MONEY MATTERS:A PERSONAL FINANCE

MANAGEMENT

APP

Project presented by

Team ID: NM2023TMID

Team Leader:

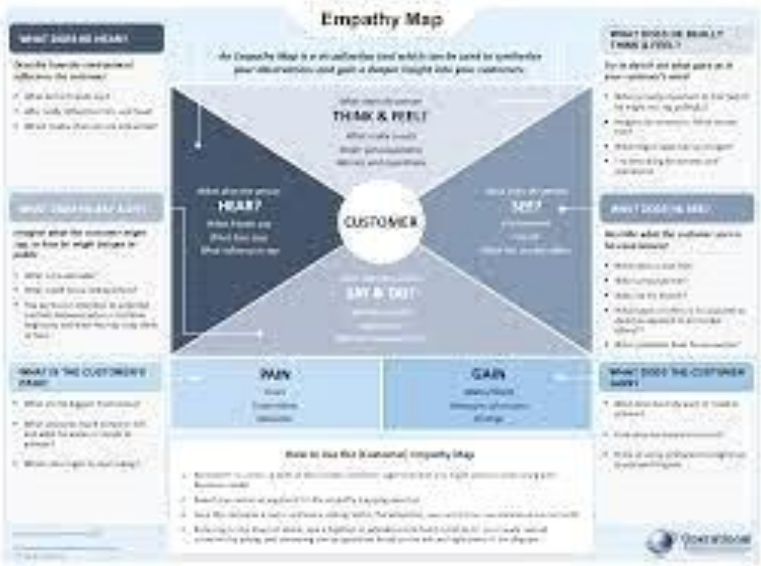
Akash.K

Team members:

Akash.C
Anantharaj.R
Easwaran.R
Sasikannan.J
Vishvaraman.N

- # INTRODUCTION:

- 1.1. Overview:
- One of the most valuable life skills is learning how to manage money, but it is important to start at the beginning. Why we use money in the first place and howto get the best value with our money should be explained. We should also explore our relationship with money.
- Taking the time to understand the influence that money has in our lives can helpto contribute to a healthy relationship with money. Students will learn through lessons and activities how money works and how they can make the most of it, thus ensuring they have better control of their money in the long run.
-
- 1.2. Purpose:
- The Money Matters module shows students how to manage their money by preparing a personal spending plan and identifying ways to decrease spendingand increase income.
- 2. PROBLEM DEFINITION & DESIGN THINKING
- 2.1. Empathy map:
-



2.2 Brainstorming map:

- **ADVANTAGES AND DISADVANTAGE:**

Advantages:

- Money gives you freedom. When you have enough money, you can live whereyou want, take care of your needs, and indulge in your hobbies. ...
- Money gives you the power to pursue your dreams. ...
- Money gives you security.

Disadvantages:

- No interest charges. There are no additional charges when you pay with cash. ...
- Makes it easier to follow a budget. Cash can help you to stick to a budget. ...
- Cons:

- Less Secure. Cash is less secure than a credit card. ...
- Less Convenient. ...
- Your cash savings may not cover certain expenses. …

OUTPUT:

- **APPLICATION:**

Money Matters Money Advice Centre has secured funding from The British Gas EnergyTrust, to allow us to provide a vital service to vulnerable and struggling clients who reside in Glasgow and South Lanarkshire. This "More Electricity & Gas Assistance" (MEGA) fund is to be used for the provision of those in priority need or who have been affected by the cost-of-living crisis.

MEGA Fund - More Electricity & Gas Assistance fund is available to customers with prepayment meters. Our Emergency Utility Credit Vouchers (maximum of 3 vouchers allowed per individual or couple/family) for all fuel company customers with pre-payment meters (each voucher value will be a maximum of £49..

Applicants' personal data will not be used for any marketing purposes. We undertake topreserve the confidentiality of all information you provide to the Money Matters Money Advice Centre.

- CONCLUSION:

Personal financial management is done by every individual on some level. The key is tostrike the right balance between income, expenses, savings, and investments. This balance will ensure that the personal financial planning and management of the individual are optimum.

- FUTURE SCOPE:

The field of finance has a huge scope in future. As finance is an integral part of our economy, Financial Managers will always be in high demand. If you want to build a career in finance, the most popular sectors include corporate finance and public banking, credit and financial planning, and asset management.

- APPENDIX:

Budgeting, investing, saving and even spending are all a part ofmoney management. So how do you build money confidence and reduceanxiety about your financial goals? Finding ways to better manage your money—and your mindset

- SORCE CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_ru
        les" android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.ExpensesTracker"
        tools:targetApi="31">
        <activity
            android:name=".RegisterActivity"
            android:exported="false"
            android:label="@string/title_activity_register"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".ViewRecordsActivity"
            android:exported="false"
```

```
            android:label="@string/title_activity_view_recor
            ds" android:theme="@style/Theme.ExpensesTracker"
            />
        <activity
            android:name=".SetLimitActivity"
            android:exported="false"
            android:label="@string/title_activity_set_limit"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".AddExpensesActivity"
            android:exported="false"
            android:label="@string/title_activity_add_expens
            es" android:theme="@style/Theme.ExpensesTracker"
            />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ExpensesTracker">

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

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Color.kt

```
package com.example.expensestracker.ui.theme


import androidx.compose.ui.graphics.Color


val          Purple200          =

Color(0xFFBB86FC) val  Purple500

=      Color(0xFF6200EE)      val

Purple700   =   Color(0xFF3700B3)

val Teal200 = Color(0xFF03DAC5)
```

Shape.kt

```
package com.example.expensestracker.ui.theme
```

```
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small =
    RoundedCornerShape(4.dp), medium
    = RoundedCornerShape(4.dp),large
    = RoundedCornerShape(0.dp)
)
```

Theme.kt

```
package com.example.expensestracker.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant =
    Purple700,secondary =
    Teal200
)
```

```
private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant =
    Purple700,secondary =
    Teal200

    /* Other default colors to
    overridebackground = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary =
    Color.Black, onBackground
    = Color.Black,onSurface =
    Color.Black,
    */
)

@Composable
fun ExpensesTrackerTheme(
    darkTheme: Boolean =
    isSystemInDarkTheme(),content:
    @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }
```

```
    MaterialTheme(
        colors =
        colors,
        typography =
        Typography,shapes =
        Shapes,
        content = content
    )
}
```

Type.kt

```
package com.example.expensestracker.ui.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import
androidx.compose.ui.text.font.FontFamily
import
androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography =
    Typography(body1 =
    TextStyle(
        fontFamily =
        FontFamily.Default,fontWeight
        = FontWeight.Normal, fontSize
        = 16.sp
```

```
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily =
        FontFamily.Default,fontWeight
        = FontWeight.W500, fontSize =
        14.sp
    ),
    caption = TextStyle(
        fontFamily =
        FontFamily.Default,fontWeight
        = FontWeight.Normal, fontSize
        = 12.sp
    )
    */
)
```

AddExpensesActivity.kt

```
package com.example.expensestracker


import android.annotation.SuppressLint
import
android.content.Context
import
android.content.Intent
import android.os.Bundle
import android.widget.Toast
```

```
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import
androidx.compose.ui.Alignment
import
androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class AddExpensesActivity : ComponentActivity() {
    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper
    private lateinit var expenseDatabaseHelper:
    ExpenseDatabaseHelper
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        itemsDatabaseHelper = ItemsDatabaseHelper(this)
        expenseDatabaseHelper =
        ExpenseDatabaseHelper(this)setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
```

```
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor =
                        Color(0xFFadbef4),modifier =
                        Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {

                            Spacer(modifier = Modifier.width(15.dp))

                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.java))
                                },
                                colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                                modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                            )
                            {
                                Text(
                                    text = "Add Expenses", color =
Color.Black, fontSize = 14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }

                            Spacer(modifier = Modifier.width(15.dp))
```

```
                    Button(
                        onClick = {
                            startActivity(
                                Intent(

                                    applicationContext,
                                    SetLimitActivity::class.java
                                )
                            )
                        },
                         colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                         modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                    )
                    {
                        Text(
                            text = "Set Limit", color = Color.Black,
fontSize = 14.sp,
                            textAlign = TextAlign.Center
                        )
                     }

                    Spacer(modifier = Modifier.width(15.dp))

                    Button(
                        onClick = {
                            startActivity(
                                Intent(
```

```
                                        applicationContext,
                                        ViewRecordsActivity::class.java
                                    )
                                )
                            },
                            colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                            modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                        )
                        {
                            Text(
                                text = "View Records", color =
Color.Black, fontSize = 14.sp,
                                textAlign = TextAlign.Center
                            )
                        }

                    }
                )
            }
        ) {
            AddExpenses(this, itemsDatabaseHelper, expenseDatabaseHelper)
        }
    }
}
```

```
@SuppressLint("Range")

@Composable

fun AddExpenses(context: Context, itemsDatabaseHelper:
ItemsDatabaseHelper,expenseDatabaseHelper: ExpenseDatabaseHelper) {

    Column(

        modifier = Modifier

            .padding(top = 100.dp, start = 30.dp)

            .fillMaxHeight()

            .fillMaxWidth(),

        horizontalAlignment =

        Alignment.Start

    ) {


        val mContext = LocalContext.current

        var items by remember { mutableStateOf("") }

        var quantity by remember { mutableStateOf("")

        }var cost by remember { mutableStateOf("") }

        var error by remember { mutableStateOf("") }


        Text(text = "Item Name", fontWeight = FontWeight.Bold, fontSize =
20.sp)

        Spacer(modifier = Modifier.height(10.dp))

        TextField(value = items, onValueChange = { items = it

        },

            label = { Text(text = "Item Name") })


        Spacer(modifier = Modifier.height(20.dp))


        Text(text = "Quantity of item", fontWeight =
FontWeight.Bold,fontSize = 20.sp)
```

```
        Spacer(modifier = Modifier.height(10.dp))

        TextField(value = quantity, onValueChange = { quantity = it

            },label = { Text(text = "Quantity") })


        Spacer(modifier = Modifier.height(20.dp))

        Text(text = "Cost of the item", fontWeight =
FontWeight.Bold,fontSize = 20.sp)

        Spacer(modifier = Modifier.height(10.dp))

        TextField(value = cost, onValueChange = { cost = it

        },

            label = { Text(text = "Cost") })


        Spacer(modifier = Modifier.height(20.dp))


        if (error.isNotEmpty()) {

            Text(

                text = error,

                color = MaterialTheme.colors.error,

                modifier = Modifier.padding(vertical = 16.dp)

            )

        }


        Button(onClick = {

            if (items.isNotEmpty() && quantity.isNotEmpty()
&&cost.isNotEmpty()) {

                val items =

                    Items(id =

                    null,
```

```
                    itemName = items,
                    quantity =
                    quantity,cost =
                    cost
                )


                val limit= expenseDatabaseHelper.getExpenseAmount(1)


                val actualvalue = limit?.minus(cost.toInt())
                // Toast.makeText(mContext, actualvalue.toString(),
Toast.LENGTH_SHORT).show()

                val expense =
                    Expense(id = 1,
                    amount = actualvalue.toString()
                )
                if (actualvalue != null) {
                    if (actualvalue < 1) {
                        Toast.makeText(mContext, "Limit
Over",Toast.LENGTH_SHORT).show()
                    } else  {
                        expenseDatabaseHelper.updateExpense(expense)
                        itemsDatabaseHelper.insertItems(items)
                    }
                }

            }
        }) {
```

```
            Text(text = "Submit")
        }

    }
}
```

Expenses.kt

```
package com.example.expensestracker

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

@Entity(tableName = "expense_table")
data class Expense(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "amount") val amount: String?,
)
```

ExpenseDao.kt

```
package com.example.expensestracker

import androidx.room.*

@Dao
interface ExpenseDao {
```

```
    @Query("SELECT * FROM expense_table WHERE amount= :amount")
    suspend fun getExpenseByAmount(amount: String): Expense?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertExpense(items: Expense)

    @Update
    suspend fun updateExpense(items: Expense)

    @Delete
    suspend fun deleteExpense(items: Expense)
}
```

ExpensesDao.kt

```
package com.example.expensestracker

import androidx.room.*

@Dao
interface ExpenseDao {

    @Query("SELECT * FROM expense_table WHERE amount= :amount")
    suspend fun getExpenseByAmount(amount: String): Expense?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertExpense(items: Expense)

    @Update
    suspend fun updateExpense(items: Expense)
```

```
    @Delete
    suspend fun deleteExpense(items: Expense)
}
```

ExpenseDatabase.kt

```
package com.example.expensestracker

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

import androidx.room.RoomDatabase

@Database(entities = [Items::class], version = 1)
abstract class ExpenseDatabase : RoomDatabase() {

    abstract fun ExpenseDao(): ItemsDao

    companion object {

        @Volatile
        private var instance: ExpenseDatabase? = null

        fun getDatabase(context: Context): ExpenseDatabase {
            return instance ?: synchronized(this) {
```

```
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    ExpenseDatabase::class.java,
                    "expense_database"
                ).build()
                instance =
                newInstance
                newInstance
            }
        }
    }
}
```

ExpenseDatabaseHelper.kt

```
package com.example.expensestracker

import
android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper


class ExpenseDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME,
    null,DATABASE_VERSION){


    companion object {
        private const val DATABASE_VERSION = 1
```

```
        private const val DATABASE_NAME = "ExpenseDatabase.db"


        private const val TABLE_NAME =

        "expense_table"private const val COLUMN_ID =

        "id"

        private const val COLUMN_AMOUNT = "amount"

    }


    override fun onCreate(db: SQLiteDatabase?) {

        val createTable = "CREATE TABLE $TABLE_NAME (" +

                "${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +

                "${COLUMN_AMOUNT} TEXT" +

                ")"


        db?.execSQL(createTable)

    }

    override fun onUpgrade(db1: SQLiteDatabase?, oldVersion: Int,
newVersion:Int) {

        db1?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")

        onCreate(db1)

    }


    fun insertExpense(expense: Expense) {

        val db1 = writableDatabase

        val values = ContentValues()

        values.put(COLUMN_AMOUNT, expense.amount)

        db1.insert(TABLE_NAME, null, values)

        db1.close()

    }
```

```
    fun updateExpense(expense: Expense) {

        val db = writableDatabase
        val values = ContentValues()

        values.put(COLUMN_AMOUNT, expense.amount)

        db.update(TABLE_NAME, values, "$COLUMN_ID=?",

arrayOf(expense.id.toString()))

        db.close()

    }




    @SuppressLint("Range")

    fun getExpenseByAmount(amount: String): Expense? {

        val db1 = readableDatabase

        val cursor: Cursor = db1.rawQuery("SELECT * FROM
${ExpenseDatabaseHelper.TABLE_NAME} WHERE

${ExpenseDatabaseHelper.COLUMN_AMOUNT} = ?", arrayOf(amount))

        var expense: Expense? =

        nullif

        (cursor.moveToFirst()) {

            expense = Expense(
                id =

                cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

                amount =

cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),

            )

        }

        cursor.close

        ()
```

```
        db1.close()

        return

        expense

    }

    @SuppressLint("Range")

    fun getExpenseById(id: Int): Expense? {

        val db1 = readableDatabase

        val cursor: Cursor = db1.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))

        var expense: Expense? =

        nullif

        (cursor.moveToFirst()) {

            expense = Expense(
                id =

                cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

                amount =

cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),

            )

        }

        cursor.close

        ()

        db1.close()

        return

        expense



    }

    @SuppressLint("Range")

    fun getExpenseAmount(id: Int): Int? {

        val db = readableDatabase

        val query = "SELECT $COLUMN_AMOUNT FROM $TABLE_NAME WHERE
```

```
$COLUMN_ID=?"
        val cursor = db.rawQuery(query, arrayOf(id.toString()))
        var amount: Int? = null
        if (cursor.moveToFirst()) {
            amount = cursor.getInt(cursor.getColumnIndex(COLUMN_AMOUNT))
        }
        cursor.close
        ()
        db.close()
        return
        amount
    }
    @SuppressLint("Range")
    fun getAllExpense(): List<Expense> {
        val expenses = mutableListOf<Expense>()
        val db1 = readableDatabase
        val cursor: Cursor = db1.rawQuery("SELECT * FROM $TABLE_NAME",
        null)if (cursor.moveToFirst()) {
            do {
                val expense = Expense(
                    id =
                    cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    amount =
cursor.getString(cursor.getColumnIndex(COLUMN_AMOUNT)),
                )
                expenses.add(expense)
            } while (cursor.moveToNext())
        }
```

```
        cursor.close(
        ) db1.close()
        return
        expenses
    }




}
```

Items.kt

```
package com.example.expensestracker


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


@Entity(tableName =
"items_table")data class Items(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "item_name") val itemName: String?,
    @ColumnInfo(name = "quantity") val quantity: String?,
    @ColumnInfo(name = "cost") val cost: String?,
)
```

ItemsDao.kt

```
package com.example.expensestracker
```

```
import androidx.room.*

@Dao
interface ItemsDao {

    @Query("SELECT * FROM items_table WHERE cost= :cost")
    suspend fun getItemsByCost(cost: String): Items?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertItems(items: Items)

    @Update
    suspend fun updateItems(items: Items)

    @Delete
    suspend fun deleteItems(items: Items)
}
```

ItemsDatabase.kt

```
package com.example.expensestracker

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

@Database(entities = [Items::class], version = 1)
```

```
abstract class ItemsDatabase : RoomDatabase() {

    abstract fun ItemsDao(): ItemsDao

    companion object {

        @Volatile
        private var instance: ItemsDatabase? = null

        fun getDatabase(context: Context): ItemsDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    ItemsDatabase::class.java,
                    "items_database"
                ).build()
                instance =
                newInstance
                newInstance
            }
        }
    }
}
```

ItemDatabaseHelper.kt

```
package com.example.expensestracker

import

android.annotation.SuppressLint
```

```
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper


class ItemsDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME,
    null,DATABASE_VERSION){

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "ItemsDatabase.db"

        private const val TABLE_NAME = "items_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_ITEM_NAME = "item_name"
        private const val COLUMN_QUANTITY = "quantity"
        private const val COLUMN_COST = "cost"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "${COLUMN_ITEM_NAME} TEXT," +
                "${COLUMN_QUANTITY} TEXT," +
                "${COLUMN_COST} TEXT" +
                ")"
```

```
        db?.execSQL(createTable)

    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion:Int) {

        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")

        onCreate(db)

    }


    fun insertItems(items: Items)

        { val db =

        writableDatabase val

        values = ContentValues()

        values.put(COLUMN_ITEM_NAME,

        items.itemName)values.put(COLUMN_QUANTITY,

        items.quantity) values.put(COLUMN_COST,

        items.cost) db.insert(TABLE_NAME, null,

        values) db.close()

    }




    @SuppressLint("Range")

    fun getItemsByCost(cost: String): Items? {

        val db = readableDatabase

        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_COST = ?", arrayOf(cost))

        var items: Items? = null
```

```
        if (cursor.moveToFirst())

            {items = Items(

                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

                itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),

                quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),

                cost = cursor.getString(cursor.getColumnIndex(COLUMN_COST)),

            )

        }

        cursor.close

        ()

        db.close()

        return items

    }

    @SuppressLint("Range")

    fun getItemsById(id: Int): Items? {

        val db = readableDatabase

        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
           $COLUMN_ID = ?",
        arrayOf(id.toString()))

       var items: Items? = null

        if (cursor.moveToFirst())

            {items = Items(

                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

                itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),

                quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),

                cost = cursor.getString(cursor.getColumnIndex(COLUMN_COST)),

            )
```

```
        }
        cursor.close
        ()
        db.close()
        return items
    }
    @SuppressLint("Range")
    fun getAllItems(): List<Items> {
        val item = mutableListOf<Items>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
        null)if (cursor.moveToFirst()) {
            do {
                val items = Items(
                    id =
                    cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    itemName =
cursor.getString(cursor.getColumnIndex(COLUMN_ITEM_NAME)),
                    quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
                    cost =
cursor.getString(cursor.getColumnIndex(COLUMN_COST)),
                )
                item.add(items)
            } while (cursor.moveToNext())
        }
        cursor.close
        ()
        db.close()
        return item
```

```
    }



}
```

LoginActivity.kt

```
package com.example.expensestracker

import

android.content.Context

import

android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import

androidx.compose.ui.Alignment

import

androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import

androidx.compose.ui.layout.ContentScale

import

androidx.compose.ui.res.painterResource

import

androidx.compose.ui.text.font.FontFamily
```

```
import
androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme


class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ExpensesTrackerTheme {
                // A surface container using the 'background' color from the
theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color =
                    MaterialTheme.colors.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
    }
```

```
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

    Image(
        painterResource(id = R.drawable.img_1), contentDescription =
        "",alpha =0.3F,
        contentScale = ContentScale.FillHeight,

        )

    var username by remember { mutableStateOf("")
    }    var    password    by    remember    {
    mutableStateOf("") } var error by remember {
    mutableStateOf("") }


    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment =
        Alignment.CenterHorizontally,verticalArrangement
        = Arrangement.Center
    ) {

        Text(
            fontSize = 36.sp,
            fontWeight =
            FontWeight.ExtraBold,fontFamily
```

```
    = FontFamily.Cursive, color =
    Color.White,
    text = "Login"
)
Spacer(modifier = Modifier.height(10.dp))

TextField(
    value = username,
    onValueChange = { username = it
    },label = { Text("Username") },
    modifier =
    Modifier.padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it
    },label = { Text("Password") },

    modifier = Modifier.padding(10.dp)
        .width(280.dp),
    visualTransformation = PasswordVisualTransformation()

)

if (error.isNotEmpty())
    {Text(
```

```
            text = error,
            color = MaterialTheme.colors.error,
            modifier = Modifier.padding(vertical = 16.dp)
        )
    }

    Button(
        onClick = {
            if (username.isNotEmpty() && password.isNotEmpty()) {
                val user = databaseHelper.getUserByUsername(username)
                if (user != null && user.password == password)
                    {error = "Successfully log in"
                    context.startActivity(
                        Intent(
                            context,
                            MainActivity::class.java
                        )
                    )
                    //onLoginSuccess()
                }
                else {
                    error =  "Invalid username or password"
                }

            } else {
                error = "Please fill all fields"
            }
        },
```

```
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    RegisterActivity::class.j
                    ava
                )
            )}
            )
            { Text(color = Color.White,text = "Sign up") }
            TextButton(onClick = {
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color.White,text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context,
    MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
```

```
}
```

MainActivity.kt

```
package com.example.expensestracker


import android.annotation.SuppressLint

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import

androidx.compose.ui.Alignment

import

androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import

androidx.compose.ui.res.painterResource

import

androidx.compose.ui.text.font.FontWeight

import

androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
```

```
class MainActivity : ComponentActivity() {
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter
    ") override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor =
                        Color(0xFFadbef4),modifier =
                        Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {

                            Spacer(modifier = Modifier.width(15.dp))

                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.java))},
                                colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                                modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                            )
                            {
```

```
                    Text(
                        text = "Add Expenses", color =
Color.Black, fontSize = 14.sp,
                        textAlign = TextAlign.Center
                    )
                }

                Spacer(modifier = Modifier.width(15.dp))

                Button(
                    onClick = {
                        startActivity(
                            Intent(
                                applicationContext,
                                SetLimitActivity::class.java
                            )
                        )
                    },
                    colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                    modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                )
                {
                    Text(
                        text = "Set Limit", color = Color.Black,
fontSize = 14.sp,
                        textAlign = TextAlign.Center
                    )
                }
```

```
                        Spacer(modifier = Modifier.width(15.dp))


                        Button(
                            onClick = {
                                startActivity(
                                    Intent(
                                        applicationContext,
                                        ViewRecordsActivity::class.java
                                    )
                                )
                            },
                            colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                            modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                        )
                        {
                            Text(
                                text = "View Records", color =
Color.Black, fontSize = 14.sp,
                                textAlign = TextAlign.Center
                            )
                        }


                    }
                )
            }
        ) {
            MainPage()
```

```
            }
        }
    }
}


@Composable
fun MainPage()
    {Column(
        modifier = Modifier.padding(20.dp).fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {


        Text(text = "Welcome To Expense Tracker", fontSize =
42.sp,fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center)


        Image(painterResource(id = R.drawable.img_1), contentDescription
="",modifier = Modifier.size(height = 500.dp, width = 500.dp))


    }
}
```

RegisterActivity.kt

```
package com.example.expensestracker




import android.content.Context
import android.content.Intent


import android.os.Bundle
```

```
import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat
```

```
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme



class RegisterActivity : ComponentActivity() {

    private lateinit var databaseHelper: UserDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        databaseHelper = UserDatabaseHelper(this)

        setContent {

            ExpensesTrackerTheme {

                // A surface container using the 'background' color from the
theme

                Surface(

                    modifier = Modifier.fillMaxSize(),

                    color = MaterialTheme.colors.background

                ) {



                    RegistrationScreen(this,databaseHelper)

                }

            }

        }
```

```
    }

}



@Composable

fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper)
{



    Image(

        painterResource(id = R.drawable.img_1), contentDescription = "",

        alpha =0.3F,

        contentScale = ContentScale.FillHeight,



        )



    var username by remember { mutableStateOf("") }

    var password by remember { mutableStateOf("") }

    var email by remember { mutableStateOf("") }

    var error by remember { mutableStateOf("") }
```

```
Column(
    modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {

    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,
        color = Color.White,
        text = "Register"
    )

    Spacer(modifier = Modifier.height(10.dp))
    TextField(
        value = username,
        onValueChange = { username = it },
        label = { Text("Username") },
```

```
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)

    )

    TextField(
        value = email,
        onValueChange = { email = it },
        label = { Text("Email") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )

    TextField(
        value = password,
        onValueChange = { password = it },
        label = { Text("Password") },
```

```
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp),
            visualTransformation = PasswordVisualTransformation()
        )



        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }


        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()
&&email.isNotEmpty()) {
```

```
            val user = User(
                id = null,
                firstName = username,
                lastName = null,
                email = email,
                password = password
            )
            databaseHelper.insertUser(user)
            error = "User registered successfully"
            // Start LoginActivity using the current context
            context.startActivity(
                Intent(
                   context,
                   LoginActivity::class.java
                )
            )

        } else {
            error = "Please fill all fields"
```

```
                }

            },

            modifier = Modifier.padding(top = 16.dp)

        ) {

            Text(text = "Register")

        }

        Spacer(modifier = Modifier.width(10.dp))

        Spacer(modifier = Modifier.height(10.dp))



        Row() {

            Text(

                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"

            )

            TextButton(onClick = {

                context.startActivity(

                    Intent(

                        context,

                        LoginActivity::class.java
```

```
                    )

                )

            })



            {

                Spacer(modifier = Modifier.width(10.dp))

                Text(text = "Log in")

            }

        }

    }

}

private fun startLoginActivity(context: Context) {

    val intent = Intent(context, LoginActivity::class.java)

    ContextCompat.startActivity(context, intent, null)

}
```

SetLimitACtivity.kt

```
package com.example.expensestracker


import android.annotation.SuppressLint
```

```
import
android.content.Context
import
android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import
androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.*
import androidx.compose.runtime.*
import
androidx.compose.ui.Alignment
import
androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import
androidx.compose.ui.text.font.FontWeight
import
androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp


import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme
```

```
class SetLimitActivity : ComponentActivity() {
    private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        expenseDatabaseHelper =
        ExpenseDatabaseHelper(this)setContent {
            Scaffold(
                // in scaffold we are specifying top bar.
                bottomBar = {
                    // inside top bar we are specifying
                    // background color.
                    BottomAppBar(backgroundColor =
                        Color(0xFFadbef4),modifier =
                        Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {

                            Spacer(modifier = Modifier.width(15.dp))

                            Button(
                               onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,
                                            AddExpensesActivity::class.java
```

```
                                    )
                                )
                            },
                            colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                            modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                        )
                        {
                            Text(
                                text = "Add Expenses", color =
Color.Black, fontSize = 14.sp,
                                textAlign = TextAlign.Center
                            )
                        }

                        Spacer(modifier = Modifier.width(15.dp))

                        Button(
                            onClick = {
                                startActivity(
                                    Intent(
                                        applicationContext,
                                        SetLimitActivity::class.java
                                    )
                                )
                            },
                            colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                            modifier = Modifier.size(height = 55.dp,
width = 110.dp)
```

```
                )
                {
                    Text(
                        text = "Set Limit", color = Color.Black,
fontSize = 14.sp,
                        textAlign = TextAlign.Center
                    )
                }

                Spacer(modifier = Modifier.width(15.dp))

                Button(
                    onClick = {
                        startActivity(
                            Intent(
                                applicationContext,
                                ViewRecordsActivity::class.java
                            )
                        )
                    },
                    colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                    modifier = Modifier.size(height = 55.dp,
width = 110.dp)
                )
                {
                    Text(
                        text = "View Records", color =
Color.Black, fontSize = 14.sp,
                        textAlign = TextAlign.Center
```

```
                                )
                            }
                        }
                    )
                }
            ) {
                val data=expenseDatabaseHelper.getAllExpense();
                Log.d("swathi" ,data.toString())
                val expense = expenseDatabaseHelper.getAllExpense()
                Limit(this, expenseDatabaseHelper,expense)
            }
        }
    }
}


@Composable
fun Limit(context: Context, expenseDatabaseHelper: ExpenseDatabaseHelper,
expense: List<Expense>) {
    Column(
        modifier = Modifier
            .padding(top = 100.dp, start = 30.dp)
            .fillMaxHeight()
            .fillMaxWidth(),
        horizontalAlignment =
        Alignment.Start
    ) {
```

```
        var amount by remember { mutableStateOf("")

        }var error by remember { mutableStateOf("")

        }


        Text(text = "Monthly Amount Limit", fontWeight =
FontWeight.Bold,fontSize = 20.sp)

        Spacer(modifier = Modifier.height(10.dp))

        TextField(value = amount, onValueChange = { amount = it

            },label = { Text(text = "Set Amount Limit ") })


        Spacer(modifier = Modifier.height(20.dp))


        if (error.isNotEmpty())

            {Text(

                text = error,

                color = MaterialTheme.colors.error,

                modifier = Modifier.padding(vertical = 16.dp)

            )

        }


        Button(onClick = {

            if (amount.isNotEmpty()) {

                val expense =

                    Expense(id =

                    null,

                    amount = amount

                )

                expenseDatabaseHelper.insertExpense(expense)

            }
```

```
        }) {
            Text(text = "Set Limit")
        }

        Spacer(modifier = Modifier.height(10.dp))

        LazyRow(
            modifier = Modifier
                .fillMaxSize()
                .padding(top = 0.dp),

            horizontalArrangement = Arrangement.Start
        ) {
            item {

                LazyColumn {
                    items(expense) { expense ->
                        Column(

                        ) {
                            Text("Remaining Amount:
${expense.amount}",fontWeight = FontWeight.Bold)
                        }
                    }
                }
            }

        }

    }
```

```
}
```

User.kt

```
package com.example.expensestracker


import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey


@Entity(tableName = "user_table")

data class User(

    @PrimaryKey(autoGenerate = true) val id: Int?,

    @ColumnInfo(name = "first_name") val firstName: String?,

    @ColumnInfo(name = "last_name") val lastName: String?,

    @ColumnInfo(name = "email") val email: String?,

    @ColumnInfo(name = "password") val password: String?,



    )
```

UserDao.kt

```
package com.example.expensestracker


import androidx.room.*


@Dao
interface UserDao {


    @Query("SELECT * FROM user_table WHERE email =
    :email")suspend fun getUserByEmail(email: String):
    User?


    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)


    @Update
    suspend fun updateUser(user: User)


    @Delete
    suspend fun deleteUser(user: User)
}
```

UserDatabase.kt

```
package com.example.expensestracker


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

```
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance =
                newInstance
                newInstance
            }
        }
    }
}
```

UserDatabaseHelper.kt

```
package com.example.expensestracker
```

```
import android.annotation.SuppressLint

import

android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper


class UserDatabaseHelper(context: Context) :

    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {


    companion object {

        private const val DATABASE_VERSION = 1

        private const val DATABASE_NAME = "UserDatabase.db"


        private const val TABLE_NAME = "user_table"

        private const val COLUMN_ID = "id"

        private const val COLUMN_FIRST_NAME =

        "first_name"private const val COLUMN_LAST_NAME =

        "last_name" private const val COLUMN_EMAIL =

        "email"

        private const val COLUMN_PASSWORD = "password"

    }


    override fun onCreate(db: SQLiteDatabase?) {

        val createTable = "CREATE TABLE $TABLE_NAME (" +

                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
```

```
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion:Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME,
        user.firstName)values.put(COLUMN_LAST_NAME,
        user.lastName) values.put(COLUMN_EMAIL,
        user.email) values.put(COLUMN_PASSWORD,
        user.password) db.insert(TABLE_NAME, null,
        values) db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
```

```
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_FIRST_NAME = ?", arrayOf(username))

        var user: User? = null

        if (cursor.moveToFirst()) {

            user = User(
                id =

                cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

                firstName =

cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),

                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =

                cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),

                password =

cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),

            )

        }

        cursor.close

        ()

        db.close()

        return user

    }

    @SuppressLint("Range")

    fun getUserById(id: Int): User? {

        val db = readableDatabase

        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))

        var user: User? = null

        if (cursor.moveToFirst())

            {user = User(

                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

```
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
                cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close
        ()
        db.close()
        return user
    }

    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
        null)if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id =
                    cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
```

```
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()

        return users
    }

}
```

ViewRecordsActivity.kt

```
package com.example.expensestracker



import android.annotation.SuppressLint

import android.content.Intent

import android.os.Bundle

import android.util.Log

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.ScrollState
```

```
import androidx.compose.foundation.layout.*

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.LazyRow

import androidx.compose.foundation.lazy.items

import androidx.compose.foundation.verticalScroll

import androidx.compose.material.*
import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.expensestracker.ui.theme.ExpensesTrackerTheme



class ViewRecordsActivity : ComponentActivity() {

    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper

    @SuppressLint("UnusedMaterialScaffoldPaddingParameter",
"SuspiciousIndentation")
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    itemsDatabaseHelper = ItemsDatabaseHelper(this)
    setContent {
        Scaffold(
            // in scaffold we are specifying top bar.
            bottomBar = {
                // inside top bar we are specifying
                // background color.
                BottomAppBar(backgroundColor = Color(0xFFadbef4),
                    modifier = Modifier.height(80.dp),
                    // along with that we are specifying
                    // title for our top bar.
                    content = {


                        Spacer(modifier = Modifier.width(15.dp))


                        Button(
```

```
                onClick = {
                    startActivity(
                        Intent(
                            applicationContext,
                            AddExpensesActivity::class.java
                        )
                    )
                },
                colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
                modifier = Modifier.size(height = 55.dp,
width = 110.dp)
            )
            {
                Text(
                    text = "Add Expenses", color =
Color.Black, fontSize = 14.sp,
                    textAlign = TextAlign.Center
                )
            }
```

```
                    Spacer(modifier = Modifier.width(15.dp))



                    Button(

                        onClick = {

                            startActivity(

                                Intent(

                                    applicationContext,

                                    SetLimitActivity::class.java

                                )

                            )

                        },

                        colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),

                        modifier = Modifier.size(height = 55.dp,
width = 110.dp)

                    )

                    {

                        Text(

                            text = "Set Limit", color = Color.Black,
fontSize = 14.sp,

                            textAlign = TextAlign.Center
```

```
            )
        }
        Spacer(modifier = Modifier.width(15.dp))

        Button(
            onClick = {
                startActivity(
                    Intent(
                        applicationContext,
                        ViewRecordsActivity::class.java
                    )
                )
            },
            colors =
ButtonDefaults.buttonColors(backgroundColor = Color.White),
            modifier = Modifier.size(height = 55.dp,
width = 110.dp)
        )
        {
            Text(
```

```
                                    text = "View Records", color =
Color.Black, fontSize = 14.sp,
                                    textAlign = TextAlign.Center
                                )
                            }


                        }
                    )
                }
            ) {
                val data=itemsDatabaseHelper.getAllItems();
                Log.d("swathi" ,data.toString())
                val items = itemsDatabaseHelper.getAllItems()
                    Records(items)
                }
            }
        }
    }
```

```
@Composable

fun Records(items: List<Items>) {

    Text(text = "View Records", modifier = Modifier.padding(top =
24.dp,start = 106.dp, bottom = 24.dp ), fontSize = 30.sp, fontWeight =
FontWeight.Bold)

    Spacer(modifier = Modifier.height(30.dp))

    LazyRow(

        modifier = Modifier

            .fillMaxSize()

            .padding(top = 80.dp),



        horizontalArrangement = Arrangement.SpaceBetween

    ){

        item {



            LazyColumn {

                items(items) { items ->

                    Column(modifier = Modifier.padding(top = 16.dp, start
=48.dp, bottom = 20.dp)) {

                        Text("Item_Name: ${items.itemName}")

                        Text("Quantity: ${items.quantity}")
```

```
                    Text("Cost: ${items.cost}")

                }

            }

        }

    }

}
```

ExampleInstrumentedTest.kt

```
package com.example.expensestracker

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
```

```
class ExampleInstrumentedTest

    {@Test

    fun useAppContext() {

        // Context of the app under test.

        val appContext =
InstrumentationRegistry.getInstrumentation().targetContext

        assertEquals("com.example.expensestracker", appContext.packageName)

    }

}
```

ExampleUnitTest.kt

```
package com.example.expensestracker


import org.junit.Test


import org.junit.Assert.*


/**

 • Example local unit test, which will execute on the development
machine(host).

 *

 • See [testing documentation](http://d.android.com/tools/testing).

 */

class ExampleUnitTest

    {@Test

    fun addition_isCorrect()

        {assertEquals(4, 2 +

        2)

    }

}
```

10.OUTPUT: