# MONEY MATTERS:

## A PERSONAL FINANCE MANAGEMENT APP

K.P.NAGADEVI

S.BHUVANESWARI

M.NANDHINI

T.TAMILSELVI

# 1. INTRODUCTION

## 1.1. Overview

A project of that app allows user to keep track of their expenses and accounts, provide an overview of their financial status. Users can set a budget for a various expenses and view their progress to it.

### Project workflow:

- User register into the application.
- After registration, User logins into the application
- Users enters into the main page.
- User can view the subject themes and selecting items and view records about it.

## 1.2. Purpose

- The purpose of the money matters apps could be to help individuals manage their personal finances such as budgeting

tools, expenses tracking, and financial education resources.

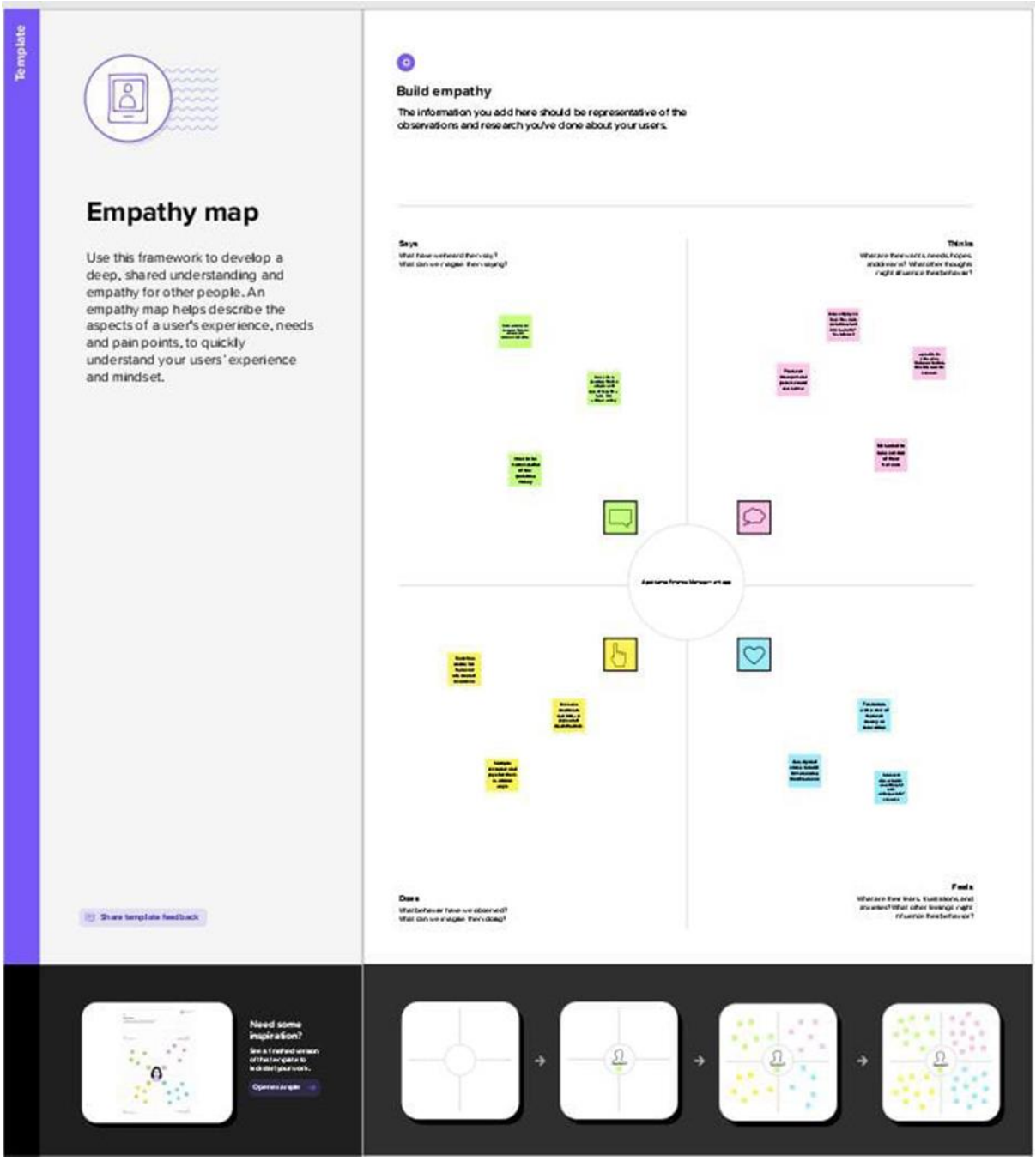
- User can better understanding of their spending habits, create and manage a budget, and track their financial progress over time. They can also receive notifications when bills are due or when they exceed their budget, which can help them avoid late fees and overspending.

- Additionally, the app may provide access to financial advice and resources, such as articles, videos, and the tutorials, that can help users improve their financial literacy and make better financial decisions.

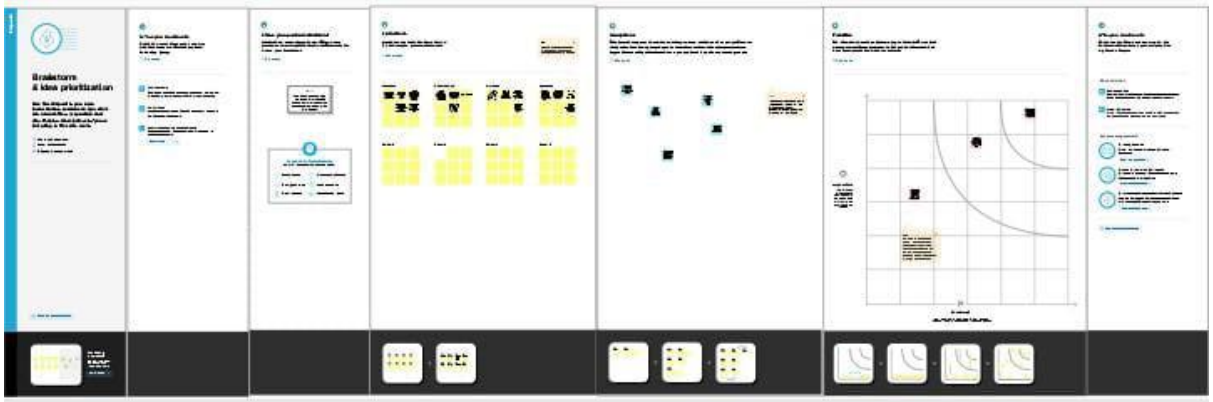
- Overall, the Money Matters app could be a useful tool for individuals who want to take control of their finances and improve their financial well-being.

# 2. Problem definition & design thinking

## 2.1 Empathy map



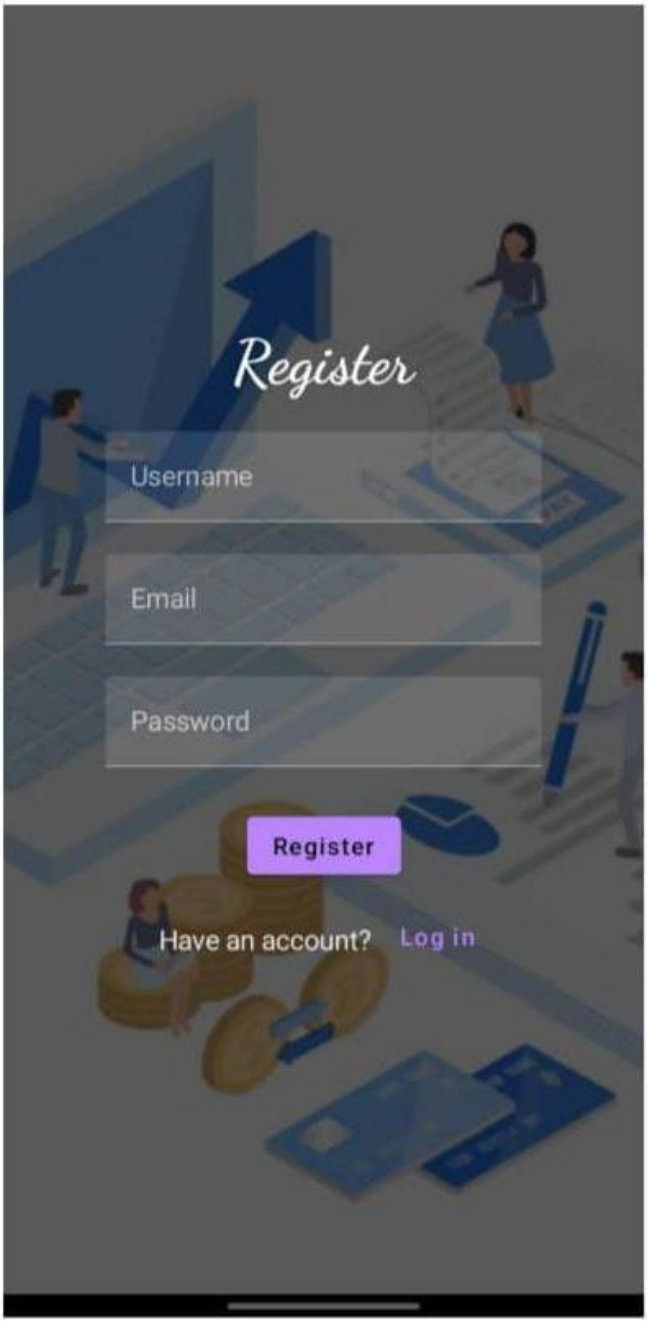
## 2.2 ideation & Brainstorming




# 3. RESULT

Login Page :


RegisterPage :


Register
Username
Email
Password
Register
Have an account?
Log in

MainPage :

# Welcome To
# Expense Tracker

Add Expenses | Set Limit | View Records

## 4. ADVANTAGES & DISADVANTAGES

### Advantages:

- Increased awareness of spending habits: Tracking expenses helps individuals become more aware of where their money is going and identify areas where they can cut back.
- Better budgeting: By tracking expenses, individuals can create a budget based on their actual spending habits and adjust their spending accordingly.
- Reduced financial stress: Keeping track of expenses can help individuals avoid overspending and the resulting financial stress that comes with it.
- Improved financial planning: Tracking expenses can help individuals plan for future expenses and set financial goals.

- Better decision-making: By having a clear understanding of their expenses, individuals can make informed decisions about their spending habits.

**Disadvantages:**

- Time-consuming: Maintaining an expense tracker requires time and effort, especially if done manually. If you have a busy schedule, it may be difficult to find the time to input all your expenses and categorize them.

- Potential for errors: Expense tracking can be prone to errors, particularly if done manually. Mistakes in data entry or categorization can lead to inaccurate financial reports and misinformed financial decisions.

- Limited scope: Expense trackers are designed to track expenses, but they may not give you a complete picture of your overall

financial health. You may need to use additional tools to track income, investments, and savings.

- May not change behavior: Even with an expense tracker, some people may still struggle to control their spending habits. It's important to use an expense tracker as part of a larger financial plan that includes budgeting, saving, and investing.

## 5. APPLICATION

Expenses Tracker app can be applied for a variety of applications including but not limited to:

### Expense Tracking:

Allow users to add, edit, and delete expenses. Users should be able to input details such as expense amount, date, category, and optional notes.

### Expense Categorization:

Implement the ability to categorize expenses into different categories, such as food, transportation, entertainment, etc. This can help users analyze their spending habits and create budgets.

### Budget Setting:

Allow users to set budgets for different expense categories or overall spending. The app can provide notifications or warnings when users approach or exceed their budget limits.

### Expense Reports:

Provide reports and summaries of expenses, such as monthly, yearly, or custom date range summaries. This can help users visualize their spending patterns and identify areas where they may need to cut back or adjust their expenses.

**Data Visualization:**

Implement graphical representations of expenses, such as charts or graphs, to provide users with a visual overview of their spending patterns.

**Search and Filtering:**

Allow users to search for specific expenses or filter expenses based on criteria such as category, date, or amount. This can help users quickly find and analyze specific expenses.

**User Accounts:**

Implement user accounts and authentication to allow users to securely log in and access their expenses across multiple devices. This can also provide data backup and synchronization features.

**Data Security:**

Implement proper data security measures, such as encryption and secure storage, to protect users' sensitive financial data.

## About Android studio Application:

●Android Studio is the official integrated development environment (IDE) for building Android apps. It was first released by Google in 2013 and has since become the most popular development environment for Android app developers.

●Android Studio is based on the INTELIJ IDEA community edition, and it includes many tools and features designed specifically for developing Android apps.

Some of the key features of Android Studio include:

## User Interface (UI) Designer:

Android Studio includes a powerful UI designer that allows developers to easily create and modify app layouts using drag-and-drop tools. The UI designer supports a variety of layouts, including linear, relative, and constraint layouts.

**Code Editor:**

Android Studio comes with a powerful code editor that offers syntax highlighting, code completion, refactoring, and debugging capabilities. It supports multiple programming languages, including Java, Kotlin, and C++, which are commonly used for Android app development.

**Emulator:**

Android Studio comes with a built-in emulator that allows developers to test their apps on virtual Android devices without needing physical devices. It supports various

Android versions and device configurations, making it convenient for testing app compatibility across different devices.

### Gradle Build System:

Android Studio uses the Gradle build system, which allows developers to manage app dependencies, build flavors, and create different app variants for release, debug, and testing purposes.

### Debugging and testing:

Android Studio has tools for debugging and testing Android apps, including a debugger, emulator, and integration with various testing frameworks.

### Version Control:

Android Studio supports version control systems like Git, allowing developers to easily manage their code changes and collaborate with other team members.

### Performance Profiling:

Android Studio includes performance profiling tools that allow developers to identify performance bottlenecks in their apps and optimize their code for better performance.

Overall, Android Studio is a powerful tool for developing high-quality Android apps. It provides a range of features and tools that make it easy for developers to build, test, and deploy their apps.

## 6. CONCLUSION

- In conclusion, an expense tracker app is a powerful tool for managing personal or business expenses in a convenient and efficient way. With the convenience of tracking expenses on-the-go using a mobile app, users can easily input and categorize

expenses in real-time, view spending patterns through visualizations, and generate reports for better financial analysis. Expense tracker apps often come with features such as automated expense tracking, budgeting, receipt scanning, and syncing across devices, making it easier to stay organized and gain insights into spending habits.

- Using an expense tracker app can lead to increased financial awareness, better financial decision-making, and improved financial management skills. Expense tracker apps can also help users track their progress towards financial goals, set reminders for bill payments, and generate reports for tax purposes, making it easier to stay on top of their finances and achieve financial objectives.

- While expense tracker apps can be a valuable tool, they should be used in

conjunction with good financial habits such as budgeting, saving, and responsible spending to achieve long-term financial success.

- An expense tracker app can be a valuable tool for managing expenses, gaining financial insights, and achieving financial goals. It's essential to choose a reliable app, use it in conjunction with good financial habits, and consistently track and analyze expenses to make informed financial decisions and improve overall financial health. Start using an expense tracker app today to take control of your finances and build a better financial future.

## 7. FUTURE SCOPE

The future scope of expense trackers is promising, with several potential advancements and opportunities for further development. Here are some potential future directions for expense trackers:

### Increased Automation:

As technology continues to advance, expense trackers could become even more automated. This could include features such as automatic transaction categorization and tagging, machine learning algorithms that can predict spending patterns and provide personalized financial insights, and integration with smart devices and Internet of Things (IOT) technologies to automatically capture and track expenses.

### Enhanced Data Analytics:

Expense trackers could leverage advanced data analytics techniques to provide users with more in-depth and meaningful insights into their spending habits. This could include data visualizations, trend analysis, and predictive analytics to help users identify spending patterns, optimize budgets, and make informed financial decisions

### Personalized Financial Guidance:

Future expense trackers could provide more personalized financial guidance based on individual financial goals, preferences, and financial literacy levels. This could include tailored recommendations on budgeting, saving, investing, and debt management, as well as personalized alerts and reminders to help users stay on track with their financial goals.

### Integration with Financial Institutions:

Expense trackers could further integrate with financial institutions, such as banks, credit card companies, and investment accounts, to provide users with a comprehensive view of their financial transactions and accounts in one centralized platform.

### Social Collaboration:

Expense tracker apps could enable social collaboration features, allowing users to share expenses and split costs with friends, family, or colleagues. This could be particularly useful for group expenses, travel expenses, or shared bills, making it easier to manage shared expenses and track contributions.

### Financial Education and Resources:

Expense tracker apps could offer built-in financial education resources, such as articles, videos, and tutorials, to help users improve their financial literacy and make more informed financial decisions. This could include topics such as budgeting, saving, investing, and debt management, providing users with valuable financial guidance and education within the app.

# 8. APPENDIX

## A. Source code

### 1. Creating the database classes

Step 1:

Create User data class

```
 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?,


    )
```

Step 2 :

Create an UserDao interface

```
 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)
```

```
}
```

Step 3 :

Create an UserDatabase class

```
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
        }
    }
  }
}
```

Step 4 :

Create an UserDatabaseHelper class

```
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(COL
UMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COL
UMN_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(COLUM
N_ID)),

                firstName =
cursor.getString(cursor.getColumnIndex(COL
UMN_FIRST_NAME)),

                lastName =
cursor.getString(cursor.getColumnIndex(COL
UMN_LAST_NAME)),

                email =
cursor.getString(cursor.getColumnIndex(COL
UMN_EMAIL)),
```

```
            password =
cursor.getString(cursor.getColumnIndex(COL
UMN_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(COLUM
N_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COL
UMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COL
UMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COL
UMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COL
UMN_PASSWORD)),
```

```
            )
            users.add(user)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return users
  }


}
```

## Database 2

### Step 1:

<u>Create Items data class</u>

```
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?,
)
```

Step 2:

Create ItemsDao interface

```
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)
}
```

Step 3:

Create ItemsDatabase class

```
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
    }
}
```

Step 4:

Create ItemsDatabaseHelper class

```
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(COLUM
N_ID)),
        itemName =
cursor.getString(cursor.getColumnIndex(COL
UMN_ITEM_NAME)),
        quantity =
cursor.getString(cursor.getColumnIndex(COL
UMN_QUANTITY)),
        cost =
cursor.getString(cursor.getColumnIndex(COL
UMN_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(COLUM
N_ID)),
```

```
            itemName =
cursor.getString(cursor.getColumnIndex(COL
UMN_ITEM_NAME)),
            quantity =
cursor.getString(cursor.getColumnIndex(COL
UMN_QUANTITY)),
            cost =
cursor.getString(cursor.getColumnIndex(COL
UMN_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(COLUM
N_ID)),
                    itemName =
cursor.getString(cursor.getColumnIndex(COL
UMN_ITEM_NAME)),
                    quantity =
cursor.getString(cursor.getColumnIndex(COL
UMN_QUANTITY)),
```

```
                cost =
cursor.getString(cursor.getColumnIndex(COL
UMN_COST)),
            )
            item.add(items)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return item
  }



}
```

## Database 3

### Step 1:

Create Expense data class

```
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?,
)
}
```

Step 2:

Create ExpenseDao interface

```
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)
}
```

Step 3:

Create ExpenseDatabase class

```
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
        }
    }
  }
}
```

Step 4:

<u>Create ExpenseDatabaseHelper class</u>

```
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? = null
        if (cursor.moveToFirst()) {
            expense = Expense(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

```
            amount =
cursor.getString(cursor.getColumnIndex(COL
UMN_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? = null
```

```
        if (cursor.moveToFirst()) {
            expense = Expense(
                id =
cursor.getInt(cursor.getColumnIndex(COLUM
N_ID)),
                amount =
cursor.getString(cursor.getColumnIndex(COL
UMN_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(COLUM
N_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(COLUM
N_ID)),
                    amount =
cursor.getString(cursor.getColumnIndex(COL
UMN_AMOUNT)),
                )
                expenses.add(expense)
            } while (cursor.moveToNext())
```

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



}
```

## Buliding application UI and connecting database

### Step 1:

Creating LoginActivity.kt with database

```
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.PasswordVisu
alTransformation

import
androidx.compose.ui.text.input.VisualTransfo
rmation

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.Expe
nsesTrackerTheme


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(userna
me)
                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.java

```
            )
        )}
        )
        { 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)
}
```

Step 2:

Creating RegisterActivity.kt with database

```
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.PasswordVisu
alTransformation

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.Expe
nsesTrackerTheme


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)
}
```

Step 3:

Creating MainActivity.kt file

```
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.Expe
nsesTrackerTheme


class MainActivity : ComponentActivity() {


@SuppressLint("UnusedMaterialScaffoldPadd
ingParameter")
   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,AddE
xpensesActivity::class.java))},
                    colors =
ButtonDefaults.buttonColors(backgroundColo
r = 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(backgroundColo
r = 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(backgroundColo
r = 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))


    }
}
```

Step 4:

Creating AddExpensesActivity.kt file

```
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("UnusedMaterialScaffoldPadd
ingParameter")
    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,AddE
xpensesActivity::class.java))},
                        colors =
ButtonDefaults.buttonColors(backgroundColo
r = 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(backgroundColo
r = 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(backgroundColo
r = 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(expe
nse)

itemsDatabaseHelper.insertItems(items)
                }
            }


        }
    }) {
```

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


    }
}
```

Step 5:

Creating SetLimitActivity.kt file

```
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.LazyColum
n
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.Expe
nsesTrackerTheme

class SetLimitActivity : ComponentActivity() {
    private lateinit var expenseDatabaseHelper:
ExpenseDatabaseHelper

@SuppressLint("UnusedMaterialScaffoldPadd
ingParameter")
    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(backgroundColo
r = 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(backgroundColo
r = 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(backgroundColo
r = 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(expen
se)
            }
        }) {
            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)
                        }
                    }
                }
```

```
            }


        }
    }
}



//@Composable
//fun Records(expense: List<Expense>) {
//    Text(text = "View Records", modifier =
Modifier.padding(top = 24.dp, start = 106.dp,
bottom = 24.dp ), fontSize = 30.sp)
//    Spacer(modifier = Modifier.height(30.dp))
//    LazyRow(
//        modifier = Modifier
//            .fillMaxSize()
```

```
//            .padding(top = 80.dp),
//
//        horizontalArrangement =
Arrangement.SpaceBetween
//    ){
//        item {
//
//            LazyColumn {
//                items(expense) { expense ->
//                    Column(modifier =
Modifier.padding(top = 16.dp, start = 48.dp,
bottom = 20.dp)) {
//                        Text("Remaining Amount:
${expense.amount}")
//                    }
//                }
//            }
```

```
//      }
//
//   }
//}
```

Step 6:

Creating ViewRecordsActivity.kt file

```
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.LazyColum
n
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.Expe
nsesTrackerTheme


class ViewRecordsActivity :
ComponentActivity() {

    private lateinit var itemsDatabaseHelper:
ItemsDatabaseHelper


@SuppressLint("UnusedMaterialScaffoldPadd
ingParameter", "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(backgroundColo
r = 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(backgroundColo
r = 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(backgroundColo
r = 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)
```

```
            }
        }
    }
}
```

## Modifying AndroidManifest.xml

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

xmlns:tools="http://schemas.android.com/to
ols">

    <application
        android:allowBackup="true"
```

```
android:dataExtractionRules="@xml/data_extraction_rules"

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=".LoginActivity"
            android:exported="true"
            android:label="ExpensesTracker"
```

```
android:theme="@style/Theme.ExpensesTrac
ker" />
        <activity
            android:name=".RegisterActivity"
            android:exported="false"
            android:label="ExpensesTracker"

android:theme="@style/Theme.ExpensesTrac
ker" />
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="ExpensesTracker"

android:theme="@style/Theme.ExpensesTrac
ker" />
```

```
        <activity
            android:name=".AddExpensesActivity"
            android:exported="false"

android:label="@string/title_activity_add_ex
penses"

android:theme="@style/Theme.ExpensesTrac
ker" />
        <activity
            android:name=".SetLimitActivity"
            android:exported="false"

android:label="@string/title_activity_set_limi
t"

android:theme="@style/Theme.ExpensesTrac
ker"/>
```

```
        <activity
            android:name=".ViewRecordsActivity"
            android:exported="false"
            android:label="ExpensesTracker"

android:theme="@style/Theme.ExpensesTrac
ker">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN"
/>

                <category
android:name="android.intent.category.LAU
NCHER" />
            </intent-filter>
        </activity>
```

```
    </application>


</manifest>
```