PROJECT REPORT

Money Matters: A Personal Finance Management App

Android application development

Team Leader : M.SUGUMAR

Team Members : P.CHINNA DURAI

T.ESAKKI MUTHU

A.JEBISON

1.INTRODUCTION:

Money management is the process of tracking expenses, investing, budgeting, banking, and assessing tax liabilities; it is also called investment management. Money management is a strategic technique to deliver the highest interest-output value for any amount spent on making money.

1.1 OVERVIEW:

Money Matters is a personal finance management app designed to help users take control of their finances and achieve their financial goals. The app will allow users to track their expenses, set budgets, and monitor their accounts in one place.

The app will have a user-friendly interface with various features, including:

1.Expense tracking:

Users will be able to track their expenses and categorize them to better understand where their money is going.

2.Budgeting:

Users will be able to set up budgets for different categories and receive alerts when they exceed their budget limits.

3.Account aggregation:

Users can connect all their bank accounts, credit cards, and other financial accounts to get a comprehensive view of their financial situation.

4.Goal setting:

Users can set financial goals such as saving for a down payment on a house or paying off a credit card and track their progress towards achieving those goals.

5.Reports and analysis:

The app will provide users with detailed reports and analysis of their spending habits, income, and net worth.

6.Bill reminders:

Users can set up reminders for bills, subscriptions, and other payments to avoid missing payments and incurring late fees.

7.Security:

The app will have robust security features, including two-factor authentication, encryption, and regular security updates, to ensure users' data is safe and secure.

Money Matters will be available on both iOS and Android platforms, and users will be able to access their accounts across multiple devices. The app will also have a free and paid version, with the paid version offering additional features such as investment tracking and financial coaching.

1.2 PURPOSE:

The purpose of the Money Matters personal finance management app is to help users take control of their finances, manage their money more effectively, and achieve their financial goals. The app aims to provide users with a comprehensive view of their finances by allowing them to track their expenses, set budgets, monitor their accounts, and analyze their spending habits.

The app's primary goal is to help users develop good financial habits by providing them with the tools and information they need to manage their finances effectively. By tracking their expenses and setting budgets, users can identify areas where they can cut back on spending and save money. This can help them achieve their financial goals, such as saving for a down payment on a house, paying off debts, or building a retirement fund.

The app also aims to make financial management more accessible and convenient for users. By providing a range of features, including account aggregation and bill reminders, the app makes it easier for users to manage their finances and stay on top of their bills and expenses.

Additionally, the app aims to provide users with valuable insights into their finances through detailed reports and analysis. This can help users identify trends and patterns in their spending habits and make informed decisions about their financial future.

Finally, the app aims to promote financial literacy and education by offering financial coaching and investment tracking features in the paid version. This can help users develop a better understanding of personal finance and make informed decisions about their investments and financial future.

In summary, the purpose of the Money Matters personal finance management app is to help users manage their finances effectively, achieve their financial goals, and develop good financial habits.

2. Problem Definition & Design Thinking:

Problem Definition:

The problem that the Money Matters personal finance management app aims to solve is the lack of effective tools and resources for individuals to manage their finances. Many people struggle to keep track of their expenses, create budgets, and achieve their financial goals, which can lead to stress, anxiety, and financial hardship.

Design Thinking Approach:

To address this problem, the Money Matters app project can adopt a design thinking approach, which involves a human-centered and iterative process of problem-solving. This approach involves five stages:

1. Empathize:

In this stage, the team behind the Money Matters app project should focus on understanding the needs and challenges of their target users. This can involve conducting user research, surveys, and interviews to gain insights into their behaviors, attitudes, and pain points.

2.Define:

Based on the insights gathered in the Empathize stage, the team can define the problem they are trying to solve and the goals of the app. This can involve identifying the key features and functionalities that are most important to users and prioritizing them based on their needs and preferences.

3.Ideate:

In this stage, the team can brainstorm and generate a range of ideas for the app's design and features. This can involve techniques such as mind mapping, sketching, and rapid prototyping to explore different concepts and possibilities.

4.Prototype:

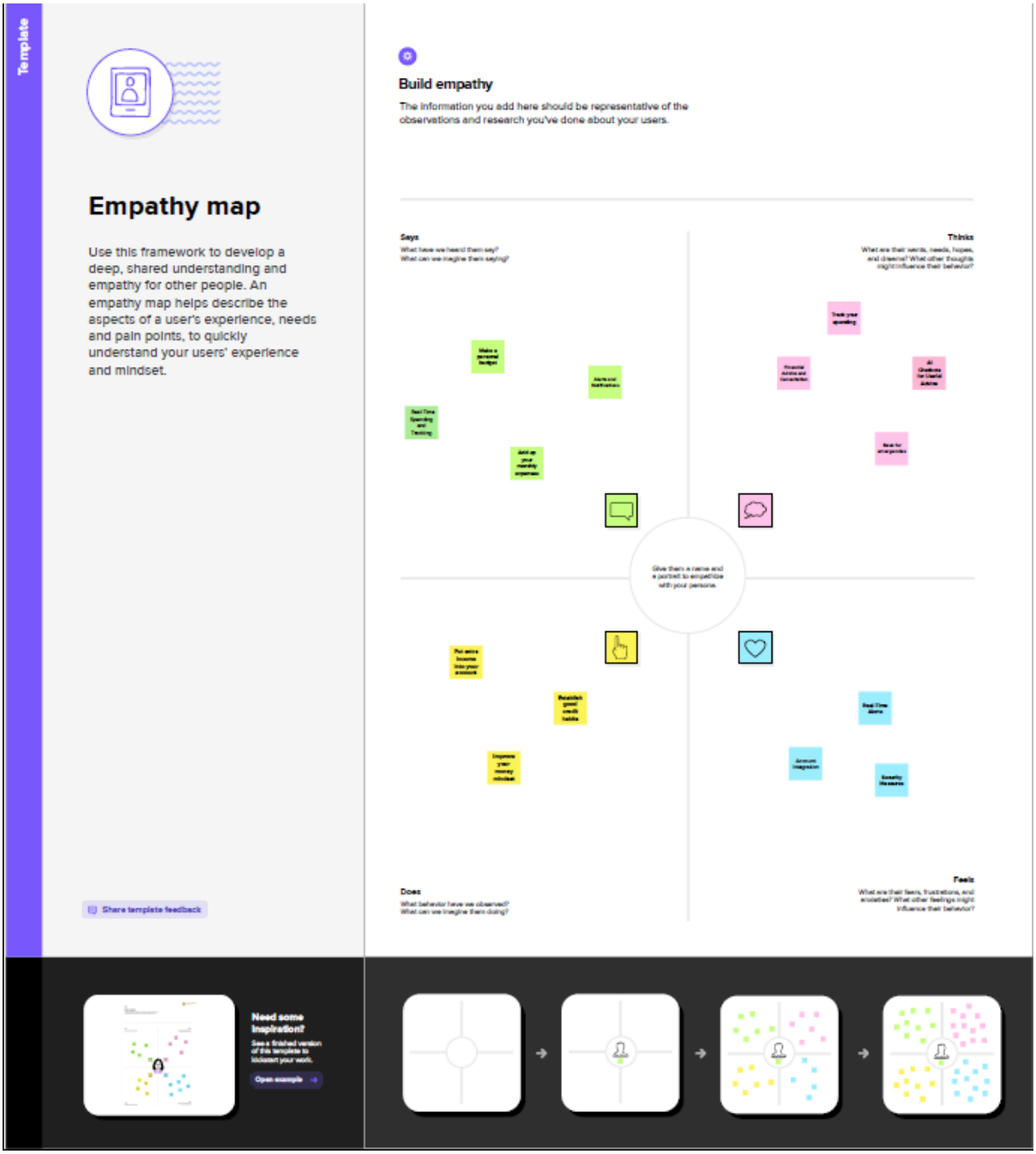
Once the ideas have been generated, the team can create prototypes of the app to test and refine its design and functionality. This can involve creating low-fidelity wireframes and high-fidelity mockups to get feedback from users and stakeholders.

5.Test:

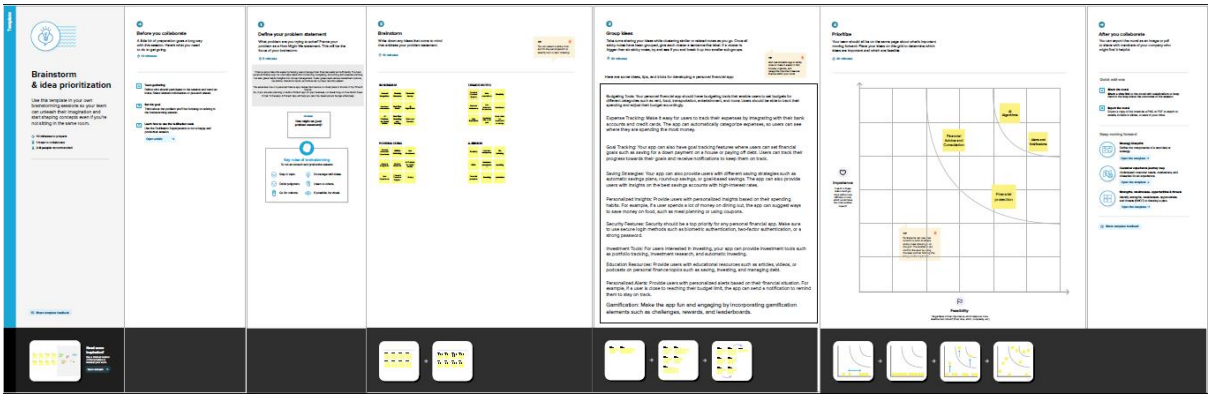
In the final stage, the team can test the app with users to gather feedback and identify areas for improvement. This can involve conducting usability tests, focus groups, and surveys to evaluate the app's effectiveness and user satisfaction.

By adopting a design thinking approach, the Money Matters app project can ensure that it meets the needs and preferences of its target users, providing a useful and effective tool for managing personal finances.

2.1 Empathy map:

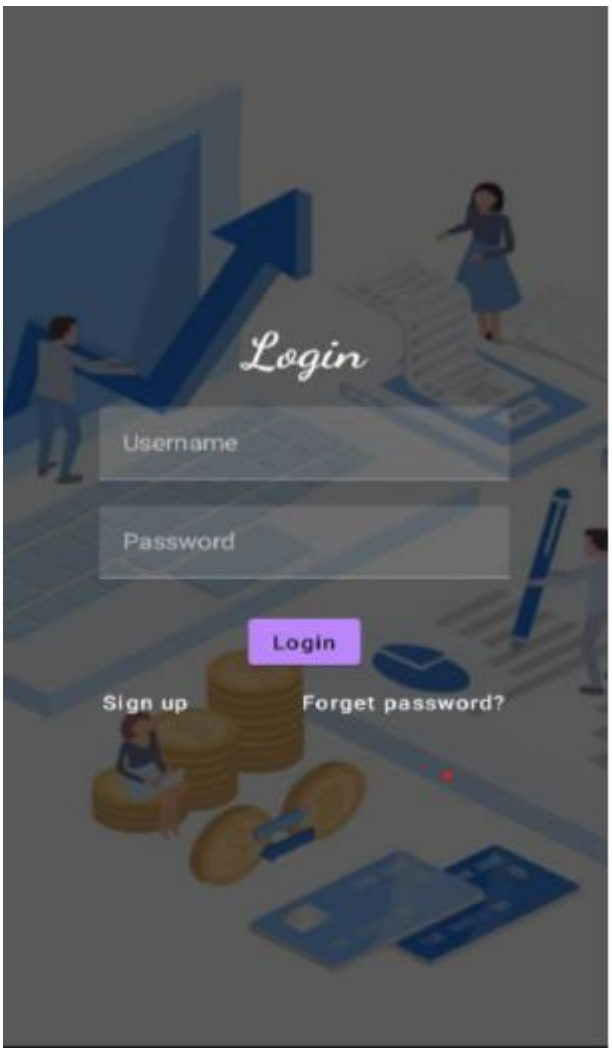


2.2 Ideation & Brainstorming Map:

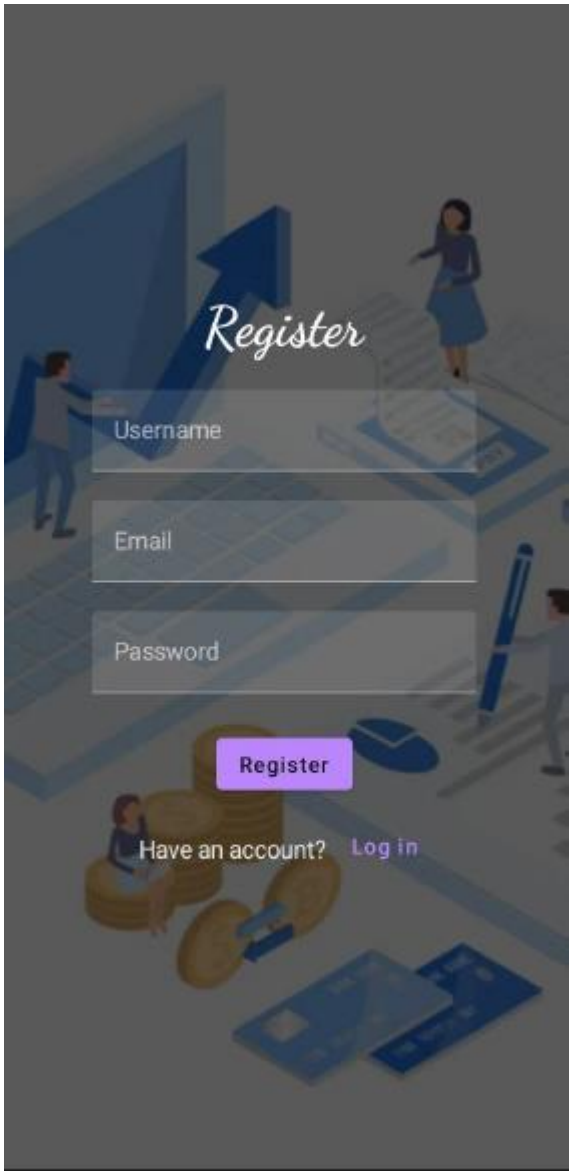


3. RESULT:

Login Page:



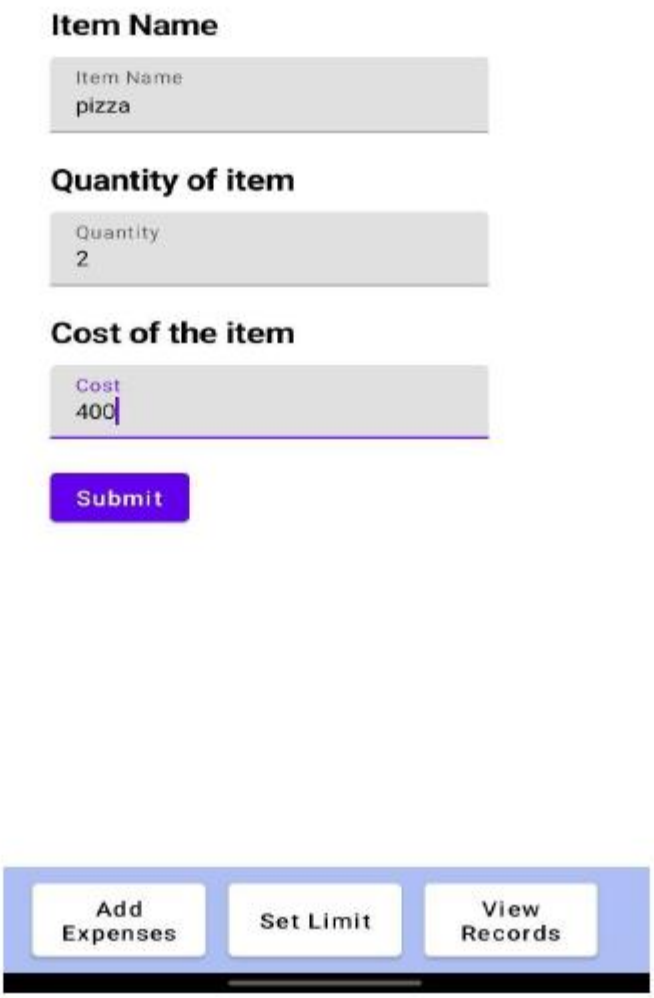
Register Page:



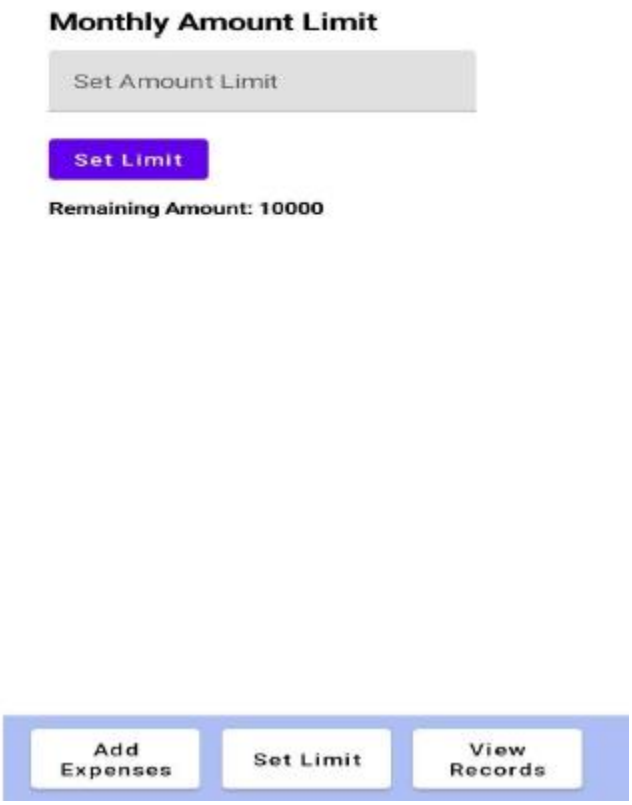
Main Page:


Welcome To
Expense Tracker
Add Expenses
Set Limit
View Records

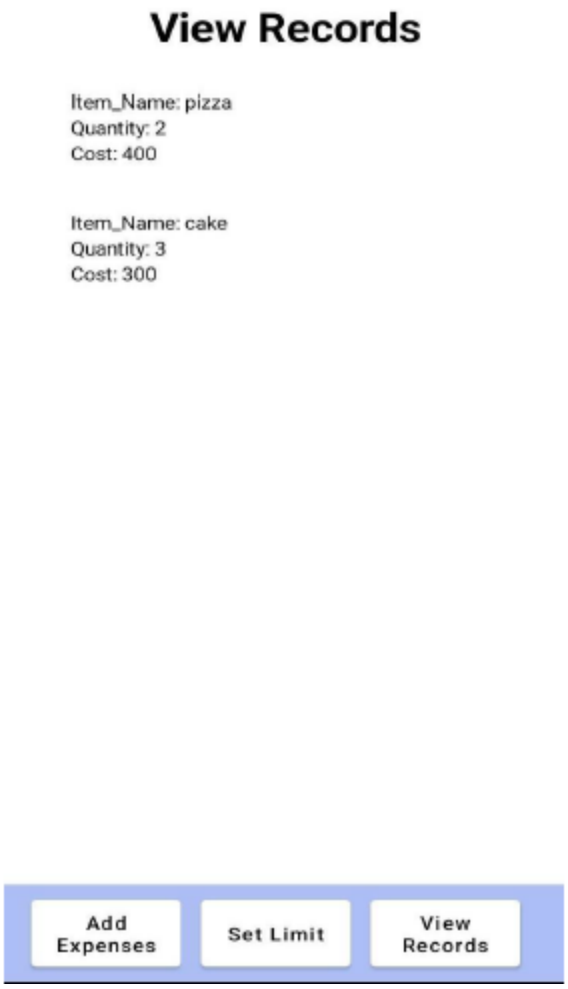
Add Expenses Page:



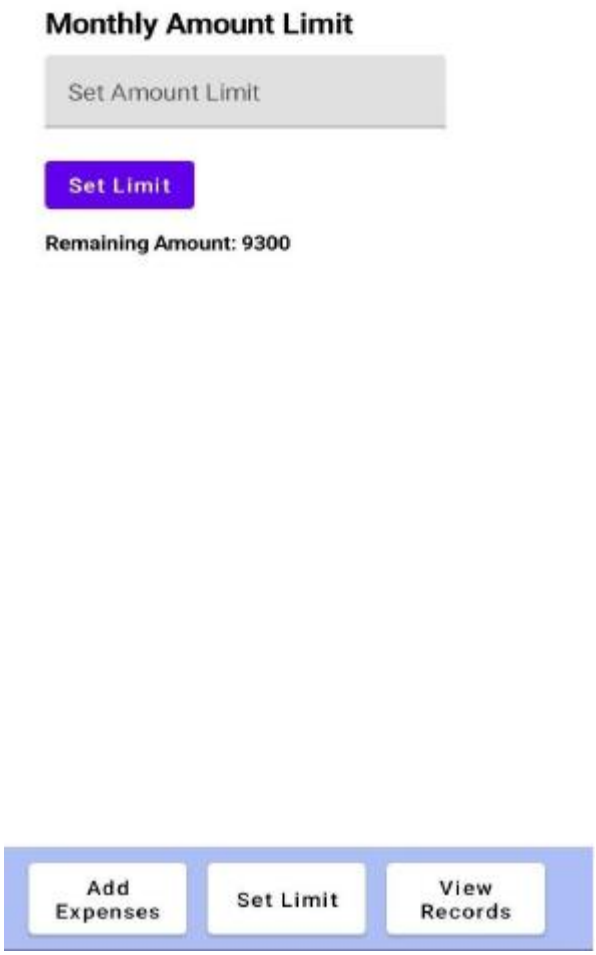
Set Limit Page before adding any data in expenses:



View Records Page:



Set Limit Page After adding expenses in add expense page:



4. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

There are several advantages to using a personal financial management app like Money Matters. Here are some of them:

1. Budgeting and Expense Tracking: Money Matters can help you create and track your budget, and keep a record of your expenses. This can help you manage your finances better and avoid overspending.
2. Goal Setting and Planning: With Money Matters, you can set financial goals and plan how to achieve them. This can help you save money, pay off debt, or invest for the future.
3. Automatic Categorization: Money Matters automatically categorizes your expenses, making it easier for you to see where your money is going. This can help you identify areas where you can cut back on spending.
4. Investment Tracking: If you have investments, Money Matters can help you track their performance and provide insights into how they are performing.
5. Secure and Convenient: Money Matters is a secure and convenient way to manage your finances. You can access it from anywhere using your smartphone or tablet, and it uses encryption to protect your data.

Overall, Money Matters can help you stay on top of your finances, save money, and achieve your financial goals.

DISADVANTAGES:

As with any personal financial management app, there are some potential disadvantages to using Money Matters. Here are a few:

1. Security concerns: Storing sensitive financial information in an app can be risky, especially if the app has vulnerabilities that could be exploited by hackers. It's important to make sure that the app you choose uses strong encryption and other security measures to protect your data.
2. Cost: While many personal finance apps are free to use, some may require a subscription or charge a fee for certain features. Before committing to a specific app, be sure to understand the full cost of using it.
3. Limited features: Some personal finance apps may not offer all of the features you need to manage your finances effectively. For example, if you have complex investment portfolios or multiple income streams, you may need a more robust app to keep track of everything.
4. Learning curve: Depending on the complexity of the app, there may be a learning curve involved in using it effectively. You may need to spend some time getting familiar with the app's interface and features before you can use it to its full potential.
5. Dependence on technology: If you rely too heavily on a personal finance app, you may be at a disadvantage if the app crashes or goes offline. It's important to have a backup plan in case you can't access your financial information through the app.

Overall, Money Matters and other personal finance apps can be helpful tools for managing your finances, but it's important to weigh the pros and cons before deciding whether or not to use one.

5.APPLICATIONS:

Money Matters is a personal financial management app designed to help users manage their finances more effectively. Here are some of its key applications:

1. Budgeting: Money Matters allows users to create and track their budget, so they can monitor their spending and stay on track with their financial goals. Users can set up categories for their expenses and income, and the app will automatically categorize transactions for easy tracking.
2. Bill reminders: The app can send reminders to users when bills are due, so they can avoid late fees and missed payments.
3. Investment tracking: Money Matters allows users to track their investments, including stocks, bonds, and mutual funds. Users can view their portfolio performance and see how their investments are performing over time.
4. Goal setting: Users can set financial goals, such as saving for a down payment on a house or paying off debt, and track their progress towards these goals using the app.
5. Transaction tracking: Money Matters allows users to track their transactions and view their account balances in real-time. Users can also set up alerts for unusual account activity or transactions that exceed a certain amount.
6. Reporting: The app provides detailed reports and charts that allow users to analyze their spending habits and identify areas where they can cut back or save more money.

Overall, Money Matters is a comprehensive personal finance app that can help users manage their money more effectively and achieve their financial goals.

6.CONCLUSION:

In conclusion, Money Matters is a personal finance management app that can help users manage their finances in a more efficient and effective way. It offers various features, including budgeting, bill reminders, investment tracking, goal setting, transaction tracking, and reporting, that can help users achieve their financial goals. While there are potential disadvantages to using a personal finance app, such as security concerns and cost, Money Matters can be a valuable tool for those looking to improve their financial management skills. Overall, Money Matters is a comprehensive app that can help users make better financial decisions and achieve greater financial stability. However, it's important to carefully consider the pros and cons before deciding whether or not to use Money Matters or any other personal finance app.

In conclusion, Money Matters is a personal finance management app designed to help users manage their finances effectively. It offers a range of features including budgeting, bill reminders, investment tracking, goal setting, transaction tracking, and reporting. While there are potential disadvantages to using a personal finance app, such as security concerns and the learning curve involved in using it effectively, Money Matters can be a helpful tool for those looking to improve their financial management skills. It's important to weigh the pros and cons before deciding whether or not to use Money Matters or any other personal finance app, but overall, Money Matters can be a valuable tool for achieving financial stability and reaching financial goals.

7.FUTURE SCOPE:

Money Matters has a bright future in the personal finance management app industry due to the increasing demand for financial management tools. Here are some potential future scopes for Money Matters:

1. Integration with other financial management tools: Money Matters could potentially integrate with other financial management tools such as tax software or accounting software, which would provide users with a more comprehensive financial management experience.
2. AI-powered financial advice: With advancements in artificial intelligence, Money Matters could potentially offer personalized financial advice to users based on their financial data and goals.
3. More robust investment tracking: Money Matters could enhance its investment tracking features to provide users with more detailed information about their portfolio performance and investment opportunities.
4. Integration with banks and financial institutions: Money Matters could potentially integrate with banks and financial institutions to provide users with real-time transaction data, which would allow for more accurate budgeting and financial planning.
5. Cryptocurrency support: As cryptocurrency becomes more popular, Money Matters could potentially add support for tracking and managing cryptocurrency investments.

Overall, Money Matters has a lot of potential for future growth and development as the demand for personal finance management tools continues to increase.

8.APPENDIX:

```
// 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(
    s@PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,


    )
// UserDao.kt
package com.example.expensestracker
```

```
import androidx.room.*


@Dao
interface UserDao {


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


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


    @Update
    suspend fun updateUser(user: User)


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

// UserDatabase.kt

```
package com.example.expensestracker


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

```
import androidx.room.Room
import androidx.room.RoomDatabase

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

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

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

```
            }
        }
    }
}
// UserDatabaseHelper.kt
package com.example.expensestracker


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


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


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


        private const val TABLE_NAME = "user_table"
```

```
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)
    }

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

```
    }

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

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

```
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME
WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

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


    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME",
null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
```

```
                    id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }


}
```

// Items.kt

```
package com.example.expensestracker


import androidx.room.ColumnInfo
```

```
import androidx.room.Entity

import androidx.room.PrimaryKey


@Entity(tableName = "items_table")

data class Items(

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

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

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

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

)
```

// ItemsDao.kt

```
package com.example.expensestracker


import androidx.room.*


@Dao

interface ItemsDao {


    @Query("SELECT * FROM items_table WHERE cost= :cost")

    suspend fun getItemsByCost(cost: String): Items?
```

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

    @Update
    suspend fun updateItems(items: Items)

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

// ItemsDatabase.kt

```
package com.example.expensestracker

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

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

    abstract fun ItemsDao(): ItemsDao
```

```
    companion object {


        @Volatile
        private var instance: ItemsDatabase? = null


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

// ItemsDatabaseHelper.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
    }
}
```

// 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?,
)
```

// ExpenseDao.kt

```
package com.example.expensestracker

import androidx.room.*

@Dao
```

```
interface ExpenseDao {

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

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

    @Update
    suspend fun updateExpense(items: Expense)

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

// ExpenseDatabase.kt

```
package com.example.expensestracker

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

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

    abstract fun ExpenseDao(): ItemsDao

    companion object {

        @Volatile
        private var instance: ExpenseDatabase? = null

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

```
}
```

// ExpenseDatabaseHelper.kt

```
package com.example.expensestracker

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

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

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

        private const val TABLE_NAME = "expense_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_AMOUNT = "amount"
```

```
    }

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

        db?.execSQL(createTable)
    }

    override fun onUpgrade(db1: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        db1?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db1)
    }

    fun insertExpense(expense: Expense) {
        val db1 = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_AMOUNT, expense.amount)
        db1.insert(TABLE_NAME, null, values)
        db1.close()
```

```
    }


    fun updateExpense(expense: Expense) {

        val db = writableDatabase

        val values = ContentValues()

        values.put(COLUMN_AMOUNT, expense.amount)

        db.update(TABLE_NAME, values, "$COLUMN_ID=?",
arrayOf(expense.id.toString()))

        db.close()

    }





    @SuppressLint("Range")

    fun getExpenseByAmount(amount: String): Expense? {

        val db1 = readableDatabase

        val cursor: Cursor = db1.rawQuery("SELECT * FROM
${ExpenseDatabaseHelper.TABLE_NAME} WHERE
${ExpenseDatabaseHelper.COLUMN_AMOUNT} = ?", arrayOf(amount))

        var expense: Expense? = 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
    }

}
```

// 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)

}
```

// 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)
}
```

// MainActivity.kt

```
package com.example.expensestracker

import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
```

```
import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.res.painterResource

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

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

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

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

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


class MainActivity : ComponentActivity() {

    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContent {

            Scaffold(

                // in scaffold we are specifying top bar.

                bottomBar = {

                    // inside top bar we are specifying

                    //  background  color.

                    BottomAppBar(backgroundColor = Color(0xFFadbef4),
```

```
                        modifier = Modifier.height(80.dp),
                        // along with that we are specifying
                        // title for our top bar.
                        content = {

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

                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.ja
va))},
                                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))


    }
}
```

// AddExpensesActivity.kt

```
package com.example.expensestracker


import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
```

```
import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import  androidx.compose.ui.graphics.Color

import androidx.compose.ui.platform.LocalContext

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

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

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp


class AddExpensesActivity : ComponentActivity() {

    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper

    private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper

    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        itemsDatabaseHelper = ItemsDatabaseHelper(this)

        expenseDatabaseHelper = ExpenseDatabaseHelper(this)

        setContent {

            Scaffold(

                // in scaffold we are specifying top bar.

                bottomBar = {
```

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

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

                            Button(
                                onClick =
{startActivity(Intent(applicationContext,AddExpensesActivity::class.ja
va))},
                                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")
        }


    }
}
```

// SetLimitActivity.kt

```
package com.example.expensestracker


import android.annotation.SuppressLint
```

```
import android.content.Context

import android.content.Intent

import android.os.Bundle

import android.util.Log

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.*

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.LazyRow

import androidx.compose.foundation.lazy.items

import androidx.compose.material.*

import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

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

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

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

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


class SetLimitActivity : ComponentActivity() {
```

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

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

                        Button(
                            onClick = {
                                startActivity(
```

```
                                        Intent(
                                            applicationContext,

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

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

                            Button(
```

```
                                onClick = {
                                    startActivity(
                                        Intent(
                                            applicationContext,

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

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

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

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

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


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

```
    ) {

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

        Text(text = "Monthly Amount Limit", fontWeight =
FontWeight.Bold, fontSize = 20.sp)
        Spacer(modifier = Modifier.height(10.dp))
        TextField(value = amount, onValueChange = { amount = it },
            label = { Text(text = "Set Amount Limit ") })

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

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

        Button(onClick = {
            if (amount.isNotEmpty()) {
```

```
        val expense = Expense(
            id = null,
            amount = amount
        )
        expenseDatabaseHelper.insertExpense(expense)
    }
}) {
    Text(text = "Set Limit")
}

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

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

    horizontalArrangement = Arrangement.Start
) {
    item {

        LazyColumn {
```

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


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


        }
    }
}
```

// 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}")
                    }
                }
            }
        }


    }
}
```