1.INTRODUCTION

The main task is to make each and everyone to know how much we are spending, our life is become so easy with the help of wireless transactions as everything is just a tap for success, what we are overlooking is that it's our hard-earned money.

If we are cautious on how much we are spending on every transaction will definitely make us aware and we will definitely reduce our usage. Knowing this we can spend our money but we can save a lot when we track our spending's.

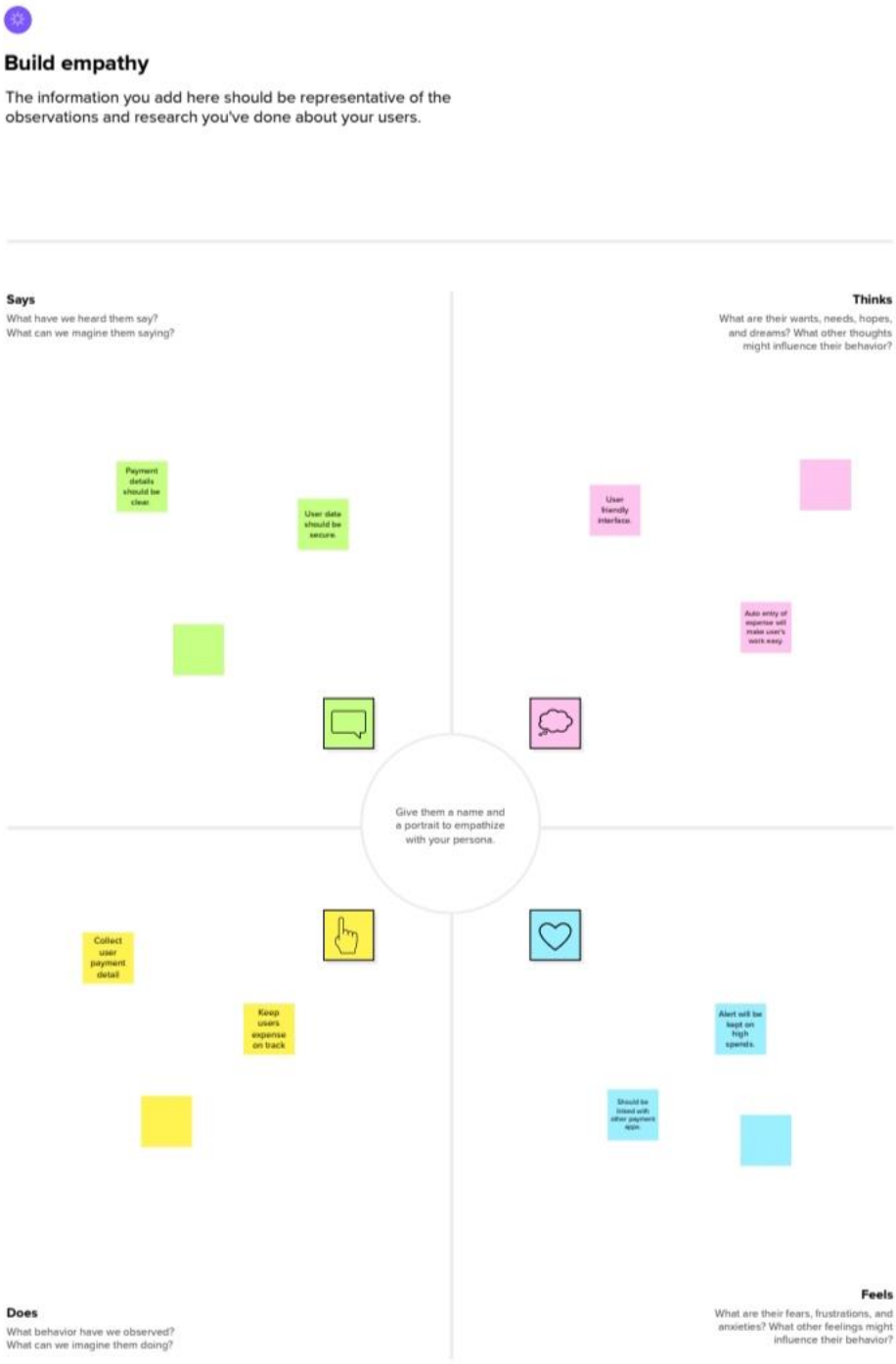
This project is a completely a standalone software as it can run on any platforms with the required packages being installed in the system in advance.

Many functions are available in our software like Adding an Expense, Deleting the expense, Even giving a sentence of how much we spend and to whom we spent.

We can also check the clear detail about a particular expense. So, in our project we are using Kotlin and Jetpackcompose.

2.PROBLEM DEFINITION & DESIGN THINKING

2.1Empathy Map



2.2 Brainstorming Map

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Vishnu E M

UI/UX must be easy

Transaction mode on every other app should be noted

Should keep track on categories separately

Sheik Md. Fahim T F

Make sure to give alerts if we are spending a lot

Saving and Filtration should be easy and understandable

We should be able to enter manually incase if we spent on cash

Kavithasan S

Should keep track from available amount only

Saving at least minimum amount should be mandatory

Mahesh V

The app should suggest ways on how to save amount

Should maintain various varieties of categories

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Savings:

User Interface:

Permissions:

Categories

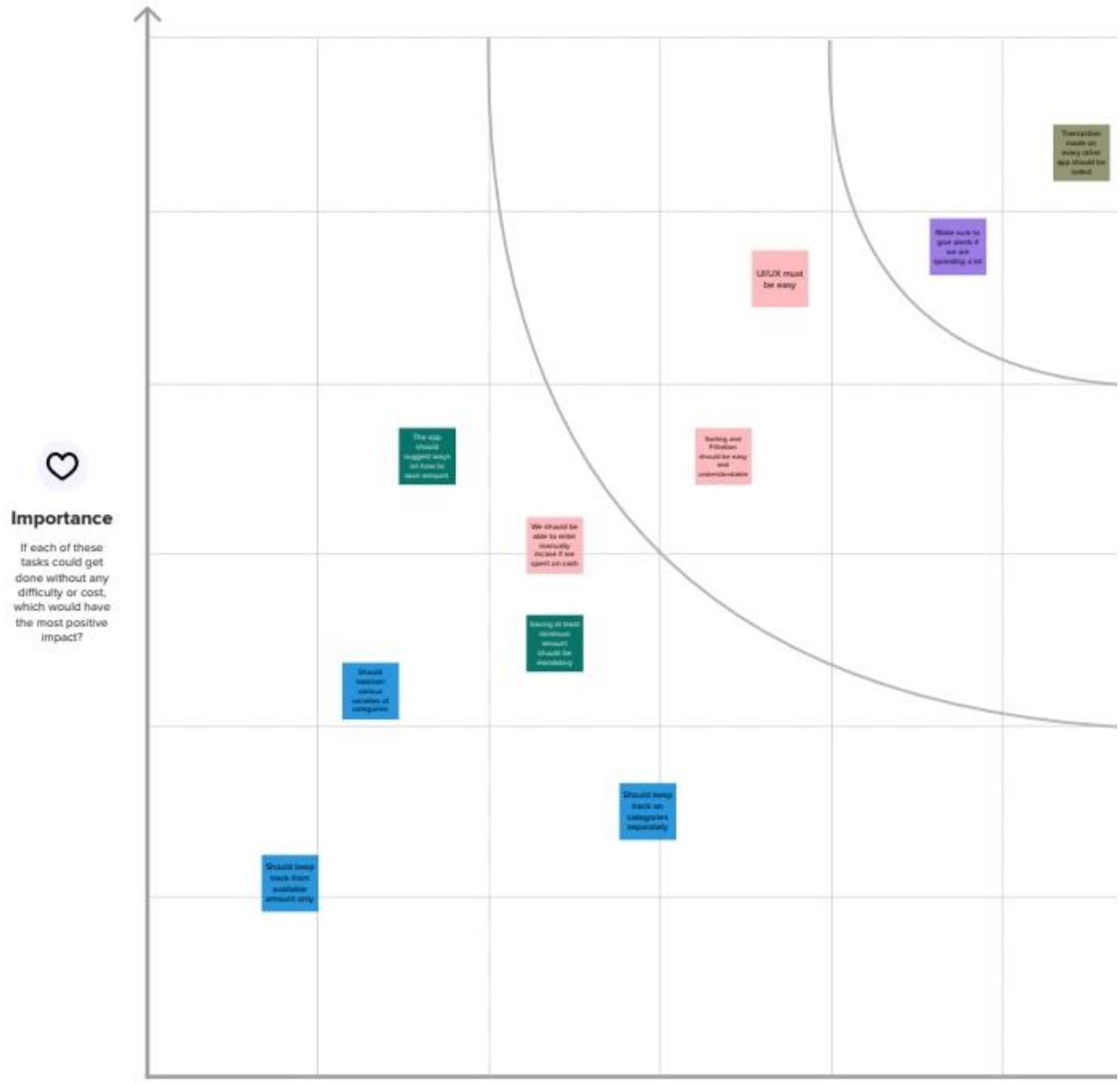
Alerts:

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⏱ 20 minutes



Importance

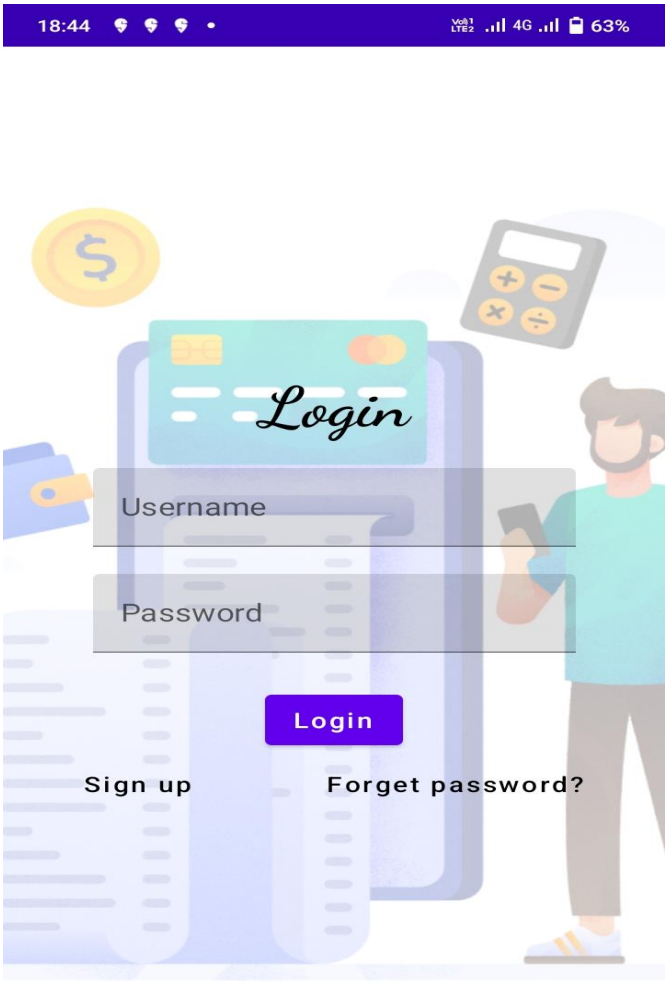
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

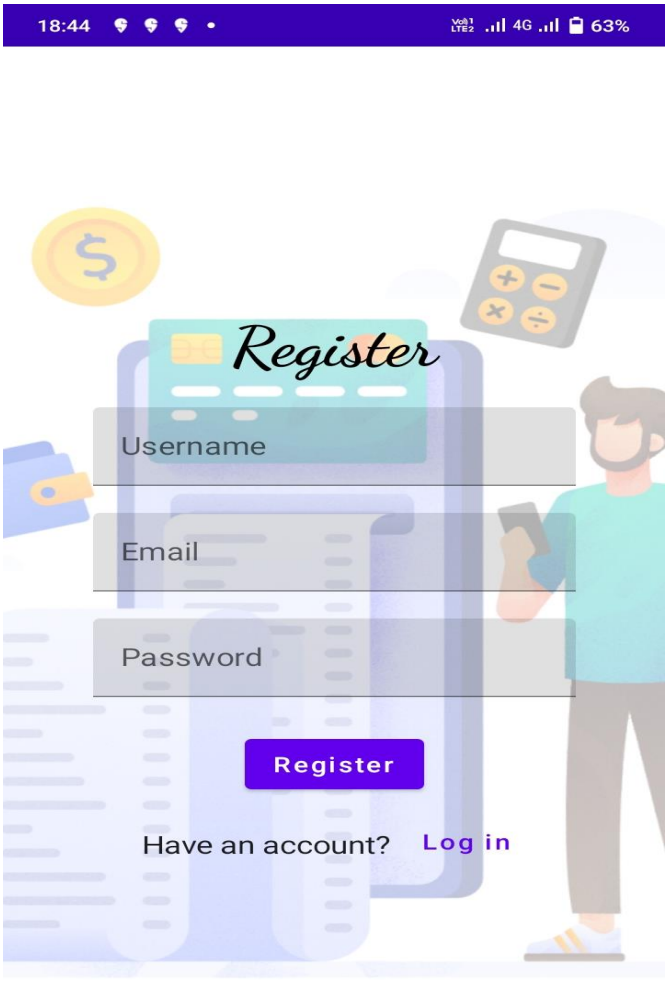
Regardless of their importance, which tasks are more

3. Results

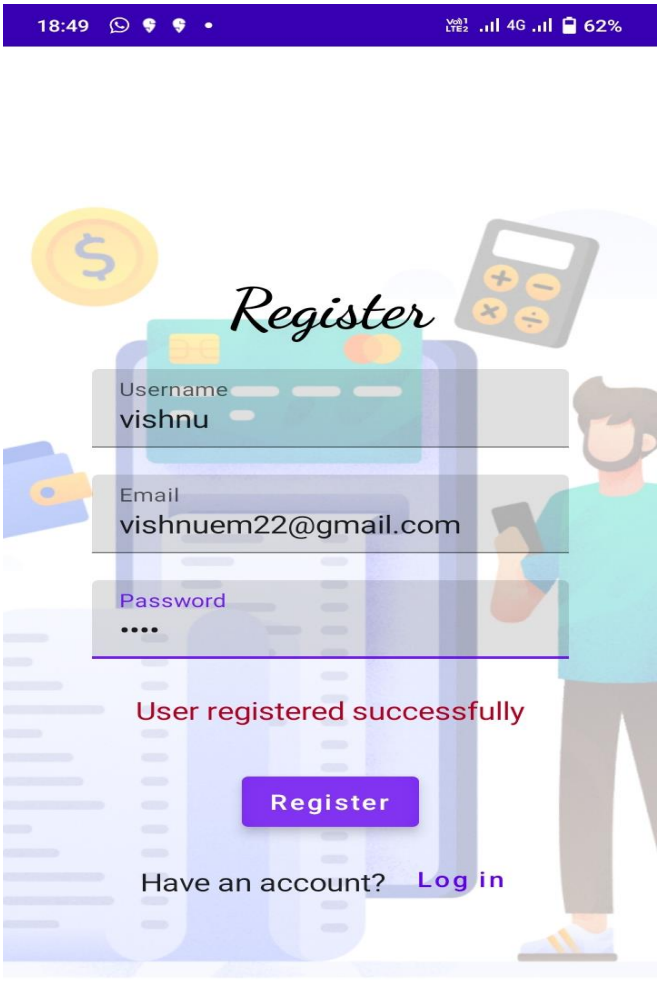
Output



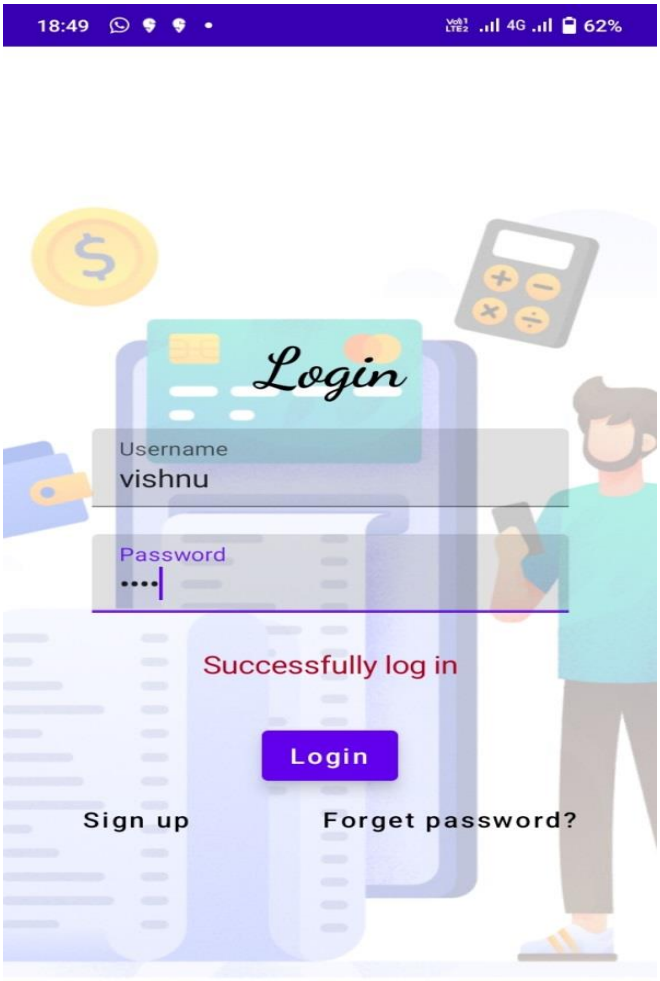
1. LOGIN SCREEN



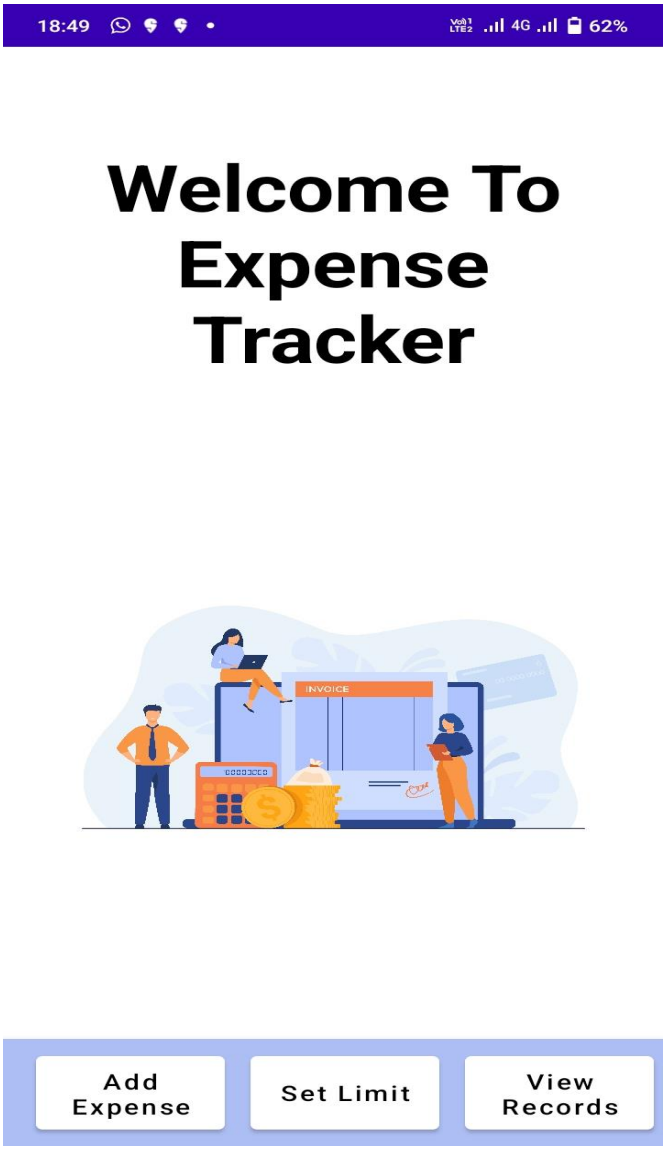
2. REGISTER SCREEN



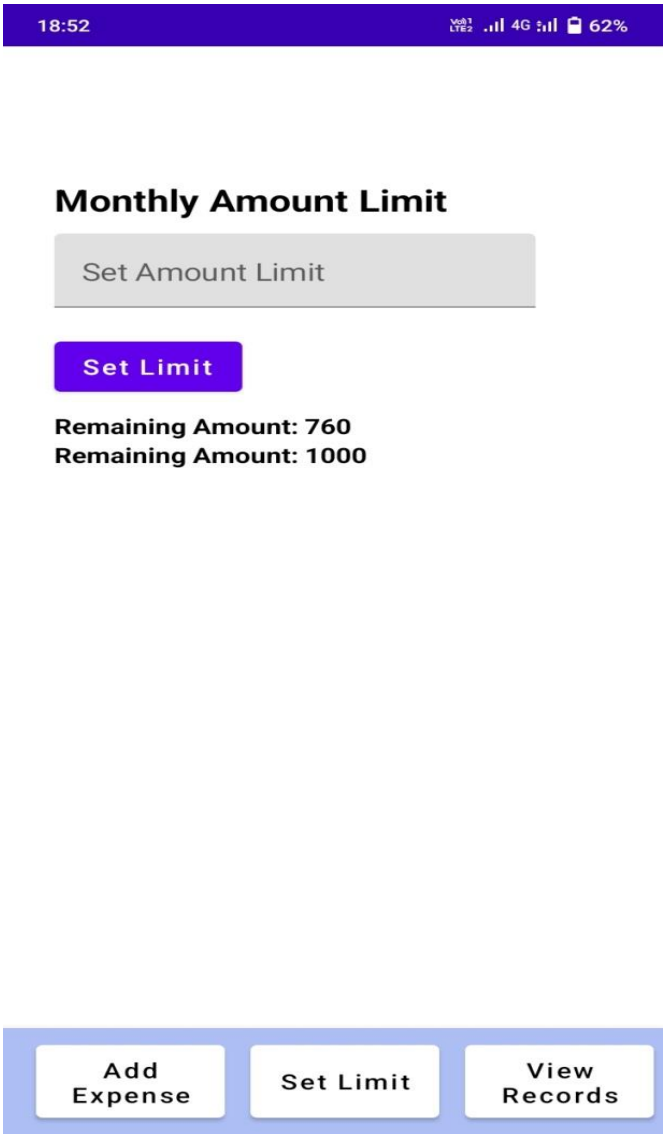
3. USER REGISTRATION SUCCESSFULL SCREEN



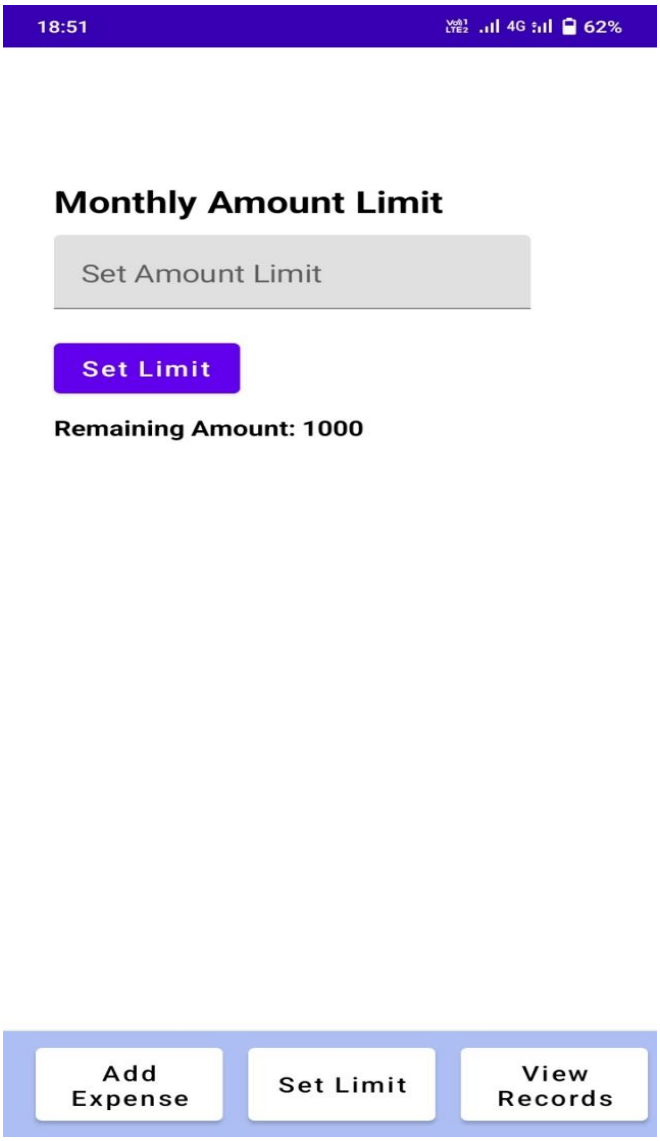
4. LOGIN SCREEN WHEN ITS SUCCESSFUL



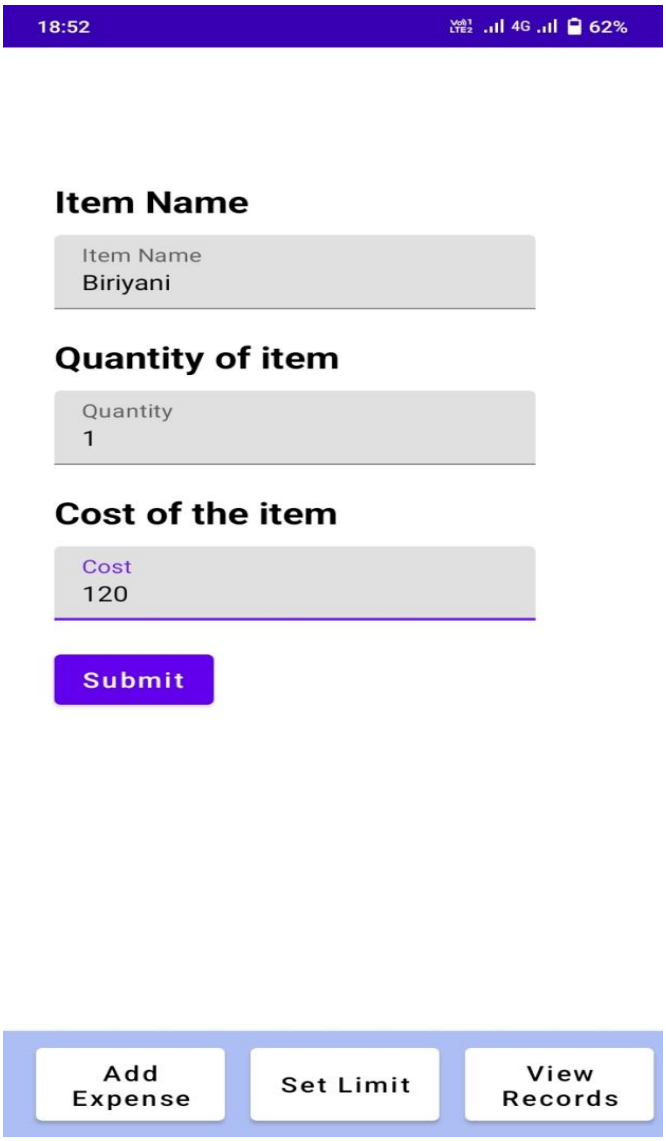
5. MAIN SCREEN



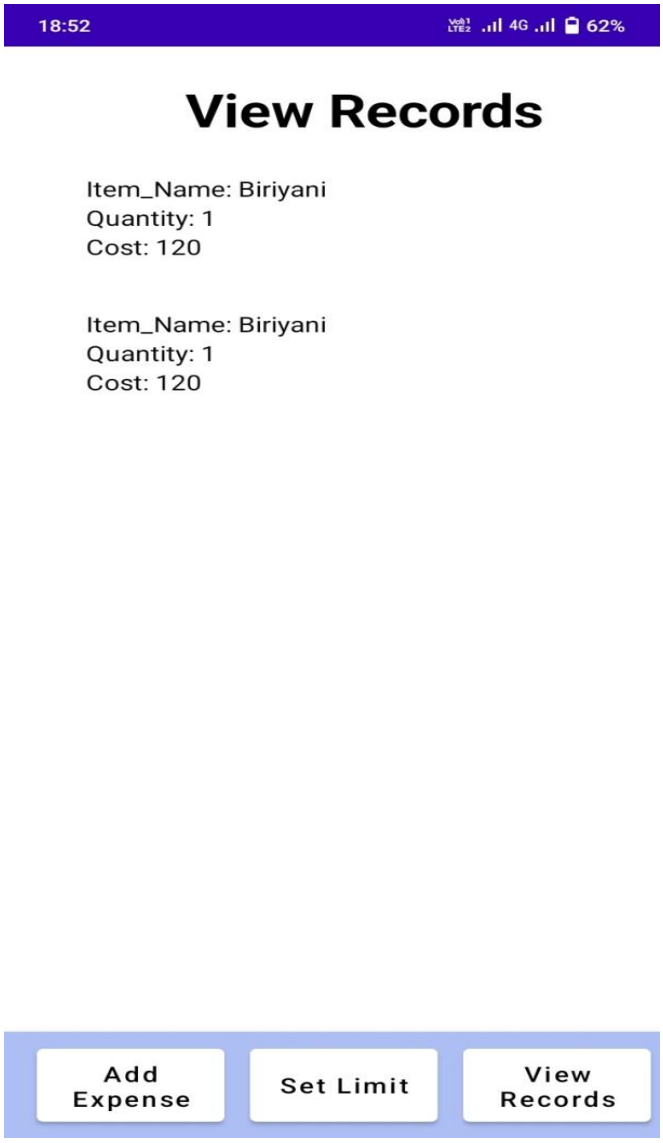
6. SET LIMIT SCREEN AFTER ENTERING VALUES



7. SET LIMIT SCREEN WHEN NO EXPENSE IS ADDED



8. ADD EXPENSE SCREEN



9. VIEW RECORD SCREEN

4. ADVANTAGES AND DISADVANTAGES

Expense tracking is a practice of monitoring and recording all the money that you spend over a period of time. An expense tracker can help you to keep a record of your spending, so you can easily see where your money is going and make adjustments as needed. In this article, we will discuss the advantages and disadvantages of using an expense tracker.

4.1ADVANTAGES:

- **Helps to manage finances:**

An expense tracker helps you to manage your finances effectively. You can keep track of your income, expenses, and savings in one place. With this information, you can make better decisions about how to spend your money.

- **Helps to control expenses:**

When you keep a record of all your expenses, you can easily see where your money is going. This can help you to identify areas where you are overspending and make changes to your budget to save money.

- **Encourages better spending habits:**

Expense tracking can encourage better spending habits. When you know you are tracking your expenses, you are more likely to think twice before making a purchase. This can help you to avoid impulsive buying and focus on spending on things that matter.

- **Helps to save money:**

An expense tracker can help you to save money. By tracking your expenses, you can identify areas where you are overspending and make changes to

your budget to save money. This can help you to achieve your financial goals faster.

- **Makes tax time easier:**

An expense tracker can make tax time easier. You can easily identify tax-deductible expenses and keep all the necessary documents in one place. This can help you to file your taxes quickly and accurately.

4.2 DISADVANTAGES:

- **Requires discipline:**

An expense tracker requires discipline. You need to be consistent in tracking your expenses to get an accurate picture of your spending habits. If you are not consistent, the data will not be useful, and you will not be able to make informed decisions about your finances.

- **Time-consuming:**

Expense tracking can be time-consuming. You need to record all your expenses, categorize them, and keep them up to date. If you have a busy schedule, it can be challenging to find the time to do this regularly.

- **Can be overwhelming:**

Expense tracking can be overwhelming, especially if you have never done it before. There are many different expenses to track, and it can be challenging to categorize them correctly. This can lead to confusion and frustration.

- **Can be expensive:**

Expense tracking tools can be expensive. Some software requires a subscription fee or a one-time payment. If you are on a tight budget, this can be a significant expense.

- **Can be misleading:**

Expense tracking can be misleading if not done correctly. If you forget to record an expense, it can throw off your budget, and you may think you have more money than you do. This can lead to overspending and financial problems.

5. APPLICATIONS

Expense Tracker app is a mobile application designed to help individuals keep track of their expenses. This app provides an efficient way of managing finances, tracking expenses, and organizing financial data in a single place. Here are some of the uses of the Expense Tracker app.

- **Budget Management**

Expense Tracker app allows individuals to create a budget and stick to it. This feature helps in managing finances, as it allows users to allocate a specific amount of money to each expense category, such as housing, food, entertainment, and transportation. By tracking the expenses in each category, users can determine if they are staying within their budget or overspending.

- **Expense Tracking**

Expense Tracker app provides an easy way of tracking expenses. It allows users to log expenses in different categories, such as groceries, bills, rent, and other expenses. Users can also add notes to each expense to provide more context or details about the transaction. By tracking expenses, users can identify areas where they can cut back on spending and save more money.

- **Goal Setting**

Expense Tracker app enables users to set financial goals, such as saving for a vacation, buying a car, or paying off debt. By setting financial goals, users can prioritize their spending and save money towards achieving their objectives. The app also provides progress reports, which enable users to monitor their progress towards achieving their financial goals.

- **Bill Reminders**

Expense Tracker app helps users stay on top of their bills. Users can set reminders for their bills, such as rent, utilities, and other recurring expenses. The app sends notifications when a bill is due, ensuring that users do not miss any payments and avoid late fees.

- **Tax Preparation**

Expense Tracker app simplifies tax preparation. It allows users to export their financial data in different formats, such as PDF, CSV, and Excel. The exported data can be used to prepare tax returns or to provide financial information to an accountant.

- **Financial Planning**

Expense Tracker app provides financial planning tools that enable users to plan for their future. The app allows users to set financial targets, such as saving for retirement or buying a house. It also provides tools to calculate how much money users need to save each month to reach their financial goals.

- **Family Budgeting**

Expense Tracker app can be used to manage family budgets. It allows users to create multiple budgets for different family members, track expenses, and allocate funds accordingly. This feature enables families to manage their finances collectively, ensuring that everyone stays on track with their financial goals.

In conclusion, Expense Tracker app is a useful tool for managing finances, tracking expenses, and organizing financial data. It provides features that enable users to set financial goals, stick to a budget, track expenses, and manage bills.

6. CONCLUSION

An expense tracker can be a useful tool for managing your finances, but it also has some disadvantages. It requires discipline and can be time-consuming and overwhelming. It can also be expensive and misleading if not done correctly. However, the benefits of expense tracking outweigh the disadvantages. It can help you to manage your finances effectively, control your expenses, and save money. If you are considering using an expense tracker, it is essential to weigh the pros and cons and choose a tool that works for you.

With the expense tracker it would be easy for us to keep track on how we are spending and where we are spending. We can get aware on where we spend a lot and where we spend less.

The Login page makes it much safer as others cannot see our data on how and where we spent.

Each and every function is much unique and plays a different role in our easy maintenance expense tracker. The functions hold a capability that this expense tracker would be nothing without it.

We can use this easy maintenance expense tracker in many ways like for a small business as they could easily keep track of the expenses they are making on each day. Also this can be used by individuals so that they might save there each and every expense clearly with the different mode of payments we provide.

7. FUTURE SCOPE

Our money matters app come with a way of holding record of what we have spent with the limit which we have set. We cannot use it if the user has used his limit fully. We might be looking forward to add more pictorial representation of our spending to give a deeper understanding to the user.

Pictorial representations like Pie chart and even options to compare spending month wise and day wise, which will make the user to understand easily about what we are trying to convey to them.

We might also be looking forward to add auto add option where the app gets the permission of messages to collect data and automatically read the expense that is done using UPI or any payment mode to make the work easy for the user.

Savings always plays an important role is a humans life. So we will be looking forward to give tips to the user according to the method of how he/she is spending the amount and where he can save it.

While Registering we will be looking forward to add various login methods so that the user will not find it difficult and most importantly to protect their data. User's personal details are much confidential and we will be working on it.

8. APPENDIX

8.A Source Code:

a. Gradle.build

```
buildscript {
    ext {
        compose_ui_version = '1.2.0'
    }
}// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id 'com.android.application' version '7.4.1' apply false
    id 'com.android.library' version '7.4.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.7.0' apply false
}
```

b. 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.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)
}
```

b. Main Activity:

```
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))
  }
}
```

c. 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)
}
```

d. Setlimit.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)
                        }
                    }
                }
            }

        }
    }
}
@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}")
                    }
                }
            }
        }

    }
}
```

e. AddExpense.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")
    }
  }
}
```

f. 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}")
                    }
                }
            }
        }
    }
}
```

g. Expense.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?,
)
```

h. ExpenseDoa.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)
}
```

I. 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
            }
        }
    }
}
```

j. 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? = null
        if (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? = null
    if (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
    }
}
```

k. 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?,
)
```

l. 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)
}
```

m. 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
            }
        }
    }
}
```

n.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
    }


}
```

o. 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?,


    )
```

p. 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)
}
```

q. 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
            }
        }
    }
}
```

r. 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
    }


}
```