

EXPENSE MANAGER APPLICATION

Submitted in partial fulfilment of the requirements for the award of
Bachelor degree in Computer Science

By

Suresh Kumar R (Reg.No:38290093)

K.Muralidharan (Reg.No:38290109)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Suresh Kumar R (Reg.No:38290093)** and **K.Muralidharan (Reg.No:38290109)** who carried out the project entitled "**EXPENSE MANAGER APPLICATION**" under our supervision from NOVEMBER 2020 to APRIL 2021.

INTERNAL GUIDE

Dr. J.ALBERT MAYAN M.E., Ph.D.

HEAD OF THE DEPARTMENT

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **Suresh Kumar R (Reg.No:38290093)** and **K.Muralidharan (Reg.No: 38290109)** hereby declare that the Project Report entitled “**EXPENSE MANAGER APPLICATION**” done by me under the guidance of **Dr.J.ALBERT MAYAN M.E., Ph.D.** is submitted in partial fulfilment of the requirements for the award of **Bachelor** degree in **Computer Science**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGMENT

We are pleased to acknowledge my sincere thanks to the Board of **MANAGEMENT OF SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean**, School of Computing and **Dr. L. LAKSHMANAN M.E., Ph.D.**, and **Dr. VIGNESHWARI M.E., Ph.D., Head of the Department, Department of Computer Science And Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr J. ALBERT MAYAN M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

Mobile applications stood top among usability and user convenience. Many applications are available in the market to manage personal and group expenses. Not many applications provide a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses, top investment options, view of current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories. The proposed application would eliminate sticky notes, spreadsheets and ledger that cause confusions, data inconsistency problems while recording and splitting of expenses, time spending on searching authenticated and verified news and also to never miss out on going market offers or discounts. With our application users can manage their expenses and budget planning more effectively.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
1.	INTRODUCTION	
1.1	AIM	1
1.2	SCOPE	1
1.3	OBJECTIVES	2
2.	PROJECT PLANS AND REQUIREMENTS	
	PLANS	3
	REQUIREMENTS	3
	DESIGNS	3
	OVERVIEW OF THE PLATFORM	4
3.	SYSTEM IMPLEMENTATION	
	ARCHITECTURE DESIGN	7
	DESIGN USER INTERFACE	8
	Layouts Designs	8
4.	RESULTS AND DISCUSSION	
	RESULT	12
	DISCUSSION	12
5.	CONCLUSION AND FUTURE WORK	
	CONCLUSION	18
	FUTURE WORK	18
	REFERENCES	21
	APPENDIX	
	A. SOURCE CODE	22
	B. PUBLICATION	27

LIST OF FIGURES

REF. No.	TITLE	PAGE No.
3.1	Architecture Design	7
	UPI payment layout / architecture	8
	Split Bill layout / architecture	9
	Expense Tracer layout / architecture	10
	Investment Consultant layout / architecture	10
	Stock Market layout / architecture	11
	Finance News layout / architecture	11
	Home Page	12
	Grant Permission	13
	Transaction History	13
	Investment Consultant	14
	Set Senders UPI	14
	Set Receivers UPI	15
	Pay Using Any PUI App	15
	Split Bills	16
	Coupons / Offers	17
	Trends	19
	Estimations	19

LIST OF ABBREVIATIONS

- APP – Application
- UPI – Unified Payment Interface
- JDK – Java Development Kit
- IDE – Integrated Development Kit
- SDK – Software Development Kit
- UI – User Interface
- UX – User Experience

CHAPTER-1

INTRODUCTION

Making day-to-day decisions on spending money is one of the biggest challenges consumers face in keeping their financial lives in order. Adding to this challenge is the fact that about one-third of all consumers live paycheck-to-paycheck, increasing the possibility that consumers may face negative consequences from overspending. Due to unanticipated spending, some consumers may find themselves short on cash and resort to dipping into savings; taking out short-term, high-interest loans; or using credit cards to try to address the difference between what they earn and what they spend.

AIM

In this world of growing technologies everything is digitalized. With large number of money transaction it is difficult for user to keep track of all your transaction. The main aim of this project is to manage personal and group expenses. Not many applications provides a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses, top investment options , view stock market , read authenticated news and grab the best ongoing offers in the market in popular categories . The application can be used to collect samples of data related to user's expenses with permissions and use those sample data as parameters to evaluate patterns of spending.

SCOPE

Some of the project's scope are:

- keep track of your transactions
- keep track of money lent or borrowed
- suggest you best investment options
- offer you best coupons in popular categories

- get latest finance news on your device
- To view the stock market
- Easy to use Interface

OBJECTIVES

The objectives of the system are as follows:

- Design an android based application which can keep track of all transactions,
And to keep track of money lent or borrowed.
- It should also provide knowledge of best investment options.
- Get the latest finance news and stock market details on your device.

CHAPTER 2

PROJECT PLANS AND REQUIREMENTS

PLANS

Planning is important before starting the project. It ensures that there's a proper plan for executing on strategic goals. The plans created before starting the project helps to manage time, cost, quality, change, risk and issues.

The plans for this project are as follows:

1. Analysing the project's objectives
2. Collection of requirements for the project
3. Designing the Application Window Layouts
4. Coding using Kotlin, Java
5. Testing

REQUIREMENTS

Hardware Requirements

- Ram – Above 2gb
- Internal storage – 30mb(file size)

Software Requirements

- Operating system – Android (version above 5 – LOLLIPOP)

Permission requirement

- SMS – Read permission
- Access to Internet permission

DESIGN

For the purposes of this project, "design" means the visual design of the application as well as the implementation details that underlie it. The design should describe what the finished application will look like and how a person will make it work. A vital part of the design will be a set of transition from one Window to another. The

final product of the design phase is a connected Window that contains different structured with different operations.

The Main Window should contain these components:

1. UPI PAYMENT
2. INVESTMENT OPTION
3. SPLIT BILL
4. AVAILABLE OFFERS/COUPONS
5. ALL TRANSACTION DETAILS

OVERVIEW OF THE PLATFORM

ANDROID STUDIO:

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains IntelliJ IDE software and designed specifically for Android development. It Java integrated development environment for software, and incorporates its code editing and developer tools. It is available for download on Windows, macOS and Linux based operating systems.

JAVA:

Java is Platform Independent. Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. It implements a strong security model, which prevents compiled Java programs from illicitly accessing resources on the system where they execute or on the network. Popular World-Wide Web browsers, as well as some World-Wide Web servers and other systems implement Java interpreters. These are used to display interactive user interfaces, and to script behaviour on these systems.

KOTLIN:

Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of its standard library depends on the Java Class Library, but type inference allows its syntax to be more concise.

Kotlin is a new open source programming language like Java, JavaScript, etc. It is a high level strongly statically typed language that combines functional and technical part in a same place. Currently, Kotlin targets Java and JavaScript. It runs on JVM.

Kotlin is influenced by other programming languages such as Java, Scala, Groovy, etc. The syntax of Kotlin may not be exactly similar to JAVA, however, internally Kotlin is reliant on the existing Java Class library to produce wonderful results for the programmers. Kotlin provides interoperability, code safety, and clarity to the developers around the world.

Kotlin is a programming language and has its own architecture to allocate memory and produce a quality output to the end user. Following are the different scenarios where Kotlin compiler will work differently, whenever it is targeting different other kind of languages such as Java and JavaScript.

Kotlin compiler creates a byte code and that byte code can run on the JVM, which is exactly equal to the byte code generated by the Java **.class** file. Whenever two byte coded file runs on the JVM, they can communicate with each other and this is how an interoperable feature is established in Kotlin for Java.

XML:

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. So what exactly is a markup language? Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

CHAPTER 3

SYSTEM IMPLEMENTATION

ARCHITECTURE DESIGN

This flowchart represents the workflow of the project. This diagrammatic Representation illustrates a solution to the given problem. This helps in systematic

Designing of the project

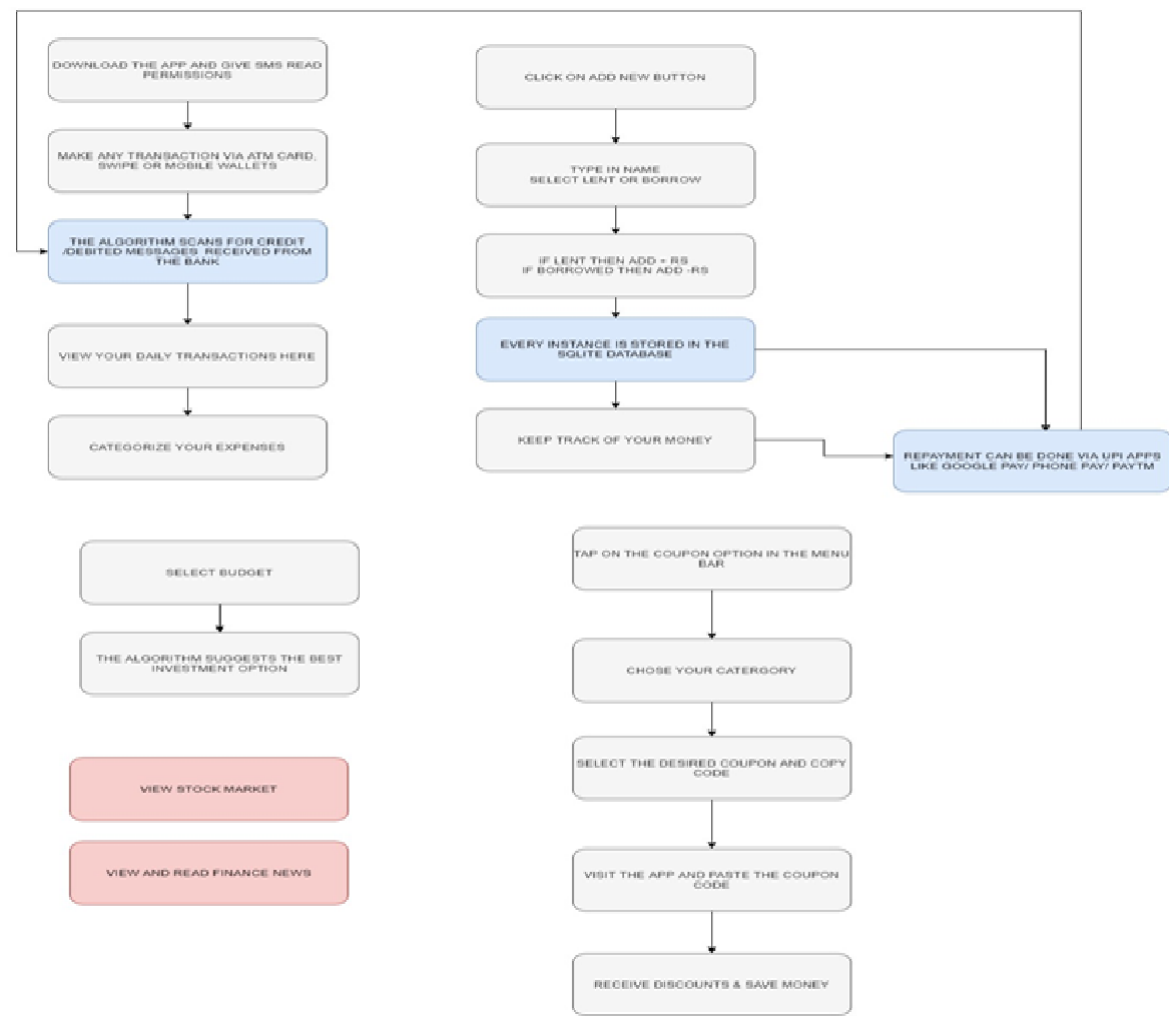


Fig 3.1 ARCHITECTURE DESIGN

Design User Interface

The application interface should be minimal and purposeful. The UI should enhance the user experience.

LAYOUTS DESIGNSUPI

PAYMENT

This feature is used for payment purposes. The UPI payment is designed and developed by the government of India. Integrated into the app this app is a gateway for mobile wallets like Google pay, phonepe, paytm, amazon pay and others. The user needs to be a registered UPI user with UPI PIN set in any of the mobile wallets.

The first layout contains two edit text, one button widgets to add senders UPI ID and name. And the second layout contain three edit text and one button for receiver UPI ID, amount , note and pay button.

Blueprint of UPI PAYMENT LAYOUT

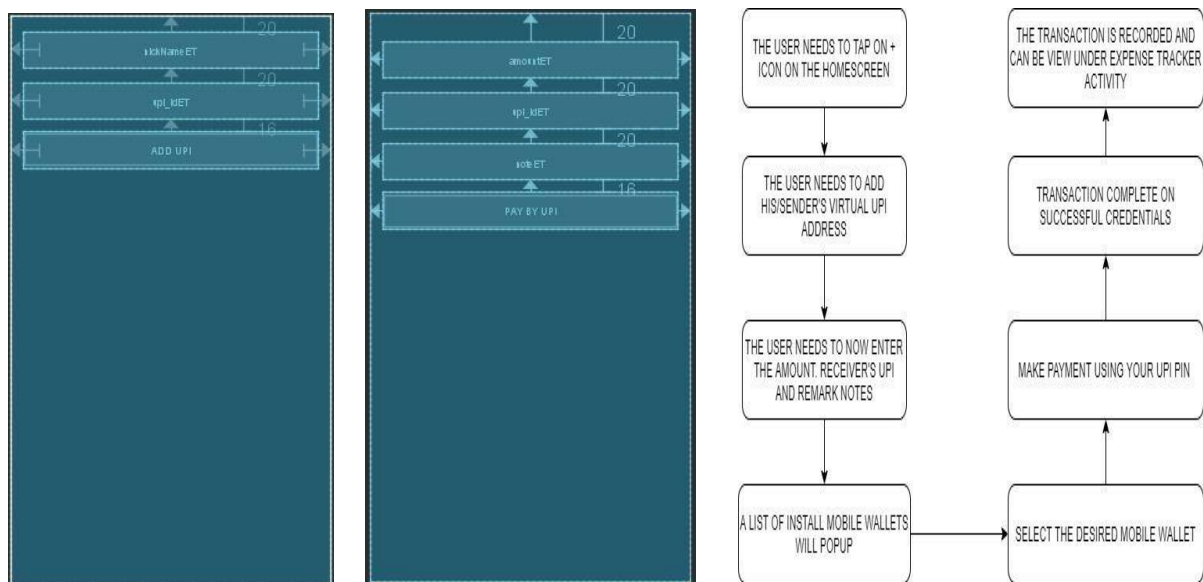


Fig 3.2.1.1 BLUEPRINT OF UPI PAYMENT LAYOUT / ARCHITECTURE

SPLIT BILL

This feature is used to split bills among friends and groups i.e. any record of money lent or borrowed is stored here. The user needs to tap on add menu and then the user needs to enter the name and amount. The user needs to add “+” rs if money is lent and “-” rs if money is borrowed .Each transaction is stored in the database. All the previous records can also be viewed from here.

Blueprint of SPLIT BILL LAYOUT

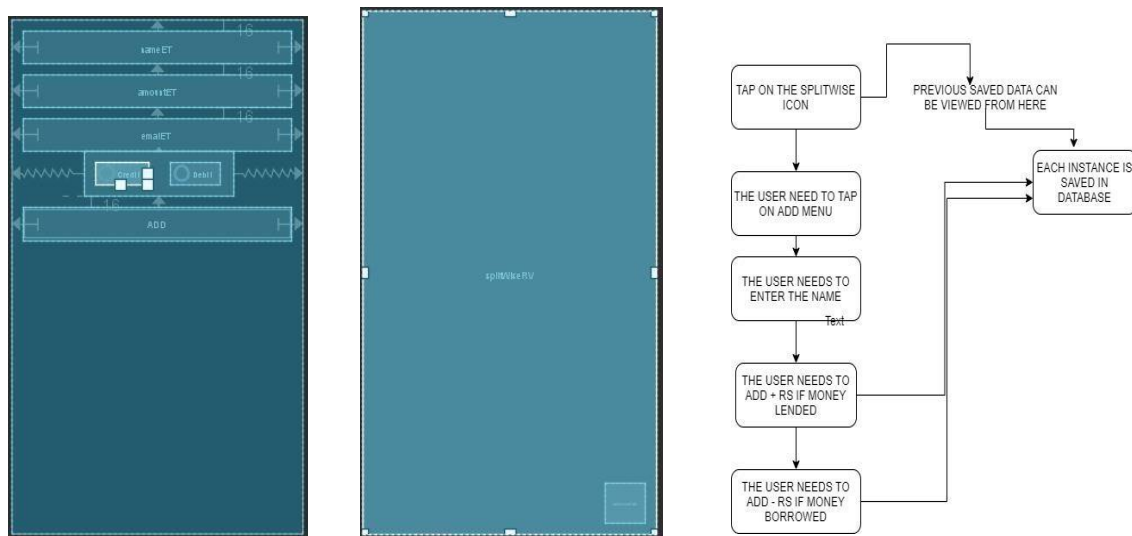


Fig 3.2.1.2 BLUEPRINT OF SPLIT BILL LAYOUT / ARCHITECTURE

EXPENSE TRACKER

This is the first phase after installing the application. We need to give sms read permission to the app since the whole idea of the app revolves around the transactions made online for which you get a message after any transaction done.

Blueprint of EXPENSE TRACKER

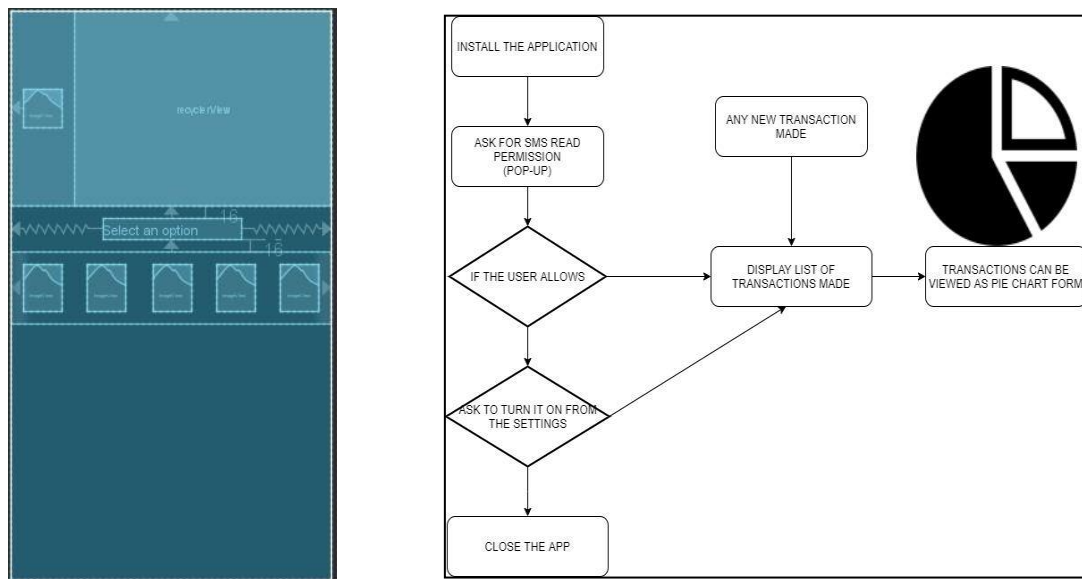


Fig 3.2.1.3 BLUEPRINT OF EXPENSE TRACKER LAYOUT / ARCHITECTURE

INVESTMENT CONSULTANT

This is the second feature of the app. It provides the knowledge for the best investment options available in the market.

Blueprint of INVESTMENT CONSULTANT

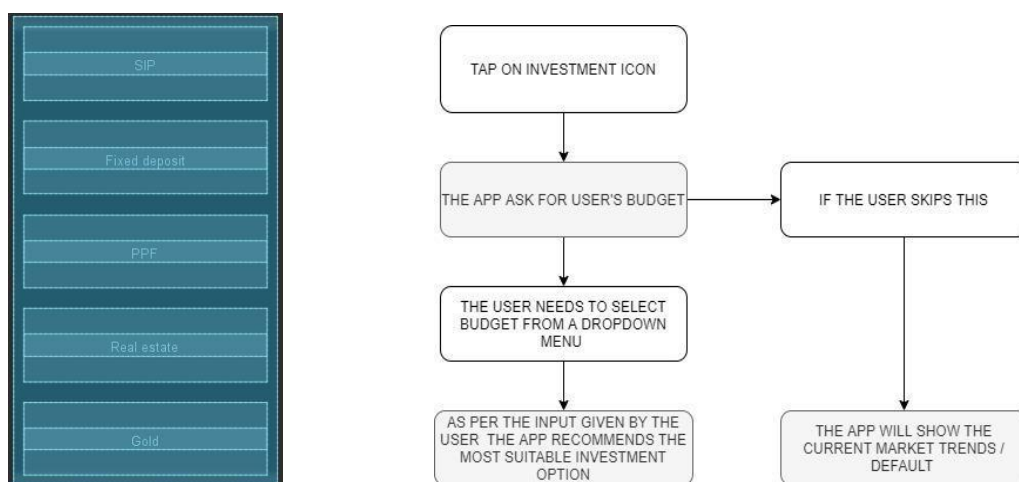


Fig 3.2.1.4 BLUEPRINT OF INVESTMENT CONSULTANT LAYOUT / ARCHITECTURE

STOCK MARKET

This feature helps the user to view the latest stock market using a API which is viewed inside a web view. The user can visit the official trading place and make transactions. Payment can be done with the UPI feature present within the app.

Blueprint of STOCK MARKET

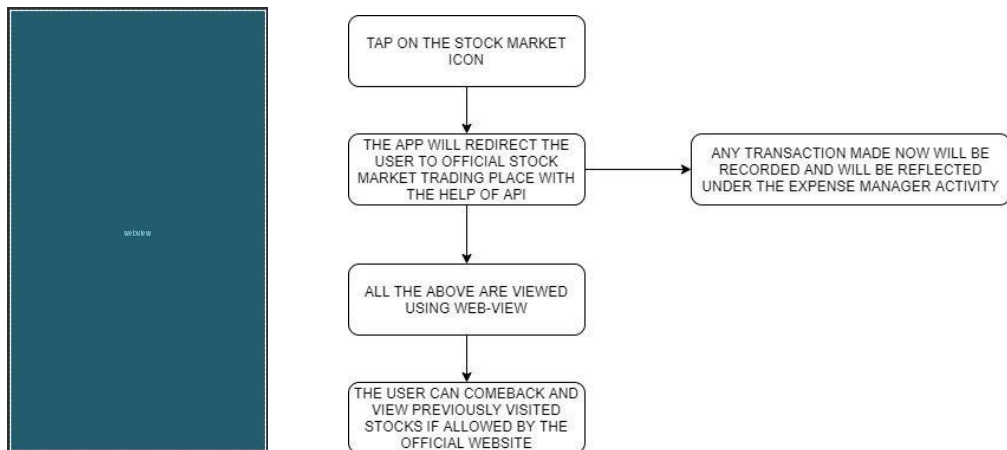


Fig 3.2.1.5 BLUEPRINT OF STOCK MARKET LAYOUT / ARCHITECTURE

FINANCE NEWS

This features provides the user with authenticated finance news. The news is fetch from various financial news websites using the help of an trusted API.

Blueprint of FINANCE NEWS

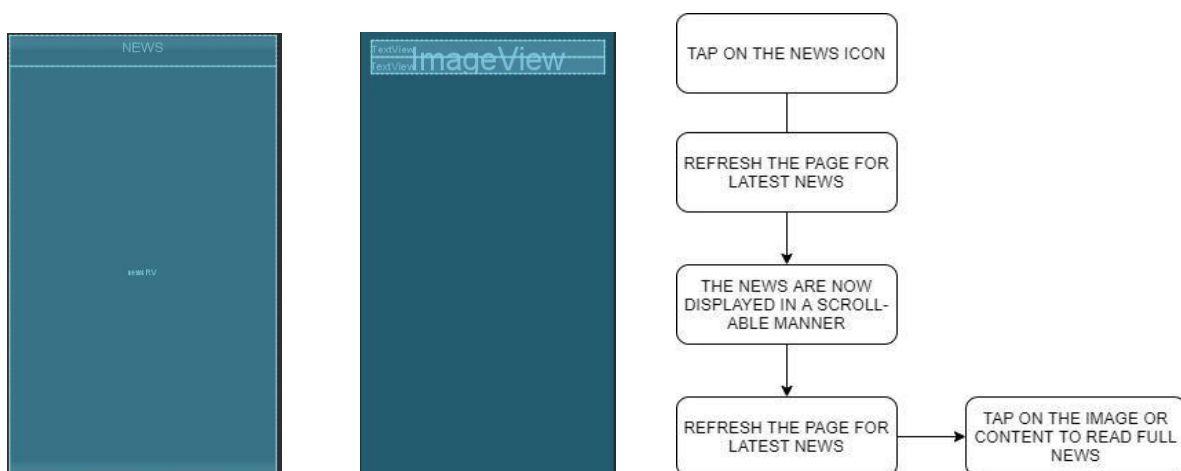


Fig 3.2.1.6 BLUEPRINT OF FINANCE NEWS LAYOUT / ARCHITECTURE

CHAPTER 4

RESULTS AND DISCUSSION

RESULT

The Finance Manager Application when first install's it will ask permission to read SMS, which will allow the application to retrieve the transaction SMS received by user on every online transaction and card payment so that the application can keep track of user's transaction. The application also provides the function of splitting bills. This application will allow the user to view Finance related news, keep track on Stock market, latest available offers and coupons, and can pay using UPI wallets provided that user should have UPI account. This application is the solution to all finance related problems.

DISCUSSION

MODULE 1

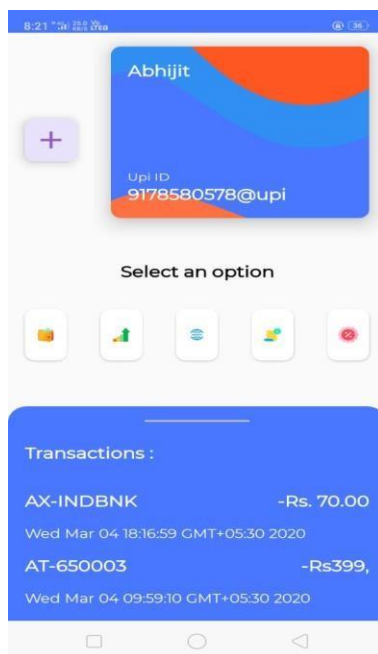


FIG 4.2.1 HOME PAGE

This module is the Home page of the application it consist of all the features present in the apps.

MODULE 2

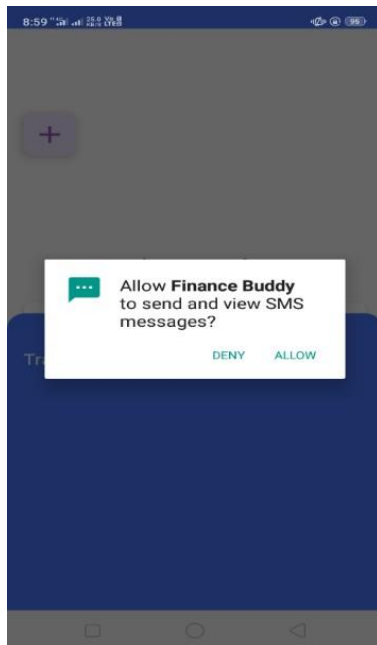


FIG 4.2.2 GRANT PERMISSION

When the application will be installed and run for first time it will ask permission to send and view SMS messages. If permission is not granted then application will not run.

MODULE 3



FIG 4.2.3 TRANSICTION HISTORY DETAILS

This module contains the details of all users' transaction. The details it contains are amount of money received or send by the user, these are sorted according to dates the payment are made.

MODULE 4

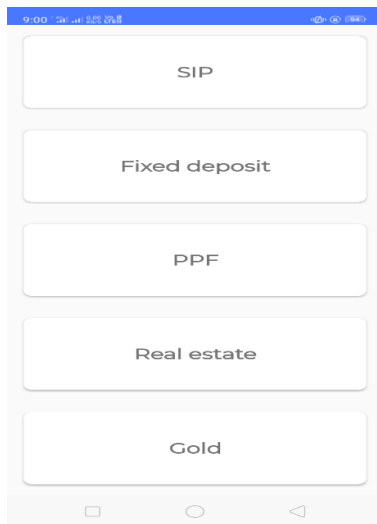


FIG 4.2.4 INVESTMENT CONSULTANT

This module consist of all investment options for users to know about where to invest their money.

MODULE 5

In this module it provides three different functionality.

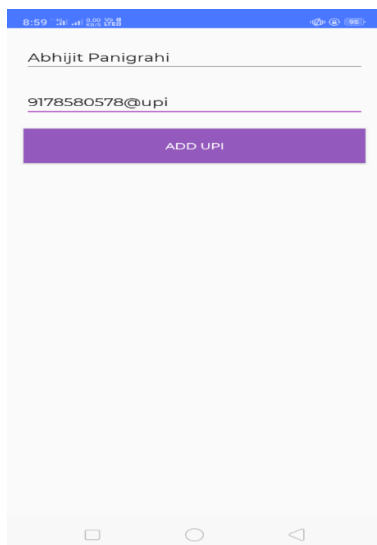


Fig 4.2.5.1 set senders UPI

In figure 5.1 SET SENDERS UPI, the user have to set their UPI ID which will enable the user to send the money.

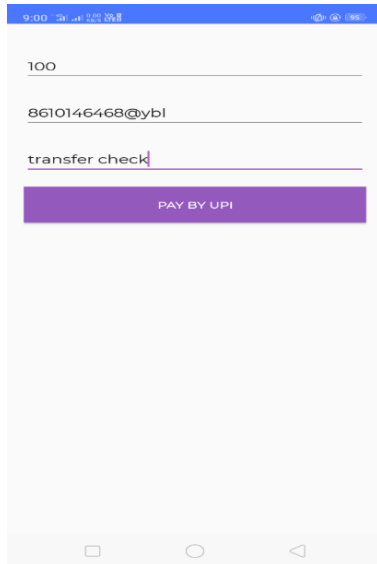


Fig 4.2.5.2 set receivers UPI

In figure 5.2 SET RECEIVERS UPI, the user have to set the receivers UPI address and amount.

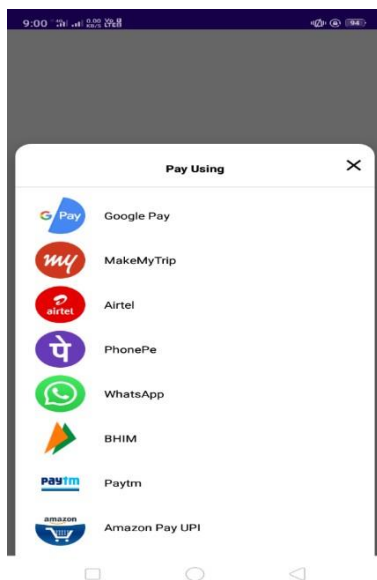


Fig 4.2.5.3 pay using any UPI app

In figure 5.3 PAY USING UPI APP, by selecting any application which support UPI the user can pay through that application.

MODULE 6

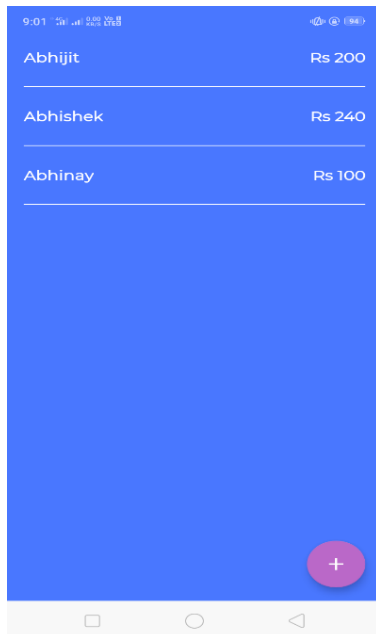


FIG 4.2.6 SPLIT BILLS

In this module the user can keep track of who owns whom, and can split bills accordingly.

MODULE 7

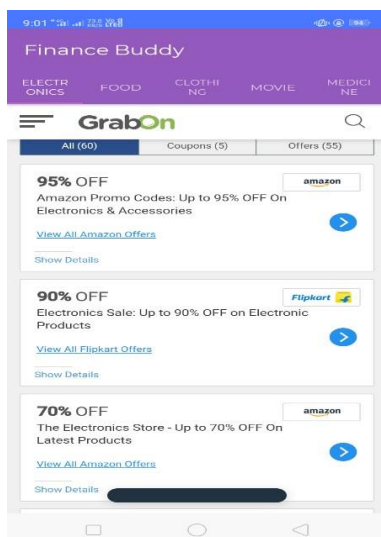


FIG 4.2.7 COUPONS/OFFERS

In this module the user can view all the offers available for online shopping.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION

In this project, we develop a mobile application that Keeps track of all your daily transactions, keeps track of your money lent or borrowed ,suggests you with the best investment options, offers your discounts in popular categories , view stock market and to read latest authenticated financial news. The application will eliminated sticky note, spreadsheet and ledger that caused confusions, data inconsistency. Now, with our application user can manage his expenses more effectively. This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs of users. As part of research, we considered adding certain features to the application to make it more useful to the user. Some of the extra features are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile information to the application, other features to be added are discussed down below in the future work section. The application delivered efficiently in calculating split expenses and recording the expenses along with date and time.

FUTURE WORK

We intend to add the following features into our expense manager application

- **SHOWING TRENDS**

In the future we will add trends. This is the graphical representation of the overall user expenses per certain period, that could be monthly, quarterly, half-yearly and yearly as well. This works with the algorithm that calculates average expenses of the user for certain period by using the expenses records created using this application.

Apart from showing trends of expenses on whole, this will also give expenses in detail by showing the trends by expense categories.

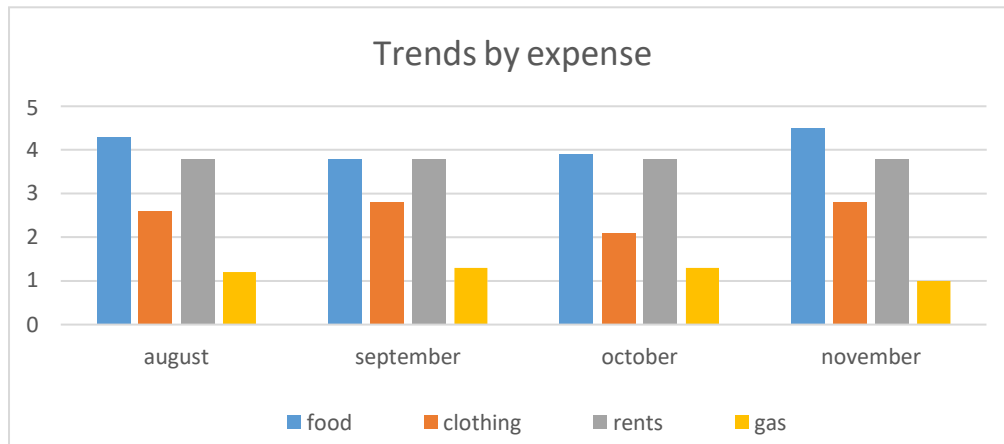


FIG 5.2.1 Trends

- **ESTIMATING EXPENSES**

The other part unique part of our expense manager application is to estimate expenses that user may spend in current or next month. This will help the user to adjust his expenses accordingly so he/she can avoid running out of money. This estimations can be derived from his/her previous expenses using algorithm that uses records from expense table.

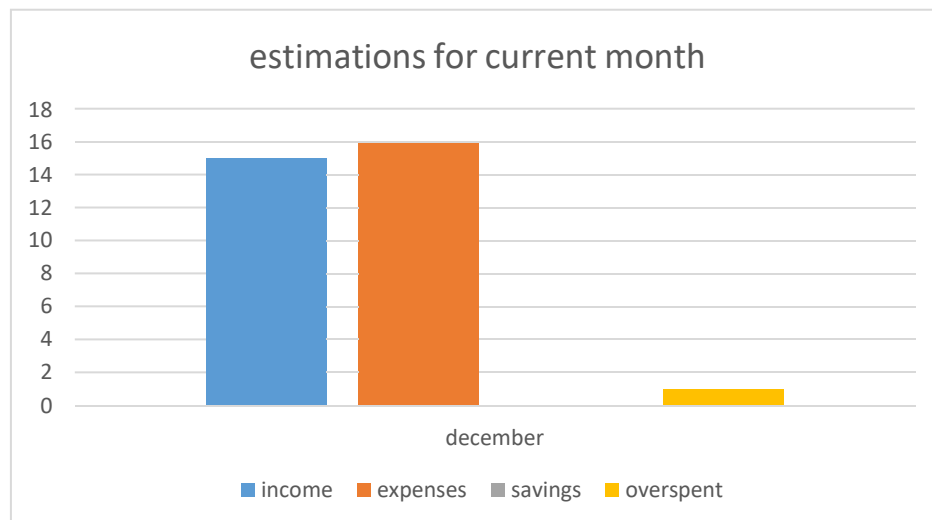


Fig 5.2.2 Estimations

During estimating expenses the algorithm should avoid using records that has huge variations because these expenses could be for long term in nature and they doesn't occur frequently. This helps users to manage their expenses efficiently. The below graph is estimations for month of December, here it shows higher expense then that shown in trends graph because it considers Christmas in this month and expects overspending of money.

- **COLLECTING USER DATA**

The collection of data of expenses and income of user from databases is easy task since every record of user activity is stored directly in database server maintained by us. But, to make our application safe and to maintain confidentiality of user data we undertake necessary steps to counter this problem. We will ask for user permissions to use user data for analysis purpose and will ensure confidentiality of every data related to data. The data samples are collected from user without disclosing any of user identity.

- **APPLYING DATA MINING TECHNIQUES**

The data collected from user is used to evolve certain user patterns, clustering and association of the expenses. The Extraction of interesting patterns that is non-trivial, implicit, previously unknown and potentially useful patterns or knowledge from huge amount of data. We would apply appropriate algorithms of some famous data mining techniques available today. However, this would need helping hand from a data mining expert.

- **USAGE PATTERNS IN MARKET ANALYSIS**

The Data collected from user and the patterns evolved after applying data mining techniques as elaborated in the above part of this report are used for market analysis. The experts in the market can use these patterns of user expenditures based on age, gender, place of user. This will help market strategist to plan accordingly so that it stabilize the demand supply in the market benefiting all the stakeholders.

REFERENCE

1. <https://developer.android.com/docs>
2. <https://newsapi.org/>
3. <https://www.groupon.com/>
4. <https://www.sqlite.org/docs.html>
5. <https://www.moneycontrol.com/stocksmarketsindia/>
6. <https://github.com/>
7. <https://stackoverflow.com/>
8. <https://www.w3schools.com/>
9. <https://economictimes.indiatimes.com/wealth/fuel-price/petrol>
10. https://www.livechennai.com/pricelist_chennai.asp

APPENDIX

SAMPLE CODE

MANIFEST

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="tech.risinglight.financebuddy">

    <uses-permission android:name="android.permission.READ_SMS" />

    <uses-permission android:name="android.permission.WRITE_SMS" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application

        android:allowBackup="true"

        android:icon="@mipmap/ic_launcher"

        android:label="@string/app_name"

        android:roundIcon="@mipmap/ic_launcher_round"

        android:supportsRtl="true"

        android:theme="@style/AppTheme"

        android:usesCleartextTraffic="true">

        <activity android:name=".view.NewsDetailsActivity"></activity>

        <activity android:name=".view.AddSplitwiseActivity" />

        <activity android:name=".view.MessageDetails" />

        <activity android:name=".view.NewsActivity" />

        <activity
```

```

        android:name=".view.CouponsActivity"

        android:label="@string/title_activity_coupons"

        android:theme="@style/AppTheme.NoActionBar" />
<activity android:name=".view.AddCardActivity" />
<activity android:name=".view.UpiPaymentActivity" />
<activity android:name=".view.SplitwiseActivity" />
<activity android:name=".view.WebActivity" />
<activity android:name=".view.InvestmentActivity" />
<activity android:name=".view.MessageActivity" />
<activity android:name=".view.ExpenseManagerActivity" />
<activity android:name=".view.MainActivity" />
<activity

        android:name=".view.SplashActivity"

        android:theme="@style/SplashTheme">

        <intent-filter>

            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />

        </intent-filter>

    </activity>
</application>
</manifest>

```

MAIN ACTIVITY

```
package tech.risinglight.financebuddy.view

import android.Manifest

import android.annotation.SuppressLint

import android.content.Context

import android.content.Intent

import android.content.pm.PackageManager

import android.os.Bundle

import android.provider.Telephony

import android.view.View

import androidx.appcompat.app.AppCompatActivity

import androidx.core.app.ActivityCompat

import androidx.core.content.ContextCompat

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.google.android.material.bottomsheet.BottomSheetBehavior

import kotlinx.android.synthetic.main.bottom_sheet_main.*

import kotlinx.android.synthetic.main.content_main.*

import kotlinx.coroutines.*

import kotlinx.coroutines.Dispatchers.IO

import tech.risinglight.financebuddy.R

import tech.risinglight.financebuddy.adapter.CardRecyclerViewAdapter

import tech.risinglight.financebuddy.adapter.TransactionsRecyclerViewAdapter

import tech.risinglight.financebuddy.model.Message
```

```

import tech.risinglight.financebuddy.repo.Repo

import tech.risinglight.financebuddy.model.Transaction

import java.util.*

import kotlin.collections.ArrayList

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        bottom_sheet.setBackgroundResource(R.drawable.custom_card_round)

        expenseManagerCV.setOnClickListener {

            startActivity(Intent(applicationContext, ExpenseManagerActivity::class.java))

        }

        investmentCV.setOnClickListener {

            startActivity(Intent(applicationContext, InvestmentActivity::class.java))

        }

        newsCV.setOnClickListener {

            startActivity(Intent(applicationContext, NewsActivity::class.java))

        }

        val layoutManager = LinearLayoutManager(this, RecyclerView.HORIZONTAL, false)

        val repo = Repo(application)

        repo.getAll().observe(this, androidx.lifecycle.Observer {

            val adapter = CardRecyclerViewAdapter(it)

```

```

recyclerView.layoutManager = layoutManager

recyclerView.adapter = adapter

})

splitwiseCV.setOnClickListener {

    val intent = Intent(applicationContext, SplitwiseActivity::class.java)

    startActivity(intent)

}

upiCardBtn.setOnClickListener {

    startActivity(Intent(applicationContext, AddCardActivity::class.java))

}

couponBtn.setOnClickListener {

    startActivity(Intent(applicationContext, CouponsActivity::class.java))

}

addbottomSheetCallBack()

if (ContextCompat.checkSelfPermission(applicationContext,
Manifest.permission.READ_SMS)

    != PackageManager.PERMISSION_GRANTED

) {

    ActivityCompat.requestPermissions(

        this,

        arrayOf(Manifest.permission.READ_SMS), 0

    )

```

```

GlobalScope.launch {

    getTransactions(applicationContext)

}

} else {

    GlobalScope.launch {

        getTransactions(applicationContext)

    }}

private fun addbottomSheetCallBack() {

    val bottomSheetBehavior = BottomSheetBehavior.from(bottom_sheet)

    bottomSheetBehavior.addBottomSheetCallback(object :

        BottomSheetBehavior.BottomSheetCallback() {

            override fun onSlide(bottomSheet: View, slideOffset: Float) {

            }

        })

    @SuppressWarnings("SwitchIntDef")

    override fun onStateChanged(bottomSheet: View, newState: Int) {

        when (newState) {

            BottomSheetBehavior.STATE_HIDDEN -> {

            }

        }
    }
}

```

```

        BottomSheetBehavior.STATE_EXPANDED -> {

            bottomSheet.setBackgroundResource(R.drawable.custom_card_noround)

        }

        BottomSheetBehavior.STATE_COLLAPSED -> {

            bottom_sheet.setBackgroundResource(R.drawable.custom_card_round)

        }

        BottomSheetBehavior.STATE_DRAGGING -> {

            bottom_sheet.setBackgroundResource(R.drawable.custom_card_round)

        }

        BottomSheetBehavior.STATE_SETTLING -> {

            bottom_sheet.setBackgroundResource(R.drawable.custom_card_round)

        } } } } }

private suspend fun getTransactions(context: Context) = withContext(IO) {

    val cursor =

        context.contentResolver.query(Telephony.Sms.CONTENT_URI, null, null, null, null)

    val messages = ArrayList<Message>()

    val regex = Regex("((?:Rs.)s*(\\d+,*\\d*(?:\\.\\d{2})?))")

    val transactions = ArrayList<Transaction>()

    while (cursor != null && cursor.moveToNext()) {

        val smsDate = cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.DATE))

        val number =

            cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.ADDRESS))

        val body = cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.BODY))

        val message = Message(number, body, Date(smsDate.toLong()))

```

```

    messages.add(message)
}

messages.sortedWith(compareBy { it.date })

messages.forEach { message ->

    val matches = regex.findAll(message.body)

    matches.forEach {

        if (message.date.toString().contains("2020"))

        ) {

            if (message.body.contains("debit") || message.body.contains("debited")

                || message.body.contains("Paid") ||

                message.body.contains("sent") ||

                message.body.contains("paid") ||

                message.body.contains("Sent") ||

                message.body.contains("charged") || message.body.contains(

                    "Debit"

                ) || message.body.contains("DEBITED") || message.body.contains(

                    "DEBIT"

                )

            ) {

                val transaction =

                    Transaction(message.number, it.value, message.date, false)

                transactions.add(transaction)
            }
        }
    }
}

```

```

        } else if (message.body.contains("credit") || message.body.contains("Credit") ||
message.body.contains(
            "credited"
        ) ||
        message.body.contains("received") || message.body.contains("Received") ||
message.body.contains(
            "CREDITED"
        ) || message.body.contains(
            "CREDIT"
        )
    ) {
        val transaction =
            Transaction(message.number, it.value, message.date, true)
        transactions.add(transaction)
    } } }
    setBottomAdapters(context, transactions)
}

```

```

private suspend fun setBottomAdapters(context: Context, transactions:
ArrayList<Transaction>) =
    withContext(Dispatchers.Main) {
        sheetRV.layoutManager = LinearLayoutManager(context)
        sheetRV.adapter = TransactionsRecyclerViewAdapter(transactions)
    }
}

```

MESSAGE ACITVITY

```
package tech.risinglight.financebuddy.view

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import tech.risinglight.financebuddy.R
```

```
class MessageActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_message)
    }
}
```

```
package tech.risinglight.financebuddy.view

import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.Telephony
import androidx.recyclerview.widget.LinearLayoutManager
import kotlinx.android.synthetic.main.activity_message_details.*
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import tech.risinglight.financebuddy.R
import tech.risinglight.financebuddy.adapter.TransactionsRecyclerViewAdapter
import tech.risinglight.financebuddy.model.Message
```

```

import tech.risinglight.financebuddy.model.Transaction
import java.util.*
import kotlin.collections.ArrayList

class MessageDetails : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_message_details)
        val number = intent.getStringExtra("number")

        GlobalScope.launch {
            getTransactions(applicationContext, number)
        }
    }

    private suspend fun getTransactions(context: Context, number: String?) =
        withContext(Dispatchers.IO) {

            val cursor =
                context.contentResolver.query(Telephony.Sms.CONTENT_URI, null, null, null, null)
            val messages = ArrayList<Message>()
            val regex = Regex("""((?:Rs.)\s*(\d+,*\d*(?:\.\d{2})?))""")
            val transactions = ArrayList<Transaction>()
            while (cursor != null && cursor.moveToNext()) {
                val smsDate = cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.DATE))
                val number =
                    cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.ADDRESS))
                val body = cursor.getString(cursor.getColumnIndexOrThrow(Telephony.Sms.BODY))
            }
        }
    }

```

```

    val message = Message(number, body, Date(smsDate.toLong()))
    messages.add(message)
}
messages.sortedWith(compareBy { it.date })
messages.forEach { message ->
    val matches = regex.findAll(message.body)
    matches.forEach {
        if (message.number.equals(number!!)) {
            if (message.date.toString().contains("2020"))
        ) {
            if (message.body.contains("debit") || message.body.contains("debited")
                || message.body.contains("Paid") ||
                message.body.contains("sent") ||
                message.body.contains("paid") ||
                message.body.contains("Sent") ||
                message.body.contains("charged") || message.body.contains(
                    "Debit"
                ) || message.body.contains("DEBITED") || message.body.contains(
                    "DEBIT"
                )
            ) {
                val transaction =
                    Transaction(message.number, it.value, message.date, false)
                transactions.add(transaction)

            } else if (message.body.contains("credit") || message.body.contains("Credit") ||
message.body.contains(
                    "credited"
                ) ||

```

```

                message.body.contains("received") || message.body.contains("Received") ||
message.body.contains(
                    "CREDITED"
                ) || message.body.contains(
                    "CREDIT"
                )) {
                val transaction =
                    Transaction(message.number, it.value, message.date, true)
                transactions.add(transaction)
            }}}}
        setAdapters(context, transactions)    } }

```

```

private suspend fun setAdapters(context: Context, transactions: ArrayList<Transaction>) =
    withContext(Dispatchers.Main) {
        messageDetailsRV.layoutManager = LinearLayoutManager(context)
        messageDetailsRV.adapter = TransactionsRecyclerViewAdapter(transactions)
    }

```

UPI ACTIVITY

```

package tech.risinglight.financebuddy.view

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import com.shreyaspatil.EasyUpiPayment.EasyUpiPayment

import kotlinx.android.synthetic.main.activity_upi_payment.*

import tech.risinglight.financebuddy.R

import kotlin.random.Random

class UpiPaymentActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

```

```

setContentView(R.layout.activity_upi_payment)

if (intent != null) {
    upi_idET.setText(intent.getStringExtra("UPI"))
}

val des= if (noteET.text.toString().isEmpty()) noteET.text.toString() else "Finance
buddy"

payBtn.setOnClickListener {
    val amount =
        if (amountET.text.toString().contains('.')) amountET.text.toString() else
amountET.text.toString() + ".00"

    val random = Random(1100000)

    val easyUpiPayment = EasyUpiPayment.Builder()
        .with(this)
        .setPayeeVpa(upi_idET.text.toString())
        .setPayeeName("via Finance Buddy")
        .setTransactionId(random.nextInt().toString())
        .setTransactionRefId(random.nextInt().toString())
        .setDescription(noteET.text.toString())
        .setAmount(amount)
        .build()

    easyUpiPayment.startPayment()
} } }

```