ANDROID DEVELOPMENT PROJECT

GOOGLE SUPPORTED FDP AND VIRTUAL INTERNSHIP PROGRAM

*In collaboration with **AICTE***

SMARTBRIDGE

PROJECT REPORT

Name of the student: BHARATH VISHNU C J
Application Id: SPS_APL_20220106823
SB Id: SB20220243270
Email: bharathvishnu2522003@gmail.com

1. INTRODUCTION

1.1 OVERVIEW

As we cannot remember everything, we frequently forget the things that we need to buy. However, with the guidance of this app, we can make a list of the groceries items that we need to buy so that we do not forget anything. This app performs in such a way that it ask us to enter the item name, item price and item quantity. So that it calculates the total price for that item and displays to us. After purchasing we can delete the particular item.

1.2 PURPOSE

So This app is created for the purpose that it remembers grocery items which we need to purchase even we fail to remember. We can add grocery items ,quantity of that grocery item and a price of that item. After that whenever we open the app we can able to access the items names that we need to purchase.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The main existing problem is that we cannot remember the items that we need to purchase. The existing method was people generally write the name of the things that they need to purchase in a paper and carry that paper every time when they going to purchase.

2.2 PROPOSED SOLUTION

To overcome that Drawback this system is made in such a way that carrying of paper can be avoided. This app asks for item name, item Quantity and item price so that with the help of this app we can even refer the item name, item Quantity also even after longer period of time.

3. THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM

Diagrammatic overview of the project :

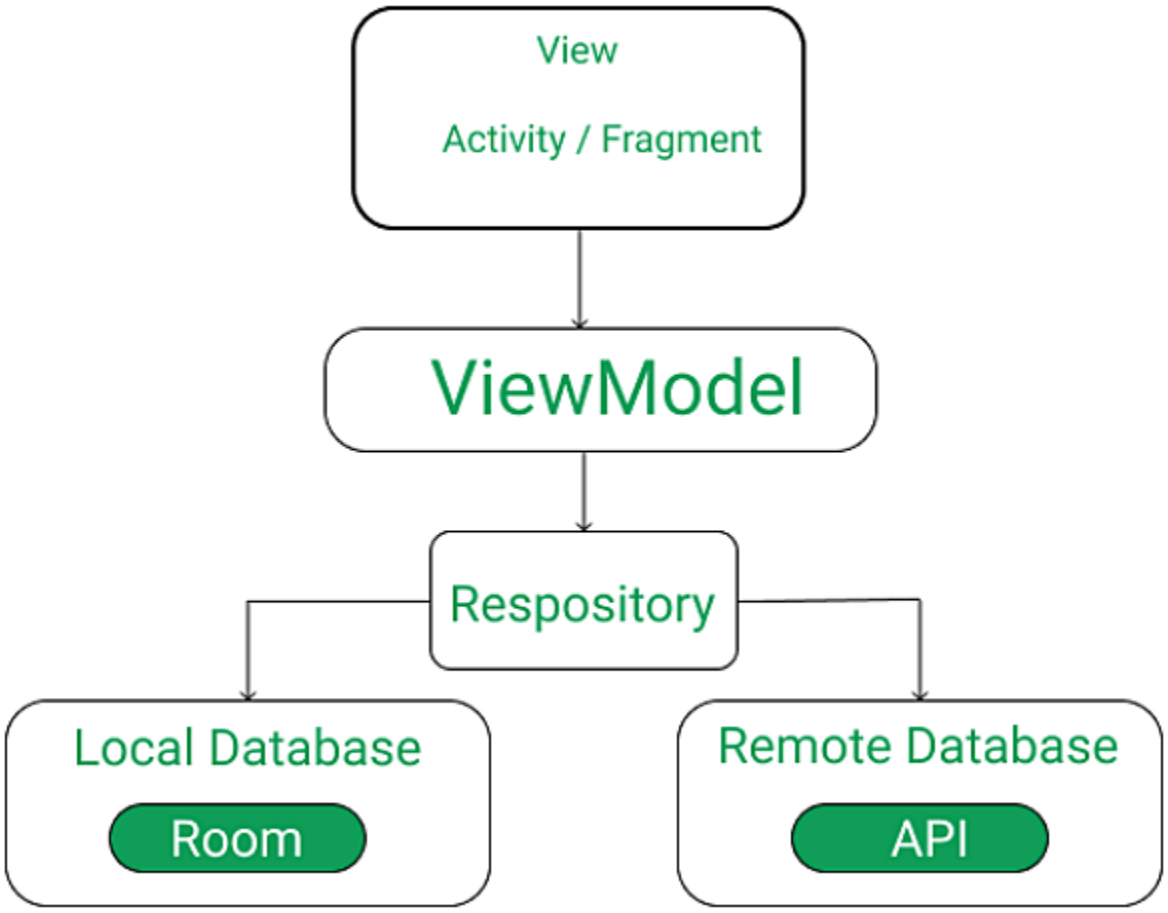

Fig 1.Block Diagram

3.2 HARDWARE / SOFTWARE DESIGNING

The requirement for the project is a Mobile device irrespective of the Android versions to run this project. The requirement for this project to built is a Laptop with Android studio installed with it and the programming language used is Kotlin for building this application. This project is developed using MVVM (Model View ViewModel) for architectural patterns, Room for database, Coroutines and RecyclerView to display the list of items.

4. EXPERIMENTAL INVESTIGATIONS

This project is developed with the use of MVVM (Model View ViewModel) for architectural patterns, Room for database, Coroutines and RecyclerView to display the list of items which are analyzed while developing this project.

5. FLOWCHART

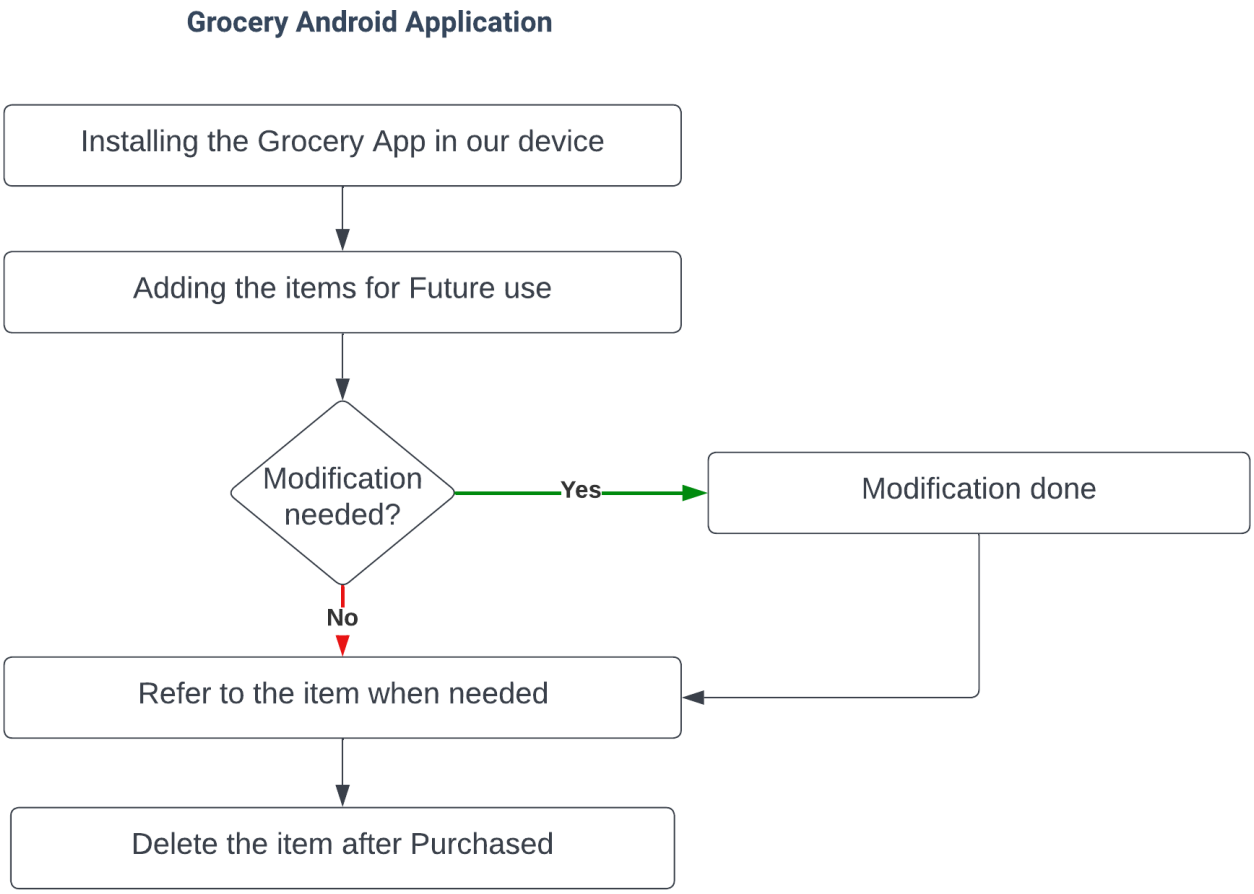

Fig 2. Flowchart

6. RESULT

1) App icon :

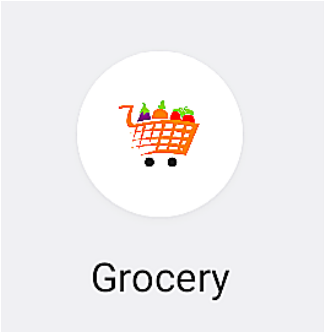

Fig 3. App icon

2) Grocery App in the Home Screen

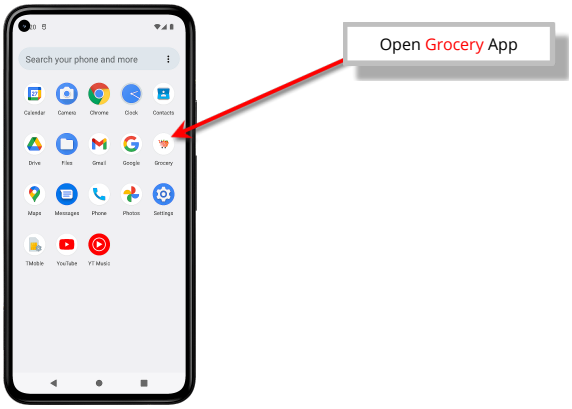

Fig 4. Basic UI Screen

3) On App Startup, This, will be the first screen on initial setup…

Fig 5. App Startup

This will be Empty, since we haven't added any items yet.

4) Click on Add FAB (i.e., Floating Action Button) on the bottom right to add items to the app and fill details in the text fields

Fig 6. Add FAB Button

5) We have to fill the details of item as shown in the picture

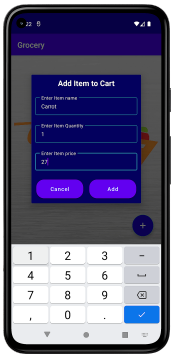

Fig 7. Add Item Fragent

6) After all fields are filled, clicking on Add Button, it saves the data in the app and show it in our Main screen

Fig 8. Main Screen(After Adding items)

7) If we click on the delete icon 🗑, the item will be deleted from the app's data.

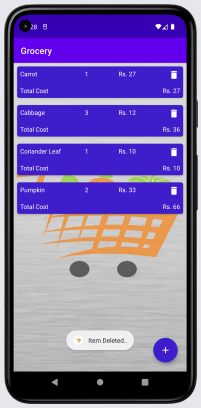

Fig 9. Main Screen(After Deleting items)

8) There will be a Toast message will be shown at the bottom side when the item is deleted or inserted in the app.

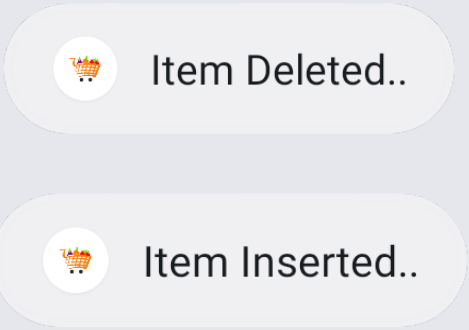

Fig 10. Toast Message

9) Final Updated App Preview After Adding items .

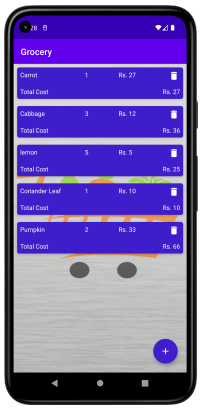

Fig 11. Main Screen(Updated Preview)

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The main advantage of this application is we can remember the things name and item quantity anytime and anywhere even it displays with the price quantiy also.
- Since most of the people are busy with their work, for them this application will be useful.

DISADVANTAGES:

- The main drawback for this application is there is need of carrying mobile phones everytime.Nowadays most of them having mobile phones with them daily in that case this disadvantage will not be that much effective.

8. APPLICATIONS

This application will be helpful in real time basis itself. Since the people are working today, and in their work tension there may be chances of forgetting the items they need to buy. In that case this application is helpful for them.

9. CONCLUSION

As we frequently forget the things that we need to buy, this application helps us in that case by taking the item name, item price and quantity and displays them to the user with the item name, quantity and total price needed for purchase. This application is surely helpful for the people who are working.

10. FUTURE SCOPE

The future scope of this application is to built a voice assistant inside that application which will read the item name typed in that list which will be very helpful for a people who may not able to read or people who cannot able to see.

11. BIBILOGRAPHY

Gold, Jack. "Enabling Mobile Business Applications: A Strategic Approach." SearchMobileComputing.com. Web. 25 Oct. 2011 http://searchmobilecomputing.techtarget.com/feature/Enabling-mobile-business-applications-A-strategic-approach

Khanna, Anuj. "The Future Of Mobile Application Storefronts." Upload & Share PowerPoint Presentations and Documents. Wireless Expertise Ltd, 2009. Web. 3 Nov. 2011. http://www.slideshare.net/vcbothra/the-future-of-mobile-application-storefronts

Laudon, Kenneth C., and Jane P Laudon. Management Information Systems: Managing the Digital Firm. Boston: Prentice Hall, 2012.

Pettey, Chrisy, and Holly Stevens. "Gartner Identifies the Top 10 Consumer Mobile Applications for 2012." Technology Research | Gartner Inc. Gartner, Inc., 18 Nov. 2009. Web. 2 Nov. 2011. http://www.gartner.com/it/page.jsp?id=1230413

Prince, Brian. "Mobile Application Developers Face Security Challenges - Security - News & Reviews - EWeek.com." Technology News, Tech Product Reviews, Research and Enterprise Analysis - News & Reviews - EWeek.com. Web. 27 Nov. 2011. http://www.eweek.com/c/a/Security/Mobile-Application-Developers-Face-Security-Challenges-822505/

Wysopal, Chris. "The Challenges Of Developing Secure Mobile Applications." Application Security | Veracode. Web. 27 Nov. 2011. http://www.veracode.com/blog/2009/07/the-challenges-of-developing-secure-mobile-applications/

Yee, Andre. "The Future of Mobile Applications - It's Not the iPhone - Cloud Talk." An SOA, BPM, Decision Management and Cloud Computing Guide for the Enterprise Community. Ebiz, 6 Apr. 2006. Web. 3 Nov. 2011. http://www.ebizq.net/blogs/cloudtalk/2009/04/the_future_of_mobile_applicati.php

APPENDIX

A. Source Code :

GroceryDao.kt

```
package com.example.groceryapplication

import androidx.lifecycle.LiveData
import androidx.room.*

@Dao
interface GroceryDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(item: GroceryItems)

    @Delete
    suspend fun delete(item: GroceryItems)

    @Query("SELECT * FROM grocerry_items")
    fun getAllGroceryItems(): LiveData<List<GroceryItems>>
}
```

GroceryDatabase.kt

```
package com.example.groceryapplication

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,
                "Grocery.db"
            ).build()
    }
}
```

GroceryItems.kt

```
package com.example.groceryapplication

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

@Entity(tableName = "grocerry_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
}
```

GroceryRepository.kt

```
package com.example.groceryapplication

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.kt

```
package com.example.groceryapplication

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.idTVToatalAmt)
        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,
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
    }
}
```

GroceryViewModal.kt

```
package com.example.groceryapplication

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()

}
```

GroceryViewModalFactory.kt

```
package com.example.groceryapplication

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
    }
}
```

MainActivity.kt

```
package com.example.groceryapplication

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.Observer
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 groceryRepository = GroceryRepository(GroceryDatabase(this))
        val factory = GroceryViewModalFactory(groceryRepository)
        groceryViewModal = ViewModelProvider(this,
factory).get(GroceryViewModal::class.java)
        groceryViewModal.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<Button>(R.id.idBtnCancel)
        val addBtn = dialog.findViewById<Button>(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()
            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()
    }
}
```