Project Report on

Grocery List Kotlin Android Application

**INDEX**

1. **INTRODUCTION**
2. **THEORITICAL ANALYSIS**
3. **FLOWCHARTS**
4. **RESULTS**
5. **ADVANTAGES & DISADVANTAGES**
6. **CONCLUSIONS**
7. **FUTURE SCOPE**
8. **BIBILOGRAPHY**
9. **APPENDIX**
10. **INTRODUCTION**

1.1 Overview:

Kotlin is a programming language used in android studio to develop android application.

In this project I am going to create an android app using Kotlin to create and store list of items going to buy. I named the app as “Grocery List”.

This app is user friendly and having a nice look as well as very useful.

In this project, I am using MVVM (Model View View Model) for architectural patterns, Room for database, Coroutines and RecyclerView to display the list of items. Before jumping to the project let’s understand these terms.

MVVM (Model View View Model) architecture in android is used to give structure to the project’s code and understand code easily. MVVM is an architectural design pattern in android. MVVM treat Activity classes and XML files as View. This design pattern completely separate UI from its logic. Here is an image to quickly understand MVVM.

ROOM DataBase Room persistence library is a database management library and it is used to store the data of apps like grocery item name, grocery item quantity.

Room is a cover layer on SQLite which helps to perform the operation on the database easily.

RecyclerView is a container and it is 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.

Coroutines are a 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.

1.2 Purpose:

As people have many things to do for all the day, they might forget the most important thing which is name of the items they have to buy. Sometimes it’s also seen that we make a list on paper and in some way that paper just disappear or destroy. All of these issues can be removed by using a digital list which is store in our smartphone. And so here is the app as a solution for this “Grocery List” app.

This app holds all the item with the quantity of that item need to buy. We can add as many as we want to add items and after buying, we can easily remove that item for the list.

**2 THEORITICAL ANALYSIS**

2.1 Block diagram:

The App is having a simple and easy layout to understand by user.

The app contains a button to add the items to list. Also, the list of items added having name and quantity. There is also a button within the row of items to delete the item if required or already/after purchased.

Fill item details

Show Item in list

Add item

Figure: Block diagram of app

2.2 Requirements:

For develop any project we need some resources and all these resources come under requirements. And for this project requirements are as follow:

System Requirement:

* Windows OS 10+
* Active Internet
* 8GB minimum RAM
* 50GB Disk Space
* X64 CPU

Software Requirement:

* Android Studio
* Android SDK
* Virtual Device/Physical Device

Device Requirement(For using app):

* Android 5.0+
* Minimum 2GB RAM
* 50+ MB empty storage

**3 FLOWCHARTS**

Diagram showing the control flow of the application

There is a button to add item to list.

New window open to fill the item detail.

Click on ***Add*** to add item to list.

Cancel

Item Quantity

Item Name

Add

Click on ***Cancel*** if not want to add.

Item Quantity

Item Name

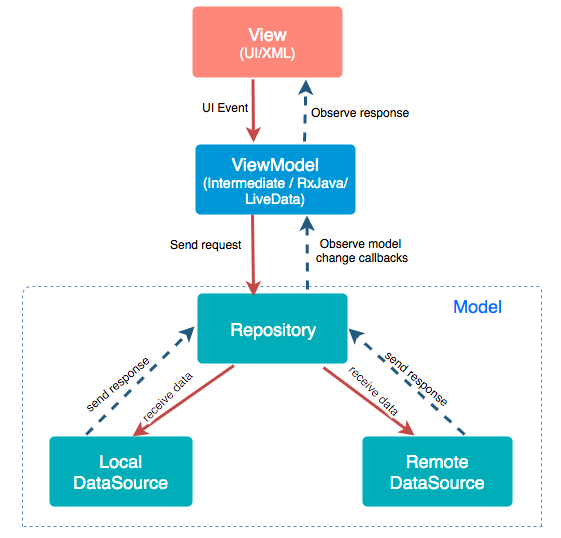
Delete

Item is added to list.

This is the whole structure of application.

Now,

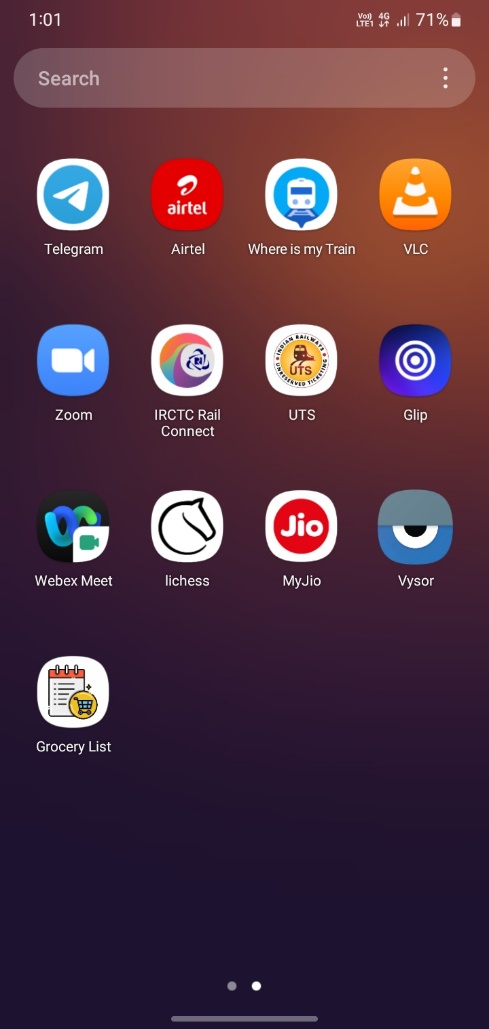
Also take a look how MVVM works:



**4 RESULTS**

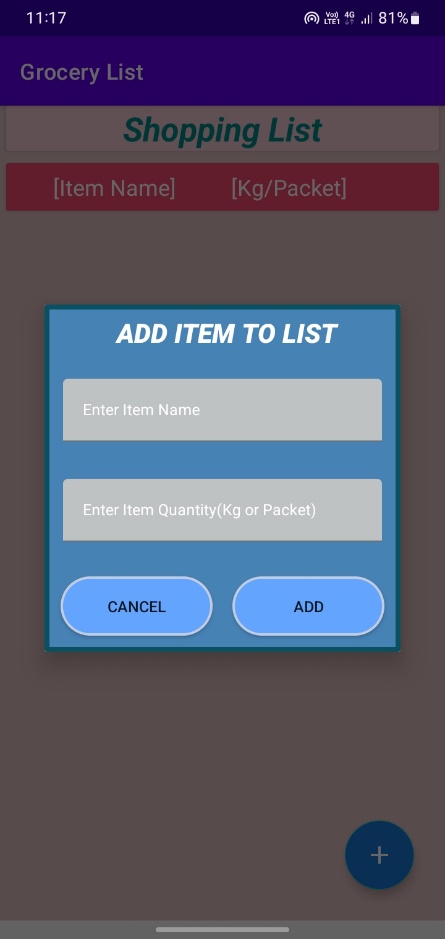
Now, let’s see the app after development how it looks and it’s working.

App Icon:

Grocery List

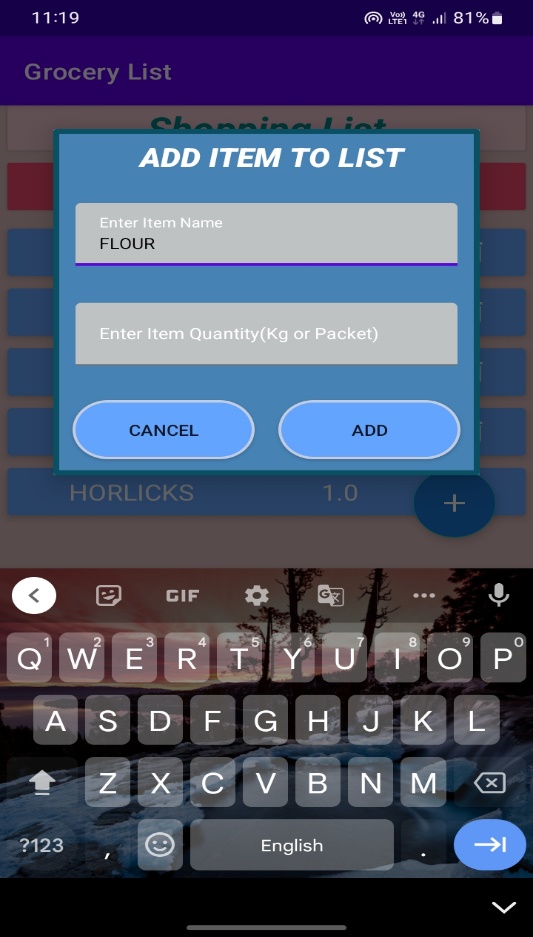
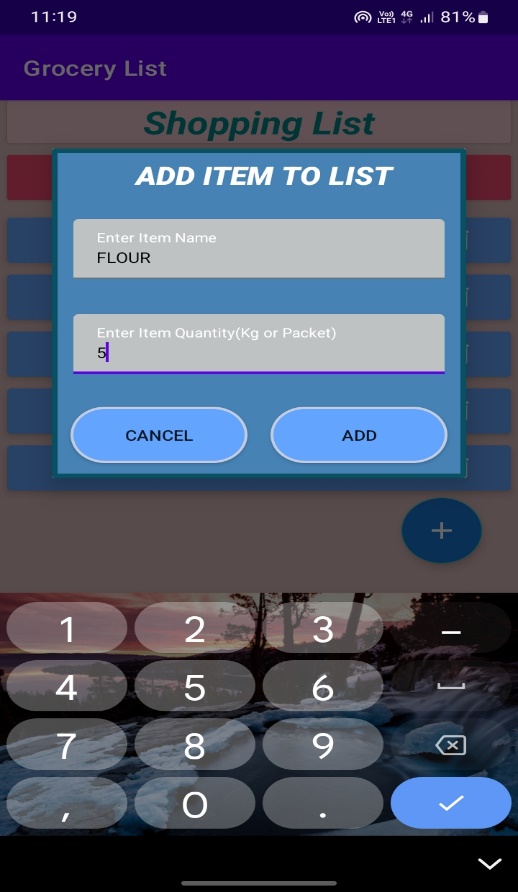
Screenshots of app:

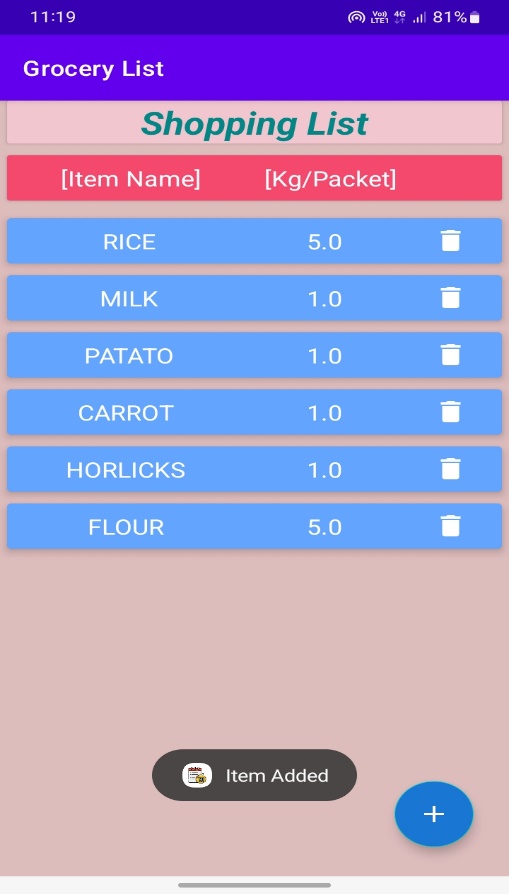
Home View Add item detail window

Now,

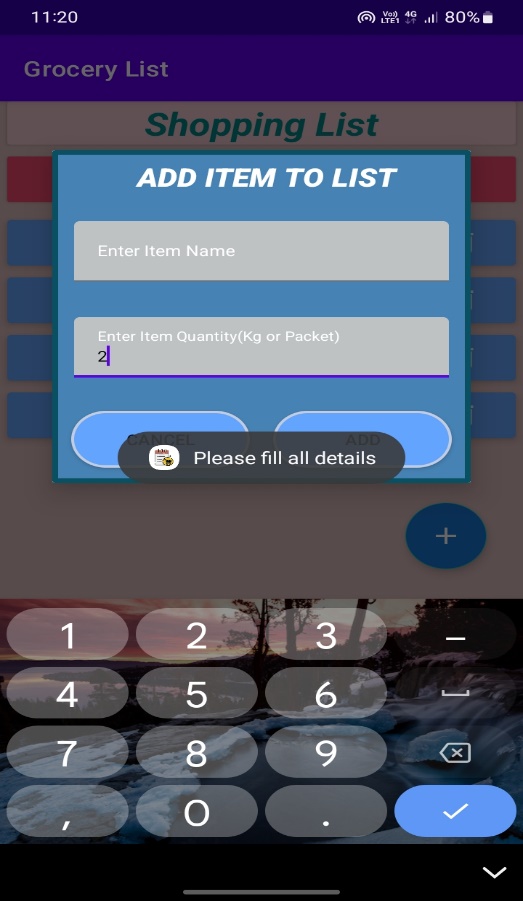
Process to add the items to the list as follows:

Fill item name Fill item quantity



Item Added to list

Deleted Item Incomplete detail

**5 ADVANTAGES & DISADVANTAGES**

In designing this project there are some advantages and some may be disadvantages.

Now, take a look on these advantages and disadvantages.

Advantages:

* Not need to remember item name to buy.
* It stores the quantity of items.
* Not fear of lost the list.
* Can any number of items.
* So simple just like paper and pen method.

Disadvantages:

* Same items maybe added twice.
* Edit is not allowed.

**6 CONCLUSIONS**

The app is very useful app in daily-to-daily life. Now, we talk about the application of this app

At first this app is used to make a list of items which is need to buy by user. This app contains the list of items with the quantity of items.

Having the user-friendly look and a less complected UI.

Not so many features are added to make the app simple to use. Anyone having very little knowledge to use a smart phone can use this app very smoothly.

**7 FUTURE SCOPE**

The is designed with so simplicity that it can be used by any level smartphone user.

This app may have many more features to make it better. The future scope of this app is high.

The can de upgraded to make it more flexible.

Take a look at some of upgrades that can be implemented to the existing app in future as follows:

1. Can be add edit feature.

We can a button just along with delete button so that if user want to update the value of item can be done easily. As of now no any option available.

1. Add item price.

As of now we only can add item name and quantity but it maybe good if we add feature of to add price of item (i.e., expected or known original).

1. Remove duplicate items.

We can add feature which can tell the use that the item already added in list, so remove occurrence of items more than once.

**8 BIBILOGRAPHY**

### 1. Android Basics in Kotlin from Google Developers.

2.Youtube channel GreekforGreeks.

**APPENDIX**

**Source Code:**

*MainActivity.kt*

package com.example.grocerylist  
  
import android.app.Dialog  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.widget.AppCompatButton  
import androidx.lifecycle.Observer  
import androidx.lifecycle.ViewModelProvider  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.android.material.floatingactionbutton.FloatingActionButton  
import com.google.android.material.snackbar.Snackbar  
import androidx.appcompat.app.AppCompatDelegate as AppCompatDelegate  
  
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,this)  
 itemRV.layoutManager = LinearLayoutManager(this)  
 itemRV.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**{** openDialog()  
 **}** }  
 fun openDialog(){  
 val dialog = Dialog(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 itemQuantityEdt = dialog.findViewById<EditText>(R.id.idEdtitemquantity)  
 cancelbtn.setOnClickListener **{** dialog.dismiss()  
 **}** addbtn.setOnClickListener **{** val itemname:String = itemEdt.text.toString()  
 val itemquantity:String = itemQuantityEdt.text.toString()  
 val qty : Double = itemquantity.toDouble()  
 if (itemname.isNotEmpty() && itemquantity.isNotEmpty()){  
 val items = GroceryItems(itemname,qty)  
 groceryViewModel.insert(items)  
 Toast.makeText(applicationContext,"Item Added",Toast.LENGTH\_SHORT).show()  
 groceryRVAdapter.notifyDataSetChanged()  
 dialog.dismiss()  
 }  
 else run **{** Toast.makeText(applicationContext,"Please fill all details",Toast.LENGTH\_SHORT).show()  
 **}  
  
 }** dialog.show()  
 }  
  
 override fun onItemClick(groceryItems: GroceryItems) {  
 groceryViewModel.delete(groceryItems)  
 groceryRVAdapter.notifyDataSetChanged()  
 Toast.makeText(applicationContext,"Item Deleted Successfully..",Toast.LENGTH\_SHORT).show()  
 }  
}

…………………………………………………………………………………………………………………………………………………………………………………….

*GroceryRVAdapder.kt*

package com.example.grocerylist  
  
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 deleteIV = itemView.findViewById<ImageView>(R.id.*idivdelete*)  
 }  
  
  
 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,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.deleteIV.setOnClickListener **{** groceryItemClickInterface.onItemClick(list.get(position))  
 **}** }  
  
 override fun getItemCount(): Int {  
 return list.size  
 }  
}

……………………………………………………………………………………………………………………………………………………

*GroceryViewModel.kt*

package com.example.grocerylist  
  
  
import androidx.lifecycle.ViewModel  
import kotlinx.coroutines.GlobalScope  
import kotlinx.coroutines.launch  
  
class GroceryViewModel(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()  
}

*………………………………………………………………………………………………………………………………………*

*GroceryDao.kt*

package com.example.grocerylist  
  
  
import androidx.lifecycle.LiveData  
import androidx.room.\*  
  
@Dao  
  
interface GroceryDao {  
 @Insert(onConflict = OnConflictStrategy.*REPLACE*)  
 fun insert(item: GroceryItems)  
  
 @Delete  
 fun delete(item: GroceryItems)  
  
 @Query("SELECT \* FROM Grocery\_items")  
 fun getAllGroceryItems(): LiveData<List<GroceryItems>>  
  
  
}

………………………………………………………………………………………………………………………………………………

*GroceryDatabase.kt*

package com.example.grocerylist  
  
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) **{** instance?: createDatabase(context).*also* **{** instance = **it  
 }  
  
 }** private fun createDatabase(context: Context) =  
 Room.databaseBuilder(  
 context.*applicationContext*,  
 GroceryDatabase::class.*java*,  
 "GroceryApp.db"  
 ).build()  
 }  
}

…………………………………………………………………………………………………………………………………………………………………………………….

*GroceryItems.kt*

package com.example.grocerylist  
  
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:Double,  
  
)  
{  
 @PrimaryKey(autoGenerate = true)  
 var id:Int?=null  
}

……………………………………………………………………………………………………………………….

*GroceryRespotory.kt*

package com.example.grocerylist  
  
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()  
}

…………………………………………………………………………………………………………………

*GroceryViewModeFactory.kt*

package com.example.grocerylist  
  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.ViewModelProvider  
  
class GroceryViewModelFactory(private val repository: GroceryRepository):ViewModelProvider.NewInstanceFactory() {  
 override fun <T : ViewModel> create(modelClass: Class<T>): T {  
 return GroceryViewModel(repository) as T  
 }  
}