**PROJECT**

**REPORT**

**Grocery App : An Android Application**

**MADE BY : YUNUS AHMED**

**SBID : SB20220235519**

**INTRODUCTION:**

Existing grocery list applications flourish, but because of their complicated user interface design, hardly any of them order to promote and nurture organic traffic. These apps are frequently used by individuals of varying ages, and therefore the user interface requires to be very simple and basic. Users to download these applications for convenience, consequently we should not really add too many security layers considering we know that convenience is compromised for security.

Coroutines, Model View ViewModel (MVVM) architectural patterns, Room for the database, and RecyclerView to display the list of elements are all utilized in this project.

This app main purpose is to make list of grocery items so that we do not forget anything. We can easily add or remove items from our list with the aid of the delete button.

**LITERATURE SURVEY:**

Before creating the application, I conducted a theoretical thorough inspection regarding to this issue. People frequently forget what they need to buy since they do not know the pricing of the products ahead of time, i.e. without visiting a grocery store or market. They try to make list on notepad/notes app which is unorganized and it is not easy to maintain that list.

I created my application for versions greater than Android KitKat in order to accommodate the vast majority of users with the help of this app a user can easily add or delete item to make his/her a maintained grocery list. However, users of prior or newer versions may use this software without losing any functionality, therefore it is not solely compatible with lollipop.

**THEORITICAL ANALYSIS:**

Technical requirement used to develop this app:

Hardware requirements:

* RAM: 16GB
* ROM: 20GB

Software requirements:

* Android studio
* Kotlin language
* Coroutines, Model View ViewModel (MVVM) architectural patterns, Room for the database, and RecyclerView

Technical requirement by Grocery app:

Hardware requirements:

* 1 GB RAM
* 2GHz dual core processor
* A touchscreen embedded in the device

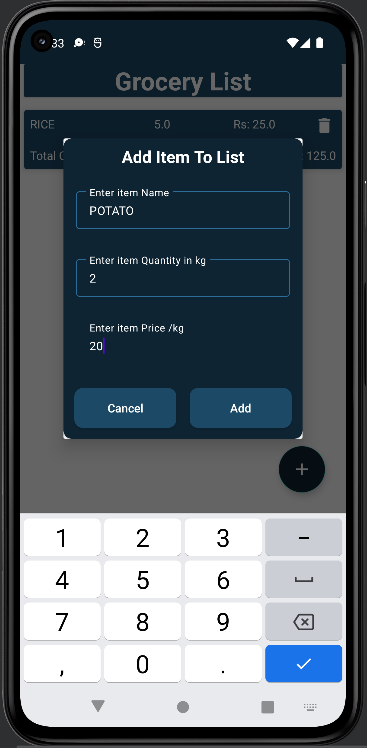
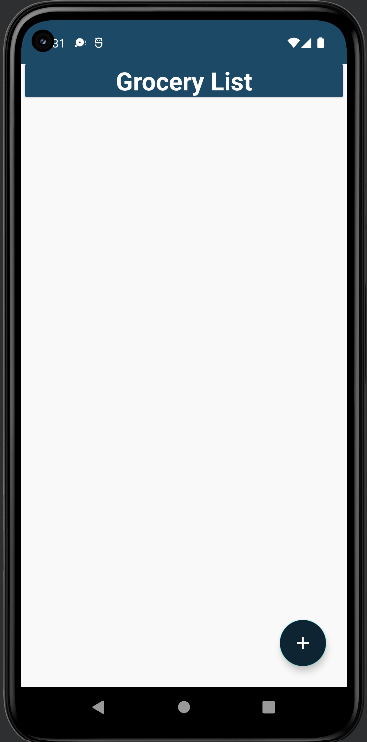
Software requirements:

* Android Lolipop or higher

**EXPERIMENTAL INVESTGATIONS:**

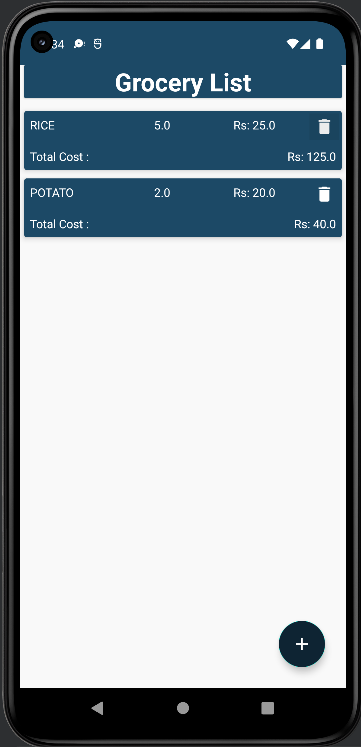
I examined a wide range of subjects both prior to and during the development of this Kotlin application to make this one as user-friendly as possible. I completed all of the programs offered on the provided website and completed the assessments.

**RESULT:**

(1) App front view (2) Panel to add item to list with details

(3) After addition of item showing all (4)After removing one item showing a pop-

Inputted details & total cost of item up message to users

****

**FLOWCHART:**

Enter the details of item

Deleting items from list

Add items to database

**ADVANTAGES & DISADVATAGES:**

Application advantages:

* User friendly
* Less complex
* More organized than any notepad/notes app

Application disadvantages:

* We cannot create different list for different types of items.
* There is no such thing to have a record of total expenditure for weekly or monthly expenses.

**Applications:**

* We can use this application to make list of our grocery items that we needed to buy.

**CONCLUSION:**

The main purpose of this application is to give user a service for which the user has to make a lot of effort. Through this application a lot of tasks are automated which saves the user a lot of time and they can increase their productivity by using this application, In a way that they can add multiple items lists, and they can keep track of what they are buying by keeping an eye on the list.

**FUTURE SCOPE:**

When new Android versions are introduced, it will be upgraded to accommodate the advanced functionality that will be contained in upcoming versions. Furthermore, this application will be developed on iOS, permitting it to be accessed by more individuals. In future we can also add login page so that user can access this app from anywhere , suggestion of items according to previous data for ease .

**BIBLIOGRAPHY:**

SmartInternz resources on android development using Kotlin language

* Google.com
* Youtube.com

**APPENDIX**

**(Source code)**

**GroceryDao.kt**

import androidx.lifecycle.LiveData

import androidx.room.\*

// This class is used to create

// function for database.

@Dao

interface GroceryDao {

// Insert function is used to

// insert data in database.

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insert(item: GroceryItems)

// Delete function is used to

// delete data in database.

@Delete

suspend fun delete(item: GroceryItems)

// getAllGroceryItems function is used to get

// all the data of database.

@Query("SELECT \* FROM Grocery\_items")

fun getAllGroceryItems(): LiveData<List<GroceryItems>>

}

**GroceryDatabase.kt**

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**

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,

@ColumnInfo(name = "itemPrice")

var itemPrice:Double,

)

{

@PrimaryKey(autoGenerate = true)

var id:Int?=null

}

**GroceryItems.kt**

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()

}

**GroceryViewModel.kt**

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()

}

**GroceryViewModelFactory.kt**

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

}

}

**GroceryRVAdapter.kt**

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 totalTV = itemView.findViewById<TextView>(R.id.idtvtotalamount)

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.rateTV.text = "Rs: "+list.get(position).itemPrice.toString()

val itemTotal :Double = list.get(position).itemQuantity \* list.get(position).itemPrice

holder.totalTV.text = "Rs: "+itemTotal.toString()

holder.deleteIV.setOnClickListener {

groceryItemClickInterface.onItemClick(list.get(position))

}

}

override fun getItemCount(): Int {

return list.size

}

}

**MainActivity.kt**

import android.annotation.SuppressLint

import android.app.Dialog

import android.os.Bundle

import android.widget.EditText

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.appcompat.widget.AppCompatButton

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 itemRV:RecyclerView

lateinit var addFAB:FloatingActionButton

lateinit var list: List<GroceryItems>

lateinit var groceryRVAdapter: GroceryRVAdapter

lateinit var groceryViewModel: GroceryViewModel

@SuppressLint("NotifyDataSetChanged")

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) {

groceryRVAdapter.list = it

groceryRVAdapter.notifyDataSetChanged()

}

addFAB.setOnClickListener{

openDialog()

}

}

@SuppressLint("NotifyDataSetChanged")

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 itemPriceEdt = dialog.findViewById<EditText>(R.id.idEdtitemprice)

val itemQuantityEdt = dialog.findViewById<EditText>(R.id.idEdtitemquantity)

cancelbtn.setOnClickListener {

dialog.dismiss()

}

addbtn.setOnClickListener {

val itemname:String = itemEdt.text.toString()

val itemprice:String = itemPriceEdt.text.toString()

val itemquantity:String = itemQuantityEdt.text.toString()

if (itemname.isNotEmpty() && itemprice.isNotEmpty() && itemquantity.isNotEmpty()){

val qty : Double = itemquantity.toDouble()

val pr : Double = itemprice.toDouble()

val items = GroceryItems(itemname,qty,pr)

groceryViewModel.insert(items)

Toast.makeText(applicationContext,"Item Added Successfully..",Toast.LENGTH\_SHORT).show()

groceryRVAdapter.notifyDataSetChanged()

dialog.dismiss()

}

else

{

Toast.makeText(applicationContext,"Please enter all the details",Toast.LENGTH\_SHORT).show()

}

}

dialog.show()

}

@SuppressLint("NotifyDataSetChanged")

override fun onItemClick(groceryItems: GroceryItems) {

groceryViewModel.delete(groceryItems)

groceryRVAdapter.notifyDataSetChanged()

Toast.makeText(applicationContext,"Item Deleted Successfully..",Toast.LENGTH\_SHORT).show()

}

}