1.INTRODUCTION

1.1 OVERVIEW:

Managing your money isn’t the easiest thing to do. Now that many of us no longer balance a checkbook, tracking expenses and keeping up with the bank balance can be difficult. Personal finance apps connect with your bank account and help you keep up with your spending. These apps can show you which categories you spend the most in, track, and, sometimes allow you to make upcoming bill payments, as well as keep up with your credit score and investment portfolio.

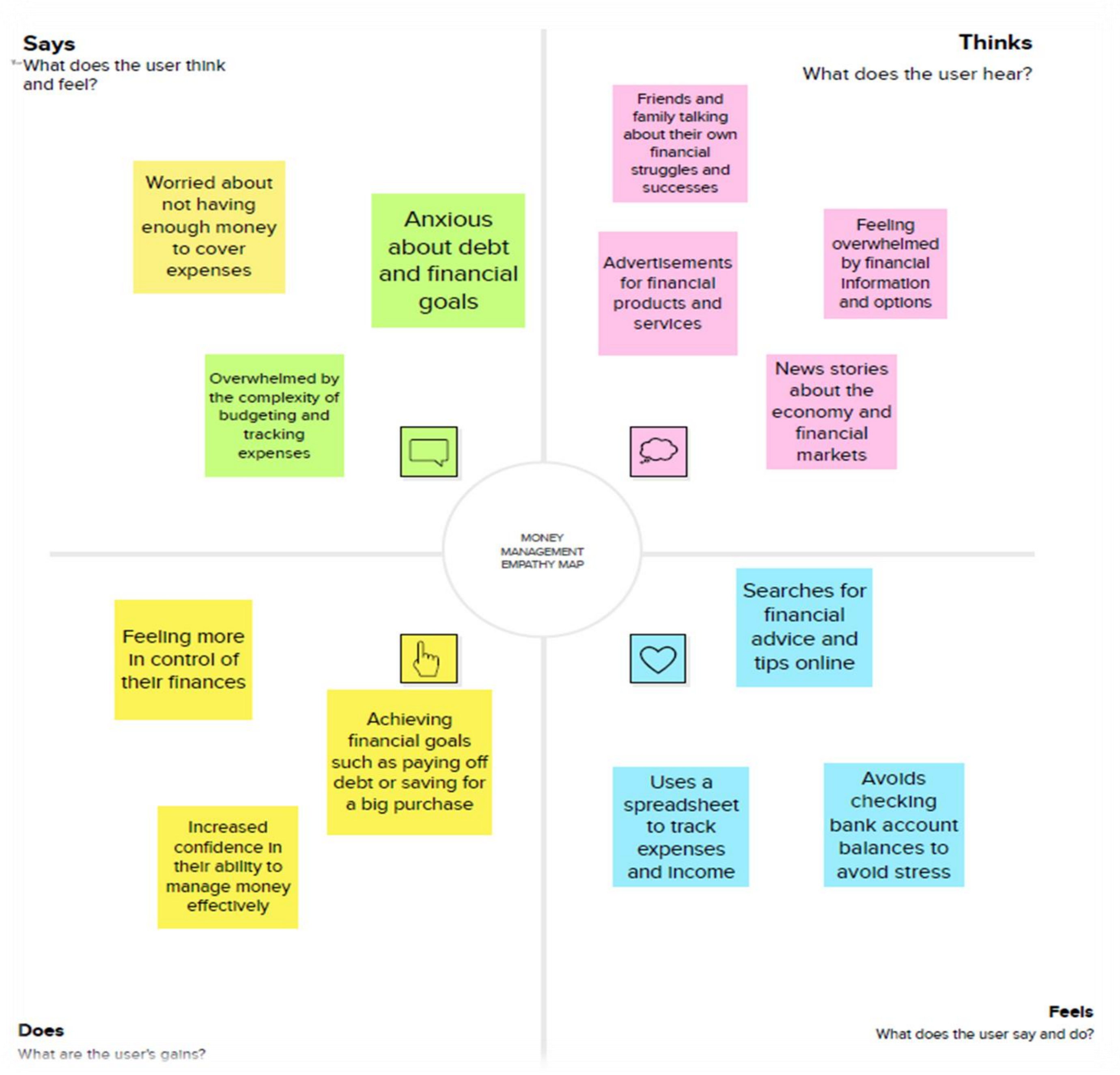
The best personal finance apps provide several features for managing your overall finances. This includes email reminders, bill due dates, track subscriptions, shared wallets, and more. All the apps on our list are available on both iOS and Android, so you can utilize them no matter which smartphone you have.

2.2 PURPOSE:

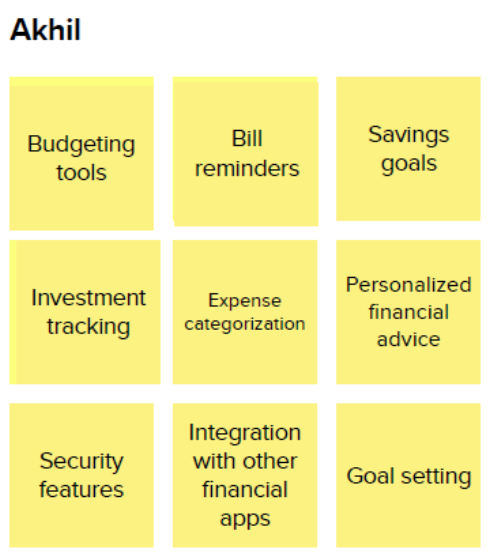
A financial management app is a useful tool to help you drive your personal financial strategy. The most immediate benefit of using a money management app is to help you stick to your budget. It allows you to track your spending and manage your cash flow on a daily basis, helping you move closer to your financial goals.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy map:

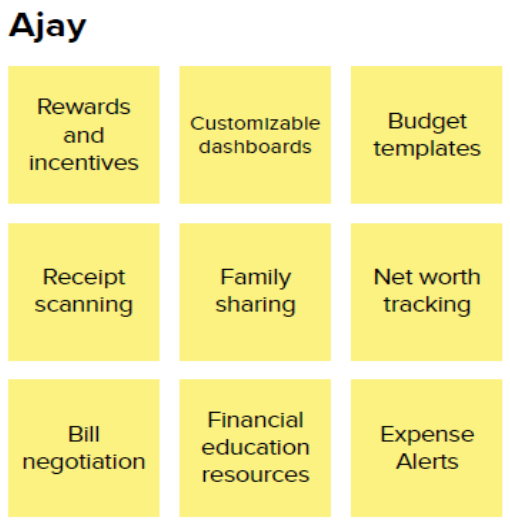

Says
What does the user think and feel?
Worried about not having enough money to cover expenses
Anxious about debt and financial goals
Overwhelmed by the complexity of budgeting and tracking expenses
Thinks
What does the user hear?
Friends and family talking about their own financial struggles and successes
Advertisements for financial products and services
Feeling overwhelmed by financial information and options
News stories about the economy and financial markets
MONEY MANAGEMENT EMPATHY MAP
Feeling more in control of their finances
Achieving financial goals such as paying off debt or saving for a big purchase
Increased confidence in their ability to manage money effectively
Does
What are the user's gains?
Searches for financial advice and tips online
Uses a spreadsheet to track expenses and income
Avoids checking bank account balances to avoid stress
Feels
What does the user say and do?

2.2 Brainstorm:



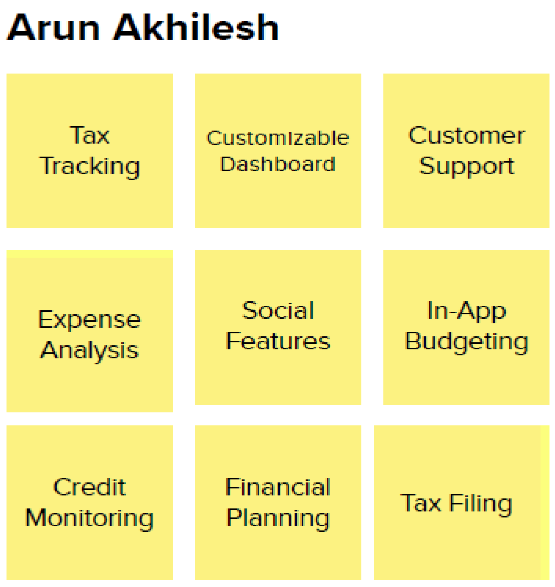
Akhil

Budgeting tools	Bill reminders	Savings goals
Investment tracking	Expense categorization	Personalized financial advice
Security features	Integration with other financial apps	Goal setting



Ajay

Rewards and incentives	Customizable dashboards	Budget templates
Receipt scanning	Family sharing	Net worth tracking
Bill negotiation	Financial education resources	Expense Alerts


Arun Akhilesh

Tax Tracking

Customizable Dashboard

Customer Support

Expense Analysis

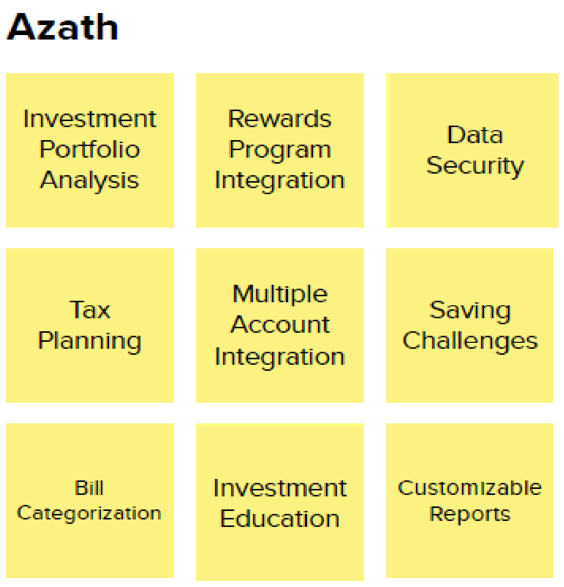
Social Features

In-App Budgeting

Credit Monitoring

Financial Planning

Tax Filing


Azath

Investment Portfolio Analysis

Rewards Program Integration

Data Security

Tax Planning

Multiple Account Integration

Saving Challenges

Bill Categorization

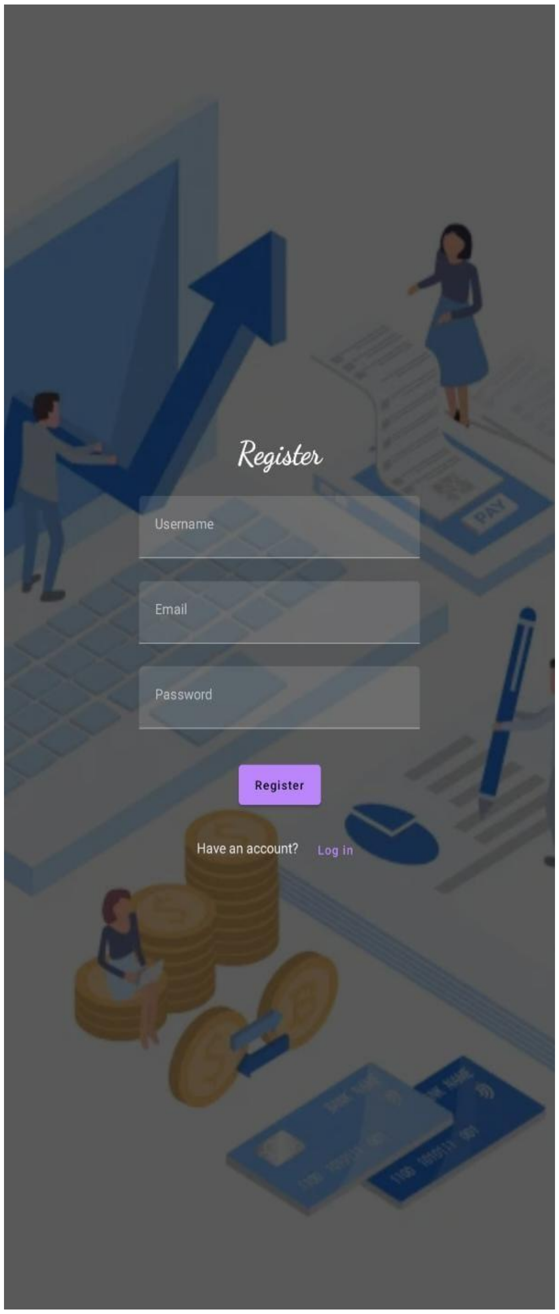
Investment Education

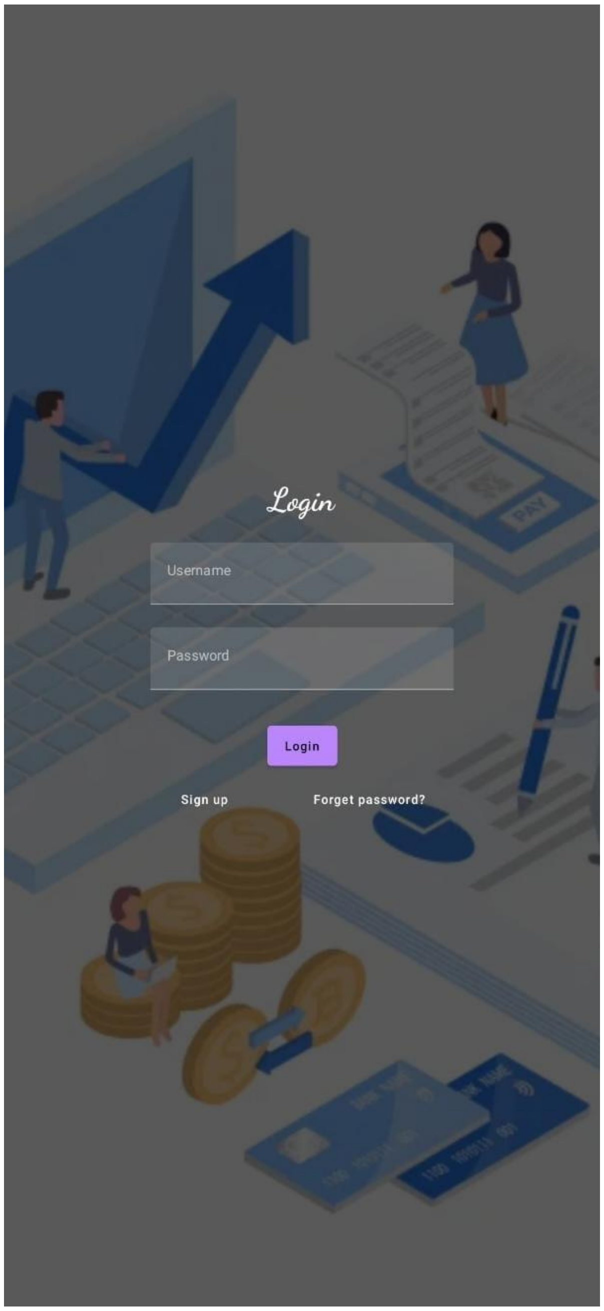
Customizable Reports

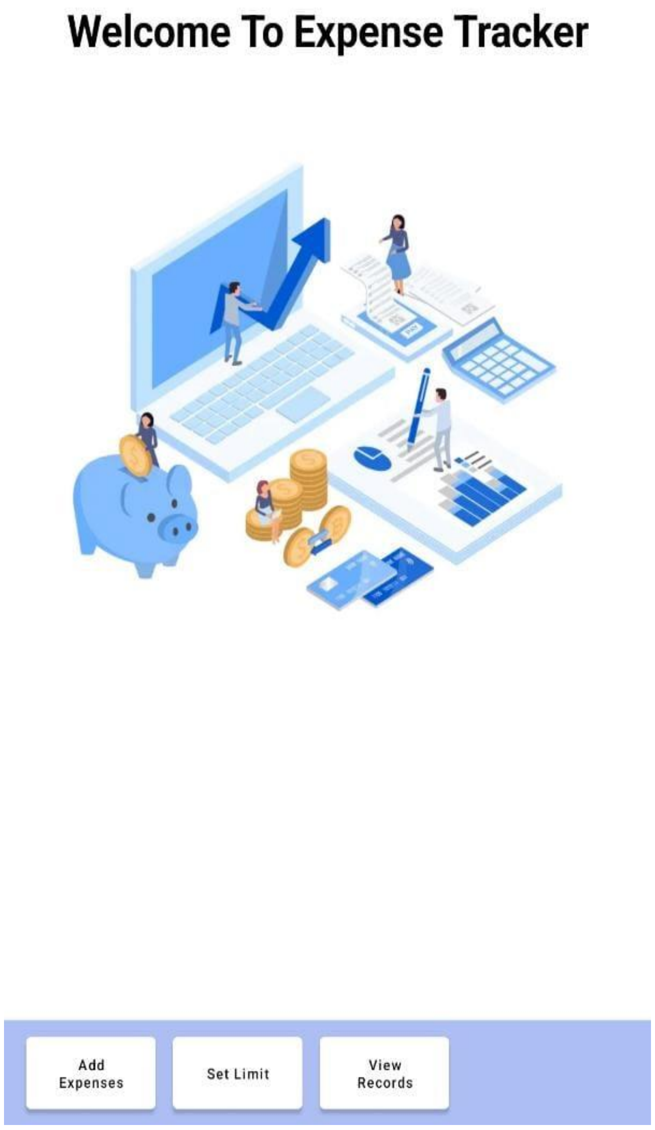
3.RESULT

OUTPUT:

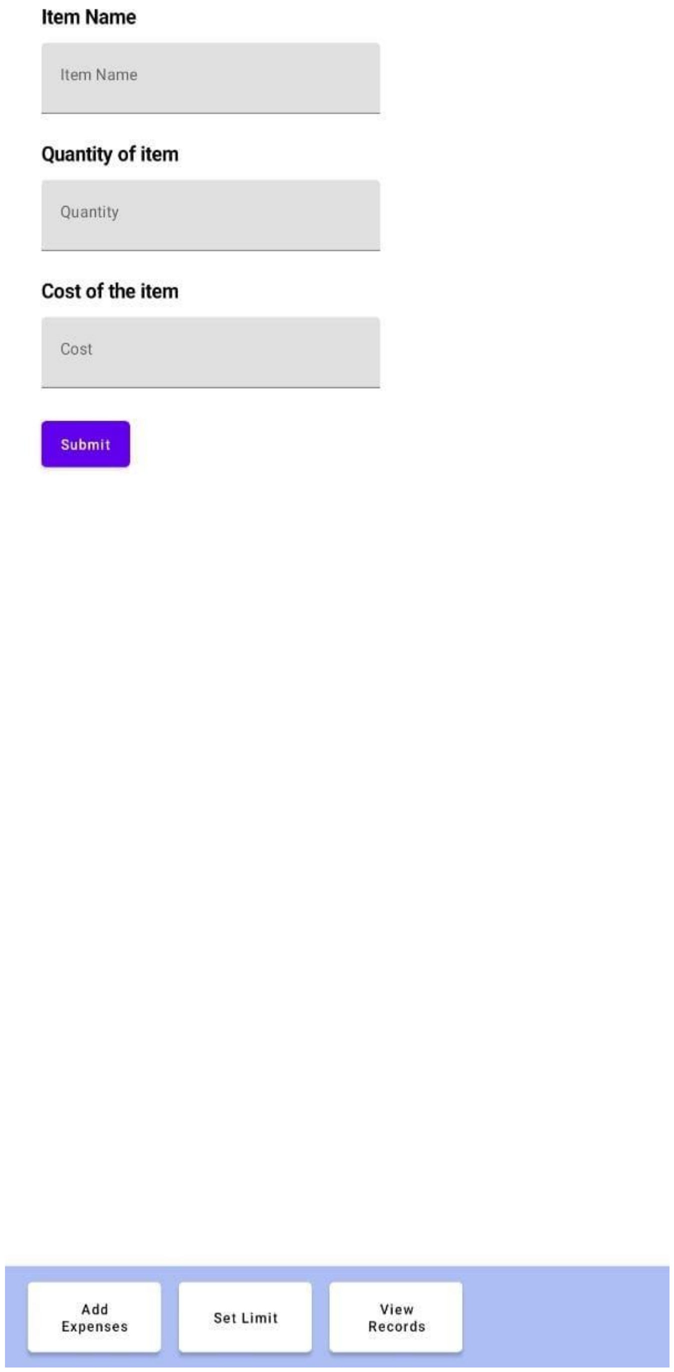
REGISTER ACTIVITY & LOGIN ACTIVITY:







Welcome To Expense Tracker



Item Name

Item Name

Quantity of item

Quantity

Cost of the item

Cost

4.ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

1. Access info from anywhere

Probably the biggest benefit of financial apps is their accessibility. Whether at work, traveling, or far from a computer, users can make changes and updates right from their mobile device, as long as they have an internet connection.

2. Deposit checks

Many bank apps now allow users to deposit a physical check through the app. You simply take a photo of the front and back of the check, upload it, and the deposit is submitted. This saves a lot of time instead of having to do it in person.

3. Save time

It can be quite a pain to have to travel to a bank, which is usually only open during regular working hours, just to make a transaction or view an account balance. With a financial app, everything can be done whenever a user thinks about it, and they don’t have to go anywhere. You don’t have to wait in long lines at the bank just to make a deposit or an online payment.

4. Stay organized

When people can keep tabs on their finances daily with information right at their fingertips, they’re more likely to stay on top of personal finances and stay organized. Apps allow for nearly real-time accuracy so there’s never any question about an account balance. All income and expenses can be monitored with ease and efficiency.

DISADVANTAGES:

Although money management apps can be helpful in managing personal finances, they also have some disadvantages. Here are a few potential drawbacks:

1. Inaccurate data: Money management apps rely on data input by users, and there's always the possibility of errors. If you don't enter your transactions correctly or forget to log a purchase, your financial data may be inaccurate.
2. Security concerns: Money management apps often require access to your bank account and other financial information. While many apps use encryption and other

security measures to protect your data, there is always a risk of a security breach.

3. Cost: While there are free money management apps available, many offer premium features for a fee. The cost of these apps can add up over time, especially if you're paying for multiple apps.
4. Dependence on technology: If your phone or computer crashes or you lose access to your money management app for any reason, you may not have immediate access to your financial information.
5. Lack of human interaction: Money management apps can be helpful, but they don't offer the same level of human interaction as a financial advisor. Without personalized advice, you may miss out on opportunities to improve your financial situation.

5. APPLICATIONS

Money management apps are becoming increasingly popular as they help people track their spending, budget effectively, and achieve their financial goals. Some of the key applications of money management apps include:

1. Budgeting: Money management apps allow you to set a budget for different categories of expenses, such as groceries, rent, entertainment, and transportation. You can then track your spending and see how much money you have left in each category.
2. Expense tracking: With a money management app, you can easily track your expenses and see where your money is going. This can help you identify areas where you might be overspending and make adjustments to your budget accordingly.
3. Saving money: Money management apps can help you save money by setting goals and tracking your progress towards those goals. For example, you can set a goal to save a certain amount of money each month for a vacation or a down payment on a house.
4. Investing: Some money management apps also allow you to invest your money and track your investment portfolio. This can help you grow your wealth over time.
5. Bill payment: Many money management apps allow you to link your bank accounts and credit cards, making it easy to pay bills and track your payments.

Overall, money management apps can help you take control of your finances, save money, and achieve your financial goals.

6. CONCLUSION

In conclusion, money management apps have become an essential tool for individuals to manage their finances effectively. They allow users to track their expenses, budget efficiently, save money, invest, and pay bills conveniently. By providing a clear overview of their financial situation, these apps can help users identify areas where they can cut back and save money, making it easier to achieve their long-term financial goals. With the increasing popularity of money management apps, it is evident that they have become an indispensable tool for individuals looking to take control of their finances and secure their financial future

7. FUTURE SCOPE

The future of money management apps looks promising as technology continues to evolve, and people become more conscious of their financial well-being. Here are some possible future trends for money management apps:

1. Increased personalization: Money management apps may become more personalized, with recommendations tailored to individual users' financial goals and spending patterns.
2. Use of AI and Machine Learning: Artificial intelligence and machine learning may be used to analyze spending patterns and provide users with personalized recommendations for saving money or investing.
3. Integration with other services: Money management apps may become integrated with other financial services, such as credit monitoring and financial planning tools, to provide users with a comprehensive view of their finances.
4. Improved security: With the increasing use of mobile devices for financial transactions, money management apps may focus on enhancing security measures to protect users' financial data.
5. Integration with smart devices: Money management apps may integrate with smart devices, such as wearables and smart home devices, to provide users with real-time alerts and notifications about their finances.

Overall, the future of money management apps looks promising as they continue to evolve and provide users with new and innovative features. As more people become aware of the importance of managing their finances effectively, money management apps will become an essential tool for achieving financial stability and security.

8. APPENDIX

Manifest.xml

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_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=".RegisterActivity"
            android:exported="false"
            android:label="@string/title_activity_register"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".ViewRecordsActivity"
            android:exported="false"
            android:label="@string/title_activity_view_records"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".SetLimitActivity"
            android:exported="false"
            android:label="@string/title_activity_set_limit"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".AddExpensesActivity"
            android:exported="false"
            android:label="@string/title_activity_add_expenses"
            android:theme="@style/Theme.ExpensesTracker" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ExpensesTracker">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

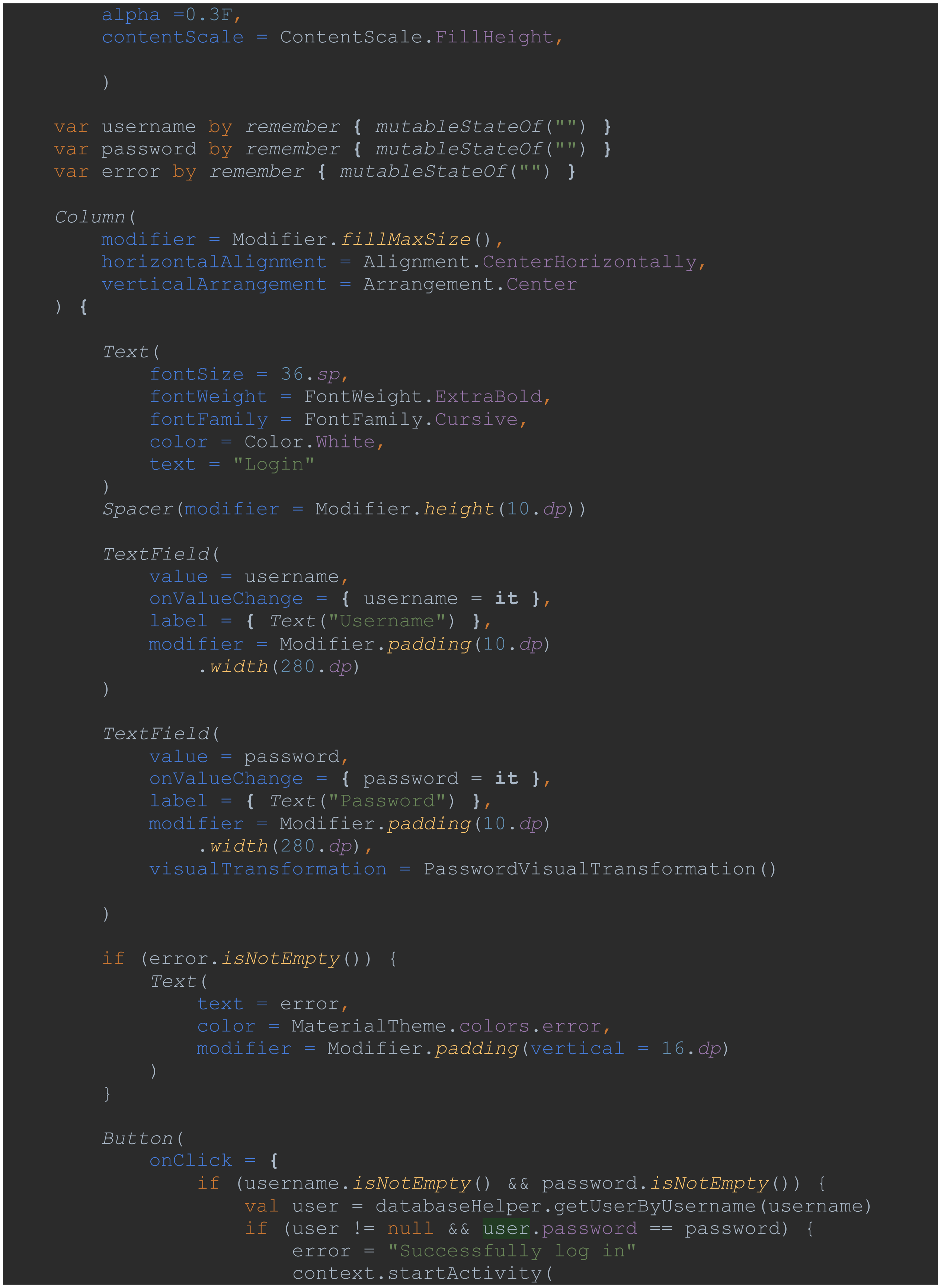
AddExpensesActivity.kt

```
package com.akhil.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")
        }

    }
}
```

LoginActivity.kt

```kotlin
package com.akhil.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.akhil.expensestracker.ui.theme.ExpensesTrackerTheme
import com.example.expensestracker.R

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 = "",
```



```kotlin
        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)
}
```

MainActivity.kt

```
package com.akhil.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
```

```kotlin
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.akhil.expensestracker.ui.theme.ExpensesTrackerTheme
import com.example.expensestracker.R

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

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

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

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

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

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

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

                        }
                    )
                }
            ) {
                MainPage()
            }
        }
    }
}

@Composable
fun MainPage() {
    Column(
        modifier = Modifier.padding(20.dp).fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "Welcome To Expense Tracker", fontSize = 42.sp,
fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center)

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

    }
}
```

RegisterActivity.kt

```
package com.akhil.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.akhil.expensestracker.ui.theme.ExpensesTrackerTheme
import com.example.expensestracker.R

class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ExpensesTrackerTheme {
```

```kotlin
                // A surface container using the 'background' color from the
theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {

                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
}

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

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

        )

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

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

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

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

        )

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

        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp),
            visualTransformation = PasswordVisualTransformation()
        )


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

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

                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))
        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            })

            {
                Spacer(modifier = Modifier.width(10.dp))
                Text(text = "Log in")
            }
        }
    }
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

SetlimitActivity.kt

```
package com.akhil.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.akhil.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}")
//                    }
//                }
//            }
//        }
//
//    }
//}
```