INTRODUCTION

1.1 Overview

Just as there is an app for almost everything today, a personal finance app is a money management application. It helps you regulate and keep track of your financial flow, set a budget, and get meaningful insights about your savings and expenditure.

1.2 Purpose

The role of a personal finance app is to make your life easier. How? By facilitating, you manage your finances efficiently.

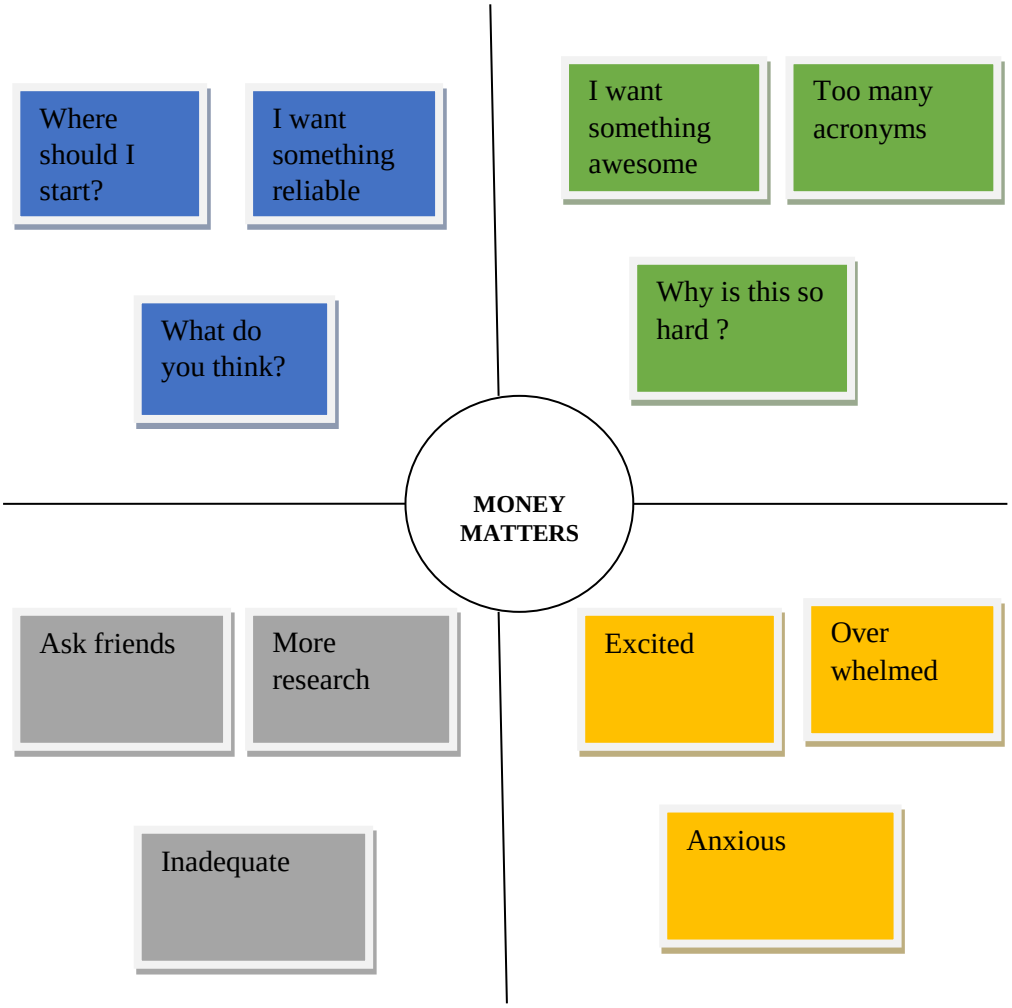
Personal finance apps simplify financial management and offer a great way to take a front seat and see where you are spending your money. Besides, you get valuable insights about better money management, access to various investment options, advice, and insurance inputs, and also prepare for financial emergencies.

Here is a list of the standard functions a personal finance app performs.

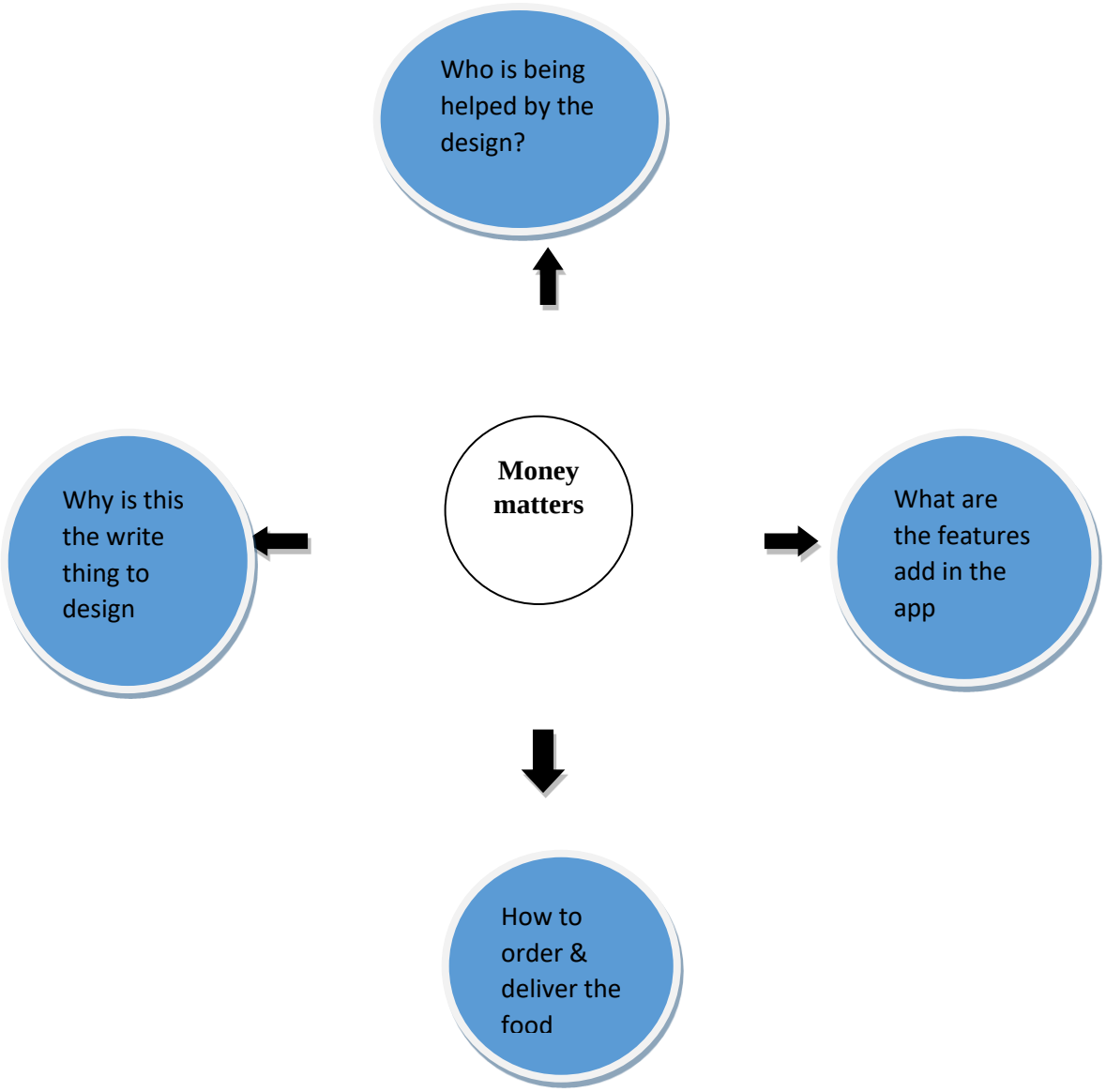
- Tracks bills and expenses.
- Calculates income and expenditure.
- Plans out the budget.
- Organizes finance.
- Analyzes the data and generates insightful reports

2. Problem Definition & design thinking

2.1 Empathy Map



2.2 Ideation & brainstorming map



3. RESULT:

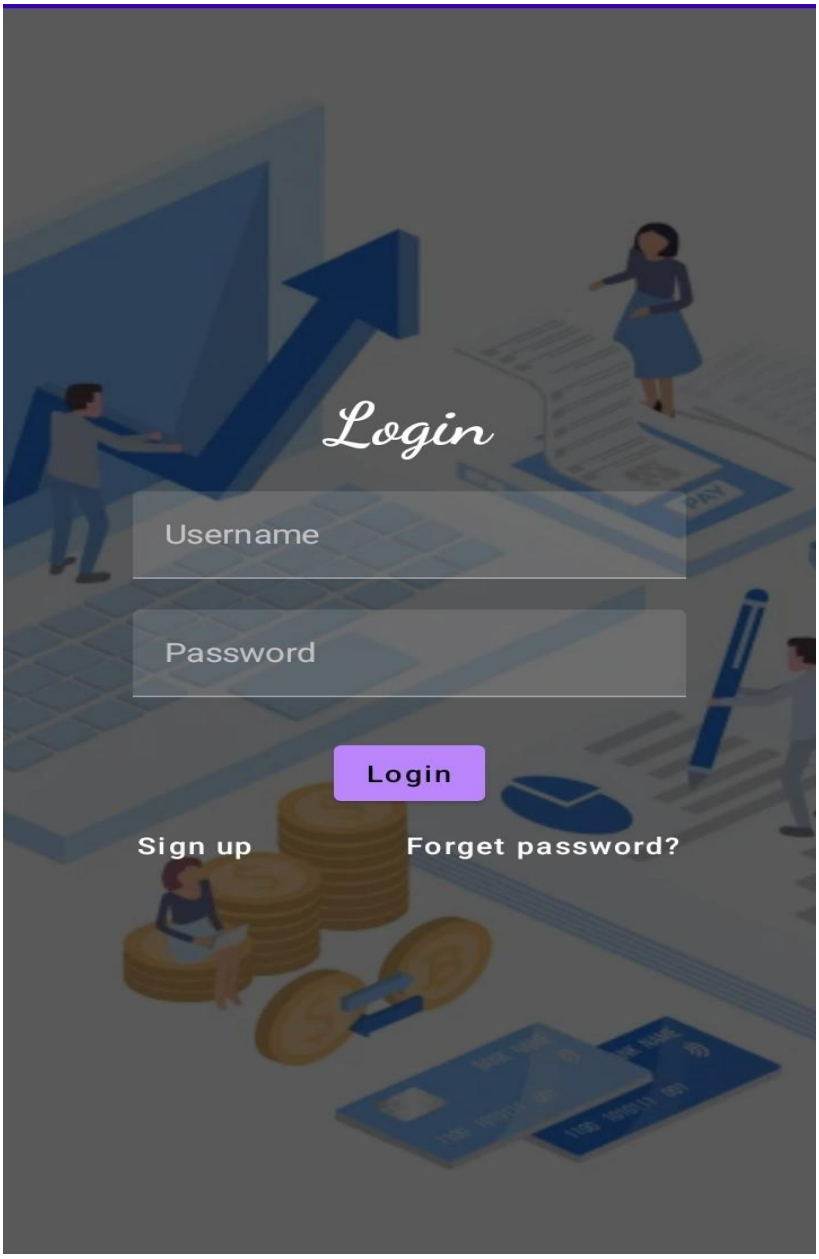


Figure 3.1

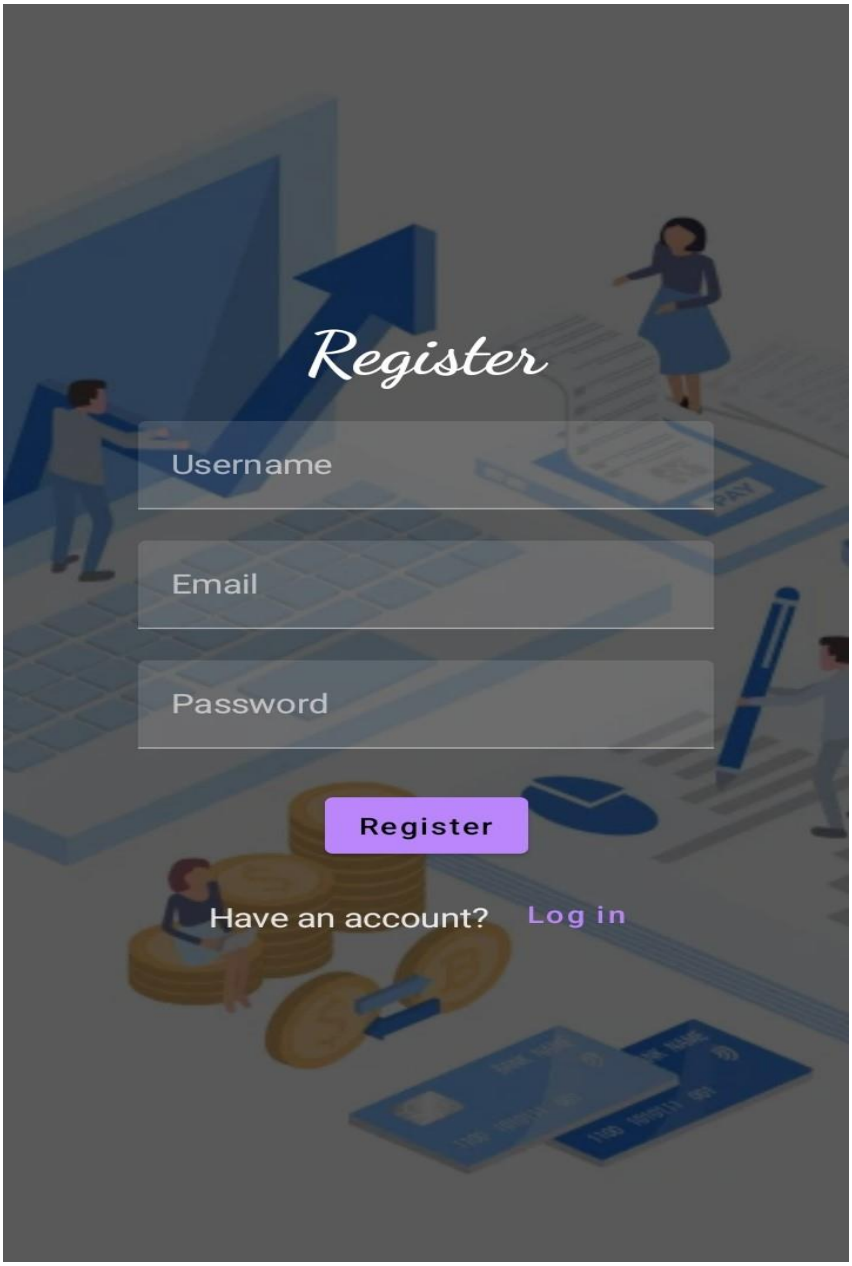

Register
Username
Email
Password
Register
Have an account?
Log in

Figure 3.2

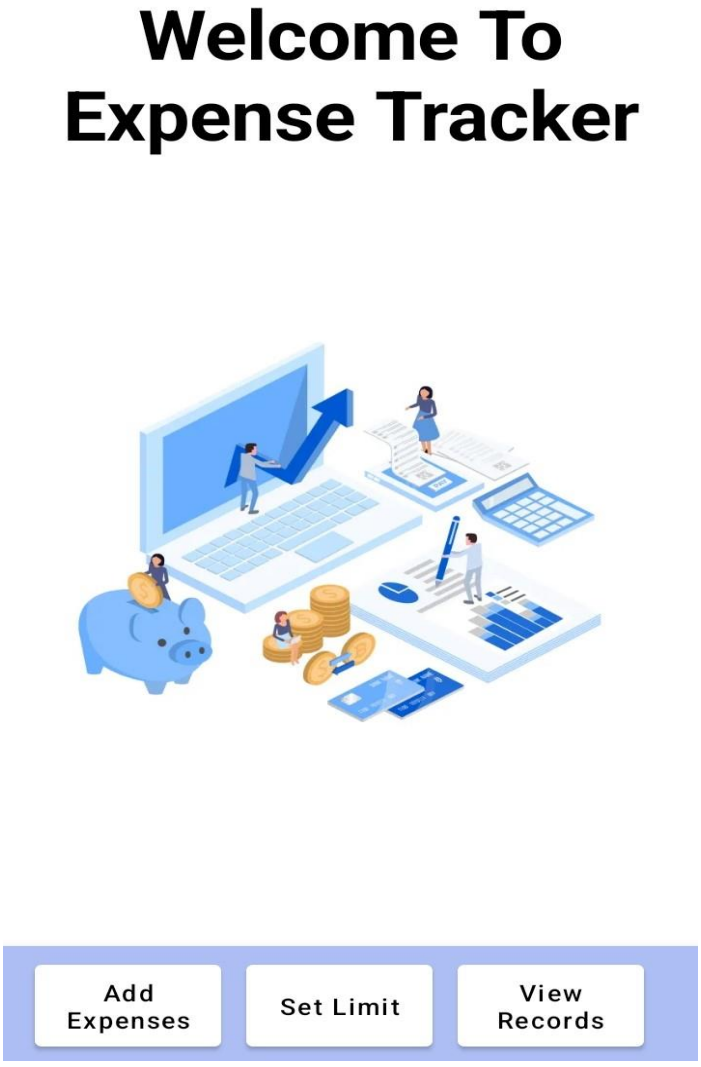

Welcome To
Expense Tracker
Add Expenses
Set Limit
View Records

Figure 3.3

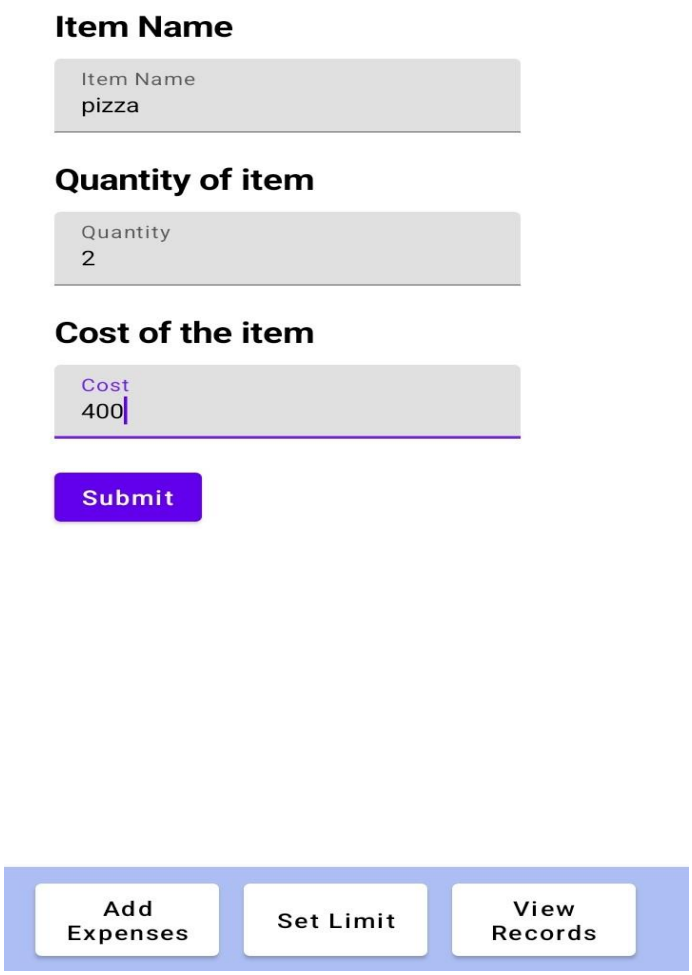

Item Name
Item Name
pizza
Quantity of item
Quantity
2
Cost of the item
Cost
400
Submit
Add Expenses
Set Limit
View Records

Figure 3.4

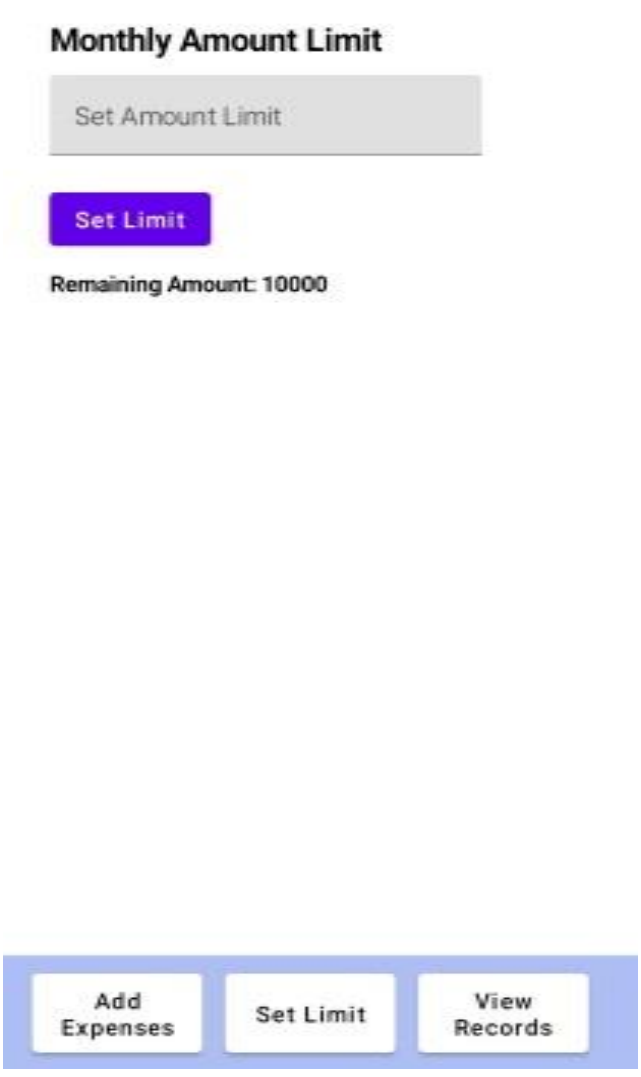

Monthly Amount Limit
Set Amount Limit
Set Limit
Remaining Amount: 10000
Add Expenses
Set Limit
View Records

Figure 3.5

View Records

Item_Name: pizza
Quantity: 2
Cost: 400

Item_Name: cake
Quantity: 3
Cost: 300

Add Expenses | Set Limit | View Records

Monthly Amount Limit

Set Amount Limit

Set Limit

Remaining Amount: 9300

Add Expenses | Set Limit | View Records

4. ADVANTAGES & DISADVANTAGES

Financial management refers to directing, organizing and control of commercial operations such as procurement and disbursal of revenue from the individual or business. It also includes the application of specific principles to financial assets of the firm, and even playing a role in revenue management. The scope of financial management includes financial decisions relating to raising money from different sources; the cost of financing projects; and the potential returns attained during the process. It also provides investment decisions in the fixed and current assets, which consist the working capital decisions.

ADVANTAGES

Sound financial management leads to increased visibility within the operations, and it supports understanding of the numbers at each level in the business or institute. The advantages of financial management make sure there is investor confidence. Investors are usually keen to look for signs of security within business operations. Effective financial management allows for the correct balance between risk and profit maximization.

Financial management also endorses better decision making. When the relevant facts are easily accessible because of digitization and organization, it becomes easier to derive solutions based on the circumstances of the situation.

As an incredible benefit, financial management assists with taxation. Taxes have often been frowned upon as one of the limitations of financial system. There are tax loopholes and exceptions for enterprises and institutes which can be taken advantage of if the terms are satisfied. For example, a business can claim tax deductions based on their quantity of office space.

Limitations of Financial Management

One of the critical limitations of financial management is the rigidity it ensures within enterprises. The standards of operation are fixed by incorporating particular accounting parameters; however, when the tasks are done, the conditions may change from the original situation. The rules are not able to keep up with the dynamic changes in the market environment, and that leads to bureaucracy and lost revenue.

Similarly, implementing standards of practice within a business or an institute comes with a cost. It requires both hardware and software installation and orientation for the entire staff so they can adjust to the new system seamlessly.

Limitations of Accounting

Financial statements illustrate the company’s financial position at the time. Unfortunately, it usually does not give trend analysis unless the reader is well versed in financial ratios. The users of financial statements who are the primary stakeholders are more interested in the future of the institution for the long term and short term which are not indicated.

Accounting also uses historical costs to assess the values, and this does not consider such things like price changes or inflation.

Finally, accounting and financial statements do not measure things which do not have a monetary value. Factors relevant to a business such as customer loyalty cannot be expressed in financial figures regardless of their importance. Other qualities like reputation and management ability also have no place within financial statements.

DISADVANTAGES:

- When we are dealing with money, security becomes a primary concern
- The task to build a complex/linked personal finance app is relatively complicated and expensive

5. APPLICATIONS

Financial planning is the starting point for effective personal finance management. It involves three important elements: a realistic budget, well-defined financial goals and real-time tracking.

A budget helps you stay committed to your financial goal, while real-time tracking helps you take the right actions at the right time. However, money management is easier said than done. Budgeting and real-time tracking can be challenging, requiring considerable time and effort investment. But money management apps can help. These apps are extremely useful in helping you effectively manage your money and personal finances.

6. CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

7. FUTURE SCOPE

Scope of financial management includes the following:

- Economic concepts (such as macro and micro economics, economic order quantity, money value discounting factor and more) are directly applied with the financial management approaches.
- Accounting plays a critical role in management decision making and in financial management.
- Financial management applies large number of mathematical and statistical tools and concepts (also known as econometric).
- Production management is the operational aspect of decision making requiring the support of financial management.
- Financial management/finance department allocates resources for marketing and related activities that play a crucial role in a firm’s marketing budget.
- Financial management is related to human resource department, which provides manpower to all the functional areas of management.

8. APPENDIX

Create User Data Class

```
package com.example.expensestracker

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,

    )
```

Create An UserDao Interface

```
package com.example.expensestracker

import androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")
```

```
    suspend fun getUserByEmail(email: String): User?

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

    @Update
    suspend fun updateUser(user: User)

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

Create An UserDatabase Class

```
package com.example.expensestracker

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

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

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null
```

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

Create An UserDatabaseHelper Class

```
package com.example.expensestracker
```

```
import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper


class UserDatabaseHelper(context: Context) :

    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {


    companion object {

        private const val DATABASE_VERSION = 1

        private const val DATABASE_NAME = "UserDatabase.db"


        private const val TABLE_NAME = "user_table"

        private const val COLUMN_ID = "id"

        private const val COLUMN_FIRST_NAME = "first_name"

        private const val COLUMN_LAST_NAME = "last_name"

        private const val COLUMN_EMAIL = "email"

        private const val COLUMN_PASSWORD = "password"
```

```
}

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

    db?.execSQL(createTable)
}

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

fun insertUser(user: User) {
    val db = writableDatabase
```

```
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

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


}
```

Create Items Data Class

```
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?,
)
```

Create ItemsDao Interface

```
package com.example.expensestracker

import androidx.room.*

@Dao
interface ItemsDao {

    @Query("SELECT * FROM items_table WHERE  cost= :cost")
    suspend fun getItemsByCost(cost: String): Items?

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

    @Update
    suspend fun updateItems(items: Items)

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

Create ItemsDatabse Class

```
package com.example.expensestracker

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

```
import androidx.room.RoomDatabase

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

    abstract fun ItemsDao(): ItemsDao

    companion object {

        @Volatile
        private var instance: ItemsDatabase? = null

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

Create ItemsDatabaseHelper Class

```
package com.example.expensestracker

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



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

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

        private const val TABLE_NAME = "items_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_ITEM_NAME = "item_name"
        private const val COLUMN_QUANTITY = "quantity"
        private const val COLUMN_COST = "cost"
    }
```

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

    db?.execSQL(createTable)
}

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

fun insertItems(items: Items) {
    val db = writableDatabase
    val values = ContentValues()
    values.put(COLUMN_ITEM_NAME, items.itemName)
    values.put(COLUMN_QUANTITY, items.quantity)
    values.put(COLUMN_COST, items.cost)
    db.insert(TABLE_NAME, null, values)
    db.close()
}



@SuppressLint("Range")
fun getItemsByCost(cost: String): Items? {
    val db = readableDatabase
```

```
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_COST = ?", arrayOf(cost))
        var items: Items? = null
        if (cursor.moveToFirst()) {
            items = Items(
                id = cursor.getInt(cursor.getColumnIndex(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
    }


}
```

Create Expense Data Class

```
package com.example.expensestracker

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "expense_table")
data class Expense(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "amount") val amount: String?,
)
```

Create ExpenseDao Interface

```
package com.example.expensestracker

import androidx.room.*

@Dao
interface ExpenseDao {

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

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

    @Update
    suspend fun updateExpense(items: Expense)

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

Create ExpenseDatabase Class

```
package com.example.expensestracker

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

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

    abstract fun ExpenseDao(): ItemsDao

    companion object {

        @Volatile
        private var instance: ExpenseDatabase? = null

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

```
                "expense_database"
            ).build()
            instance = newInstance
            newInstance
        }
    }
}
```

Create ExpenseDatabaseHelper Class

```
package com.example.expensestracker

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

class 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
    }
```

}

Creating LoginActivity.Kt With Database

```
package com.example.expensestracker


import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.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)
}
```

Creating RegisterActivity.Kt With Database

```
package com.example.expensestracker

import android.content.Context
import android.content.Intent
```

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.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)
}



package com.example.expensestracker

import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme

class MainActivity : ComponentActivity() {
    @SuppressLint("UnusedMaterialScaffoldPaddingParameter")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

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

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

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

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

                    Button(
                        onClick = {
                            startActivity(
```

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

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

Button(
    onClick = {
        startActivity(
            Intent(
                applicationContext,
                ViewRecordsActivity::class.java
            )
        )
    },
    colors = ButtonDefaults.buttonColors(backgroundColor = Color.White),
    modifier = Modifier.size(height = 55.dp, width = 110.dp)
)
{
    Text(
```

```
                    text = "View Records", color = Color.Black, fontSize = 14.sp,
                    textAlign = TextAlign.Center
                )
            }

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

Creating MainActivity.Kt File

```
@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))

    }
}
```

Creating AddExpensesActivity.Kt File

```
package com.example.expensestracker

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class AddExpensesActivity : ComponentActivity() {
    private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper
    private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper
    @SuppressLint("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")
```

```
        }


    }
}
```

Creating SetLimitActivity.Kt File

```
package com.example.expensestracker

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.expensestracker.ui.theme.ExpensesTrackerTheme

class SetLimitActivity : ComponentActivity() {
```

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

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


                        Button(
                            onClick = {
                                startActivity(
                                    Intent(
                                        applicationContext,
                                        AddExpensesActivity::class.java
                                    )
                                )
                            },
                            colors = ButtonDefaults.buttonColors(backgroundColor = Color.White),
                            modifier = Modifier.size(height = 55.dp, width = 110.dp)
                        )
                        {
```

```
    Text(
        text = "Add Expenses", color = Color.Black, fontSize = 14.sp,
        textAlign = TextAlign.Center
    )
}

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

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

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

Button(
    onClick = {
        startActivity(
```

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

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

@Composable
fun Limit(context: Context, expenseDatabaseHelper: ExpenseDatabaseHelper, expense: List<Expense>) {
    Column(
```

```
    modifier = Modifier
        .padding(top = 100.dp, start = 30.dp)
        .fillMaxHeight()
        .fillMaxWidth(),
    horizontalAlignment = Alignment.Start
) {

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

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

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

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

    Button(onClick = {
        if (amount.isNotEmpty()) {
            val expense = Expense(
                id = null,
                amount = amount
            )
            expenseDatabaseHelper.insertExpense(expense)
```

```
            }
        }) {
            Text(text = "Set Limit")
        }

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

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

            horizontalArrangement = Arrangement.Start
        ) {
            item {

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

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

        }
    }
}
```

```
//@Composable
//fun Records(expense: List<Expense>) {
//    Text(text = "View Records", modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom = 24.dp ), fontSize = 30.sp)
//    Spacer(modifier = Modifier.height(30.dp))
//    LazyRow(
//        modifier = Modifier
//            .fillMaxSize()
//            .padding(top = 80.dp),
//
//        horizontalArrangement = Arrangement.SpaceBetween
//    ){
//        item {
//
//            LazyColumn {
//                items(expense) { expense ->
//                    Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) {
//                        Text("Remaining Amount: ${expense.amount}")
//                    }
//                }
//            }
//        }
//
//    }
//}
```

Creating ViewRecordsActivity.Kt File

```
package com.example.expensestracker


import android.annotation.SuppressLint
import android.content.Intent
```

```
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.ScrollState
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.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}")
                }
            }
        }
    }
}
```

-
  - Running The Application

Modifying AndroidManifest.Xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/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.TravelApp"
        tools:targetApi="31">
        <activity
```

```
    android:name=".RegisterActivity"
    android:exported="false"
    android:label="RegisterActivity"
    android:theme="@style/Theme.TravelApp" />
<activity
    android:name=".SingaporeActivity"
    android:exported="false"
    android:label="@string/title_activity_singapore"
    android:theme="@style/Theme.TravelApp" />
<activity
    android:name=".ParisActivity"
    android:exported="false"
    android:label="@string/title_activity_paris"
    android:theme="@style/Theme.TravelApp" />
<activity
    android:name=".BaliActivity"
    android:exported="false"
    android:label="@string/title_activity_bali"
    android:theme="@style/Theme.TravelApp" />
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="@string/app_name"
    android:theme="@style/Theme.TravelApp"/>
<activity
    android:name=".LoginActivity"
    android:exported="true"
    android:label="@string/app_name"
    android:theme="@style/Theme.TravelApp">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
```

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

</manifest>
```