURVEY

a. Existing Problem

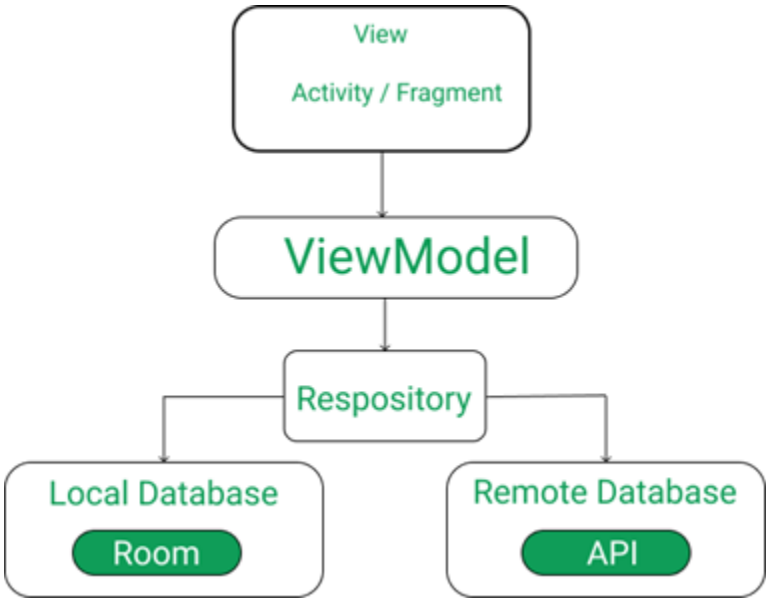
Users frequently forget items to buy because of which they have to run to shops again and again which is quite a frustrating and tiring situation and if our expenses cross out budget while shopping that could be a matter of concern.

b. Proposed Solution

To overcome this problematic situation, I built a grocery app which helps you to list down all the item that you need to buy along with its price.

3. THEORETICAL ANALYSIS

a. Block Diagram



b. **Hardware/Software Designing**

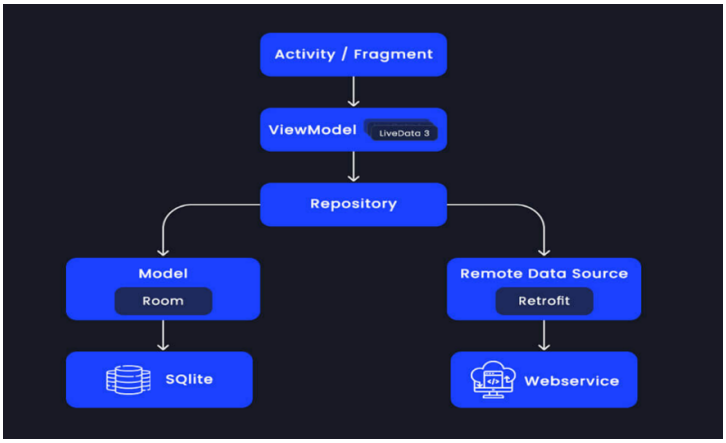
- Windows/Linux/MacOS
- Android Studio
- Android Device / Emulator
- Minimum 8GB RAM
- 15 GB free hard disk space

4. **EXPERIMENTAL INVESTIGATION**

In this project MVVM (Model View Model) was used for architectural patterns, Room for database, Coroutines and Recycler View to display the list of items.
live Data: A data holder class that can be observed. Always holds/caches the latest version of data, and notifies its observers when data has changed. Live Data is lifecycle aware. UI components just observe relevant data and don't stop or resume observation.

Live Data automatically manages all of this since it's aware of the relevant lifecycle status changes while observing.
View Model: 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. View Model 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 SQLite Open Helper). 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.
Recycler View: 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.

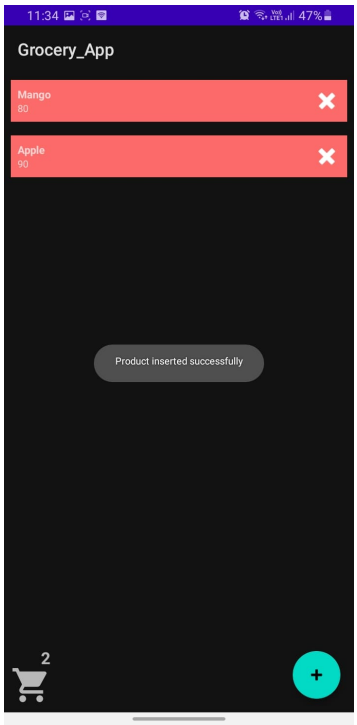
5. FLOWCHART



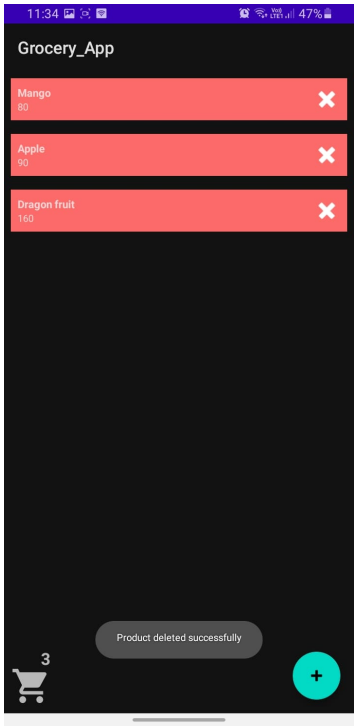
6. RESULT

Home screen

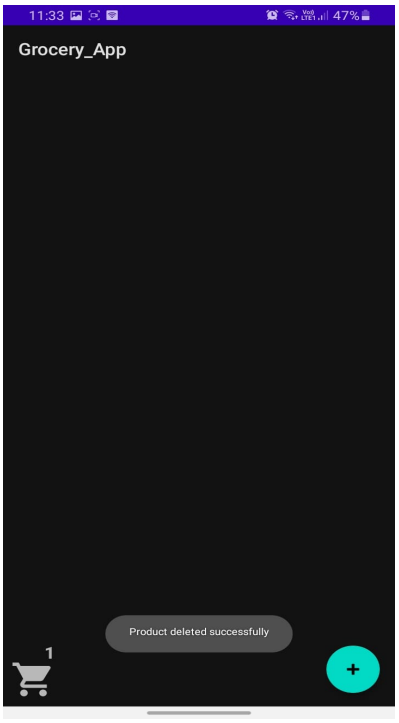
Inserted data



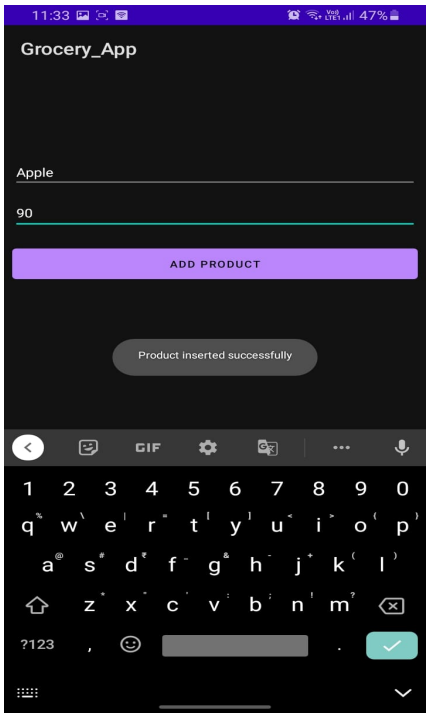
Entry Deleted



All Deleted



Adding Items



Demo video

https://drive.google.com/file/d/1j1wkpWueCA3IpDEAfByPDRSIl_vhMjFs/view?usp=sharing

1. **REFERENCE**

Google Scholar: https://www.scholar.google.com/

GeeksforGeeks:https://www.geeksforgeeks.org/how-to-build-a-grocery-androidapp-using-mvvm-and-room-database/

Android Developer: https://developer.android.com/codelabs/android-room-witha-view-kotlin#0

Youtube: https://www.youtube.com/watch?v=vdcLb_Y71Ic

2. **CONCLUSION**

Things but also as a user we generally forget items to purchase while shopping. Working on this project made me confident enough to apply my knowledge on android app development and create such an app. I have used Kotlin to build this application. All the functionality is coded in the classes and interfaces created and the layout is designed using xml.

3. **FUTURE SCOPE**

This project can be implemented with other ecommerce apps and companies to provide grocery list and also can be integrated with other ecommerce sites like Amazon, Jiomart, Flipkart etc.

4. **BIBILOGRAPHY**

APPENDIX

A. **Source Code**

Full Source code at : https://github.com/smartinternz02/SPSGP-63877-Virtual-Internship---Android-Application-Development-Using-Kotlin

MainActivity.kt

```
package com.jobin.grocery_app_android

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity(), CountInterface, ClickInterface, DeleteInterface {

    lateinit var productViewModel: ProductViewModel

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

        recyclerView.layoutManager = LinearLayoutManager(this)
        val productAdapter = ProductAdapter(this,this,this,this)
        recyclerView.adapter = productAdapter

                                                    productViewModel = ViewModelProvider(this,ViewModelProvider.AndroidViewModelFactory.getInstance(application)).get(
            ProductViewModel::class.java)
        productViewModel.getAllProductList.observe(this,{list->
            list?.let {
                productAdapter.updateList(it)
            }
        })

        floatingActionButton.setOnClickListener {
            val intent = Intent(this, AddEditActivity::class.java)
            startActivity(intent)
        }

    }

    override fun onCount(count: Int) {
        textView3.setText(count.toString())
    }
```

```
    override fun onClick(productModel: ProductModel) {
        val intent = Intent(this, AddEditActivity::class.java)
        intent.putExtra("type","Edit")
        intent.putExtra("productId",productModel.id)
        intent.putExtra("productName",productModel.productName)
        intent.putExtra("productPrice",productModel.productPrice)
        startActivity(intent)
    }

    override fun onDelete(productModel: ProductModel) {
        productViewModel.deleteProduct(productModel)
                                    Toast.makeText(this,"Product deleted successfully",
Toast.LENGTH_LONG).show()
    }
}
```

Activitymain.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"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floatingActionButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:clickable="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_input_add" />
```

```
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/cart" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textStyle="bold"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="@+id/imageView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.09"
        app:layout_constraintStart_toStartOf="@+id/imageView2"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.94" />

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

ProductDatabase.kt

```
package com.jobin.grocery_app_android

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

@Database(entities = arrayOf(ProductModel::class), version = 1, exportSchema = false)
abstract class ProductDatabase : RoomDatabase(){

    abstract fun getProductDao(): ProductDao

    companion object{
```

```
        private var INSTANCE: ProductDatabase?=null

        fun getDatabase(context: Context): ProductDatabase {
            return INSTANCE ?: synchronized(this){
                var instance = Room.databaseBuilder(
                    context.applicationContext,
                    ProductDatabase::class.java,
                    "product_database"
                )
                    .build()
                INSTANCE = instance
                instance
            }
        }
    }

}
```

Product model.kt

```
package com.jobin.grocery_app_android

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

@Entity(tableName = "products")
class ProductModel(
    @ColumnInfo(name = "productName")
    val productName:String,
    @ColumnInfo(name = "productPrice")
    val productPrice:String
) {
    @PrimaryKey(autoGenerate = true)
    var id = 0
}
```

ProductRepository.kt

```
package com.jobin.grocery_app_android

import androidx.lifecycle.LiveData
import com.jobin.grocery_app_android.ProductDao
import com.jobin.grocery_app_android.ProductModel

class ProductRepository(private val productDao: ProductDao) {
```

```
val getAllProduct:LiveData<List<ProductModel>> = productDao.getAllProducts()

suspend fun insert(productModel: ProductModel){
    productDao.insert(productModel)
}

suspend fun update(productModel: ProductModel){
    productDao.update(productModel)
}

suspend fun delete(productModel: ProductModel){
    productDao.delete(productModel)
}
```