PROJECT REPORT

MONEYMATTER APP

Submitted by:

GOPI KUMAR E

SATHYA THARMA M

SATHISH KUMAR N

JEYARAMAN A

1.INTRODUCTION

1.1 OVERVIEW:

MoneyMatters is a financial management platform that helps individuals and businesses with financial planning, budgeting, and investment management. The platform offers a variety of tools and resources to help users understand their financial situation and make informed decisions about their finances. MoneyMatters aims to empower users with knowledge and tools to improve their financial well-being

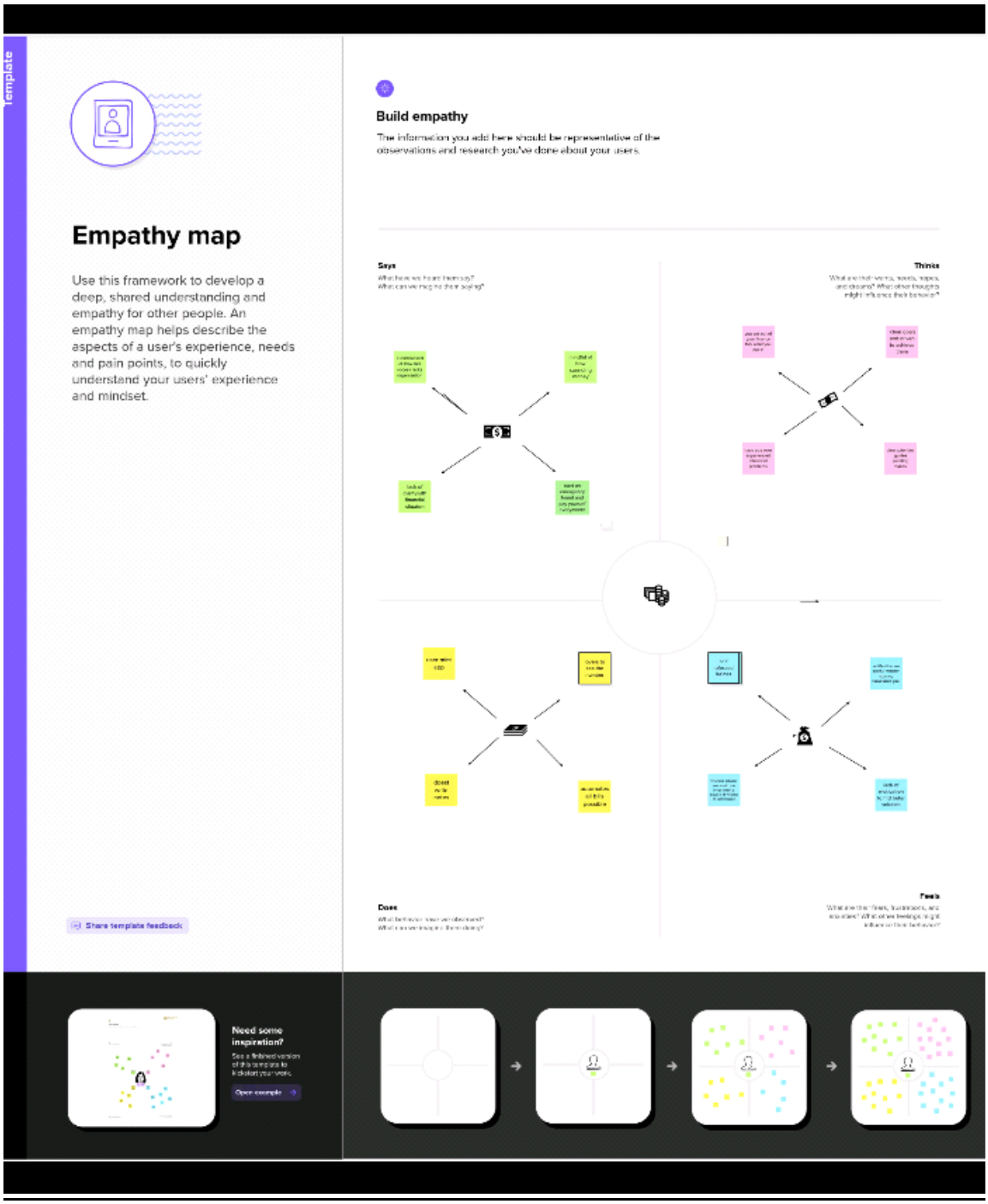
and achieve their financial goals. Whether it's saving for retirement, managing debt or investing in a portfolio, MoneyMatters provides a comprehensive financial planning solution for everyone.

1.2 PURPOSE:

MoneyMatter is likely created with the purpose of educating and empowering people about their finances- from budgeting, saving, investing, to managing debt and making informed financial decisions. Its objective may be to increase financial literacy and help individuals achieve financial stability and security.

2 PROBLEM DEFINITION&DESING THINKING

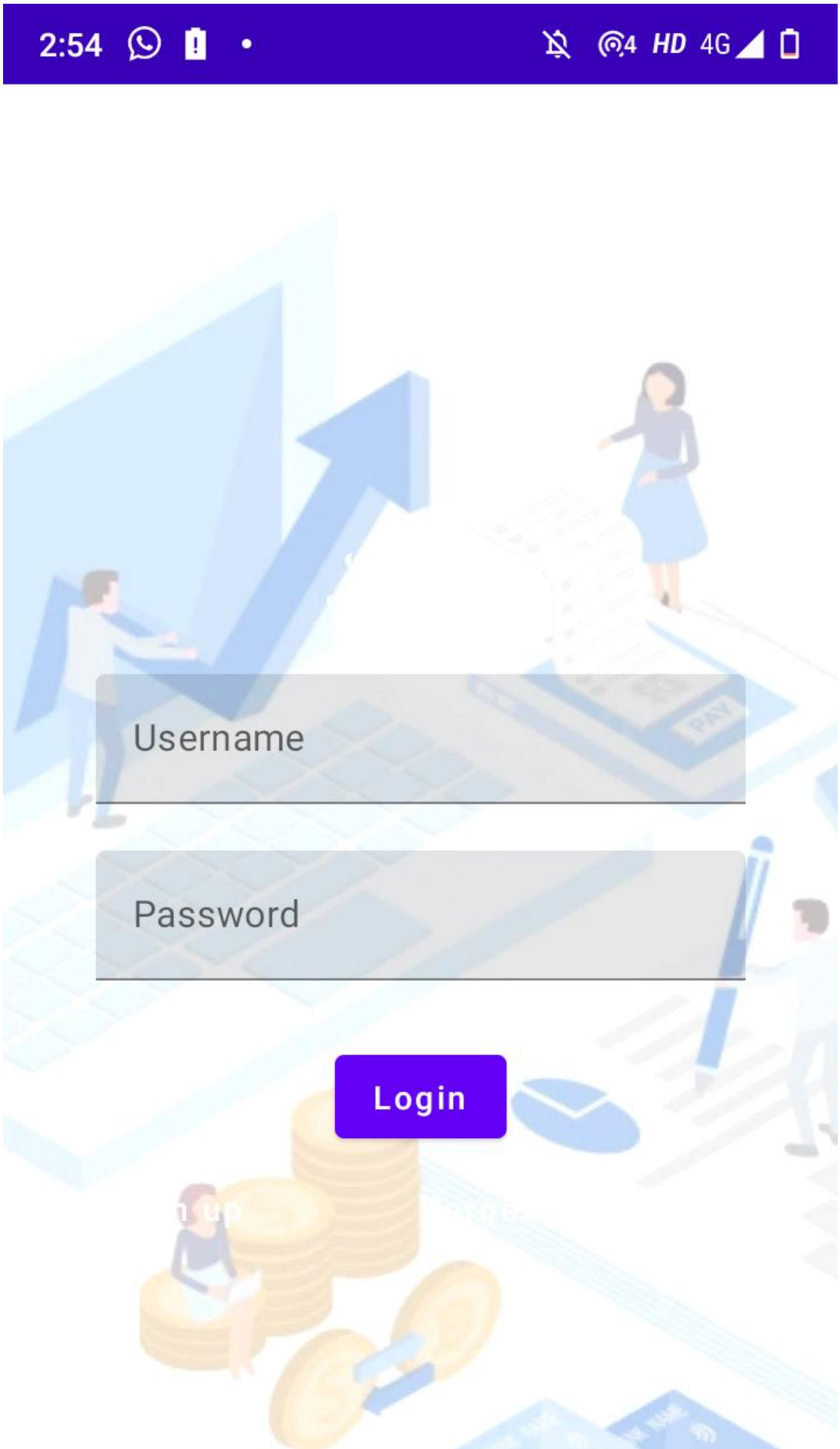
2.1 EMPATHY MAP:



2.2 IDEATION&BRAINSTORMING MAP:

3 RESULT

It can have features such as expense tracking, budget creation, bill payment reminders, and financial goal setting. Users can track their spending habits, stay on top of their bills and loans, monitor net worth, set goals and create savings plans. The app can also provide some useful tips on how to save more money and better manage financial resources.


2:54
HD 4G
Username
Password
Login

Welcome To Expense Tracker

Monthly Amount Limit

Set Amount Limit

Set Limit

Item Name

Item Name

Quantity of item

Quantity

Cost of the item

Cost

Submit

View Records

4 ADVANTAGES & DISADVANTAGES

As an AI language model, I cannot have preferences. However, here are some advantages of Money Matters that might be of interest:

1. Financial education: Money Matters provides financial education through articles, videos, and podcasts that help people make informed financial decisions.

2. Personalized advice: Money Matters offers personalized financial advice to help individuals meet their unique financial goals.

3. Investment opportunities: Money Matters offers investment opportunities through its

network of financial advisors and wealth management services.

4. Budgeting tools: Money Matters provides budgeting tools to help individuals manage their finances and create a budget that works for them.

5. Access to financial experts: Money Matters offers access to financial experts who can provide guidance on specific financial topics such as taxes, retirement planning, and insurance.

6. Security and reliability: Money Matters is a trusted source of financial information and advice, ensuring the security and reliability of its services.

7. Connection to financial resources: Money Matters connects individuals with a range of

financial resources such as credit cards, loans, and savings accounts to help them achieve their financial goals.

As an AI language model, I do not have personal opinions or feelings, but I can provide general information on the disadvantages of money matters:

Money can be a source of stress and anxiety for many people, especially when they are struggling to make ends meet or manage their finances.

The pursuit of money can become an obsession and can lead to unethical behavior, such as lying, cheating, or stealing.

Money can create social inequality, as those who have more wealth often have access to

better education, health care, and social opportunities.

The focus on making money can lead to a neglect of other important aspects of life, such as relationships, personal growth, and spirituality.

The use of money as a measure of success can lead to a lack of appreciation for non-monetary sources of happiness and fulfillment, such as love, friendship, and creativity.

5 APPLICATIONS

As an AI language model, I do not have financial requirements or needs. However, I can provide some examples of how individuals or businesses can apply MoneyMatters:

1. Personal finance management - The MoneyMatters mobile app can help individuals keep track of their expenses, create budgets, and set financial goals. This application can be used to pay bills, monitor bank transactions, and get reminders for upcoming payments.

2. Expense tracking for small businesses - MoneyMatters can be used by small business owners or freelancers for tracking expenses, documenting receipts, creating and sending invoices, and managing payments. With this application, they can easily reconcile their books and keep their finances organized.

3. Investment planning and management - MoneyMatters can be used as an investment planning tool for individuals

who are looking to invest their money. With this application, they can select investment options suitable to their risk tolerance and monitor the performance of their investments.

4. Financial reporting and analysis - MoneyMatters can be used for generating financial reports and conducting financial analysis. This application can be used to generate balance sheets, income statements, and cash flow statements, helping businesses and individuals get a firm understanding of their financial position.

5. Collaborative financial planning - MoneyMatters allows individuals and businesses to share their financial information and collaborate on financial

planning with their financial advisors or accountants. This feature makes financial planning and decision-making more efficient and effective.

6 CONCLUSION

As an AI language model, I do not have personal opinions or beliefs. However, I can provide a summary and conclusion of the topic of Moneymatters.

Moneymatters is an important issue that affects everyone. It is an essential part of our daily life and can have a significant impact on our well-being. Managing our money effectively can help us achieve financial stability, meet our needs, and lead a comfortable life.

The best way to manage money is to create and stick to a budget, save for the future, and invest wisely. It is essential to avoid overspending and living beyond our means, as this can lead to financial problems and debt.

In conclusion, Moneymatters can be challenging, but with proper planning, budgeting, and investment, it is possible to achieve financial stability and security. It is essential to educate ourselves about money management and seek professional advice if needed.

7 FUTURE SCOPE

As an AI language model, I am not able to predict the future. However, some potential future scope in MoneyMatters platform are:

1. Expanding to other countries: Currently, MoneyMatters is only available in the United States. However, there is a possibility to expand its reach and become available in other countries.

2. Integrating with more financial institutions: MoneyMatters currently works with a limited number of financial institutions. Future scope includes partnering with more financial institutions to provide users with more options.

3. Offering more financial products: MoneyMatters currently offers a limited number of financial products. Offering more financial products, such as loans, insurance, and more investment types, could be a potential future scope.

4. Incorporating blockchain technology: Blockchain technology offers enhanced security, transparency, and decentralization, which could be beneficial to the finance industry. Incorporating blockchain technology into MoneyMatters could be a potential future scope.

5. Artificial intelligence: MoneyMatters could incorporate artificial intelligence into its platform to offer personalized financial advice and investment suggestions to its users. This would require advanced machine learning algorithms and data analysis techniques.

8 APPENDIX

A. Source Code

MainActivity:

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.ExpensesTracker
Theme
```

```
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,AddExpensesActi
vity::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))

}

}

LoginAcitivity:

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.PasswordVisualTransfor
mation
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.ExpensesTracker
Theme


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)
}
```

AddExpeseAcitivity:

```
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,AddExpensesActi
vity::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")
        }


    }
}
```