ANDROID STUDIO – EXPERIENCE BASED PROJECT LEARNING

Money Matters: A Personal Finance Management App

Submitted by

ABINAYA S	-	711022104001
NEHHA DHARSHINI G	-	711022104035
ROSMITHA M	-	711022104042
SWARGINAH MELVIN S	-	711022104055

BACHELOR OF COMPUTER SCIENCE AND ENGINEERING

IN

FIFTH SEMESTER

COMPUTER SCIENCE AND ENGINEERING

INFO INSTITUTE OF ENGINEERING, COIMBATORE – 641107

NOVEMBER/DECEMBER - 2024

BONAFIDE CERTIFICATE

Certified that this project “Money Matter: A Personal Finance Management App” is the Bonafide work of ABINAYA S(711022104001), NEHHA DHARSHINI G(711022104035), ROSMITHA M(711022104042), SWARGINAH MELVIN S(711022104055) who carried out the project work under any supervision.

SIGNATURE	**SIGNATURE**
STAFF COORDINATOR	**HEAD OF THE DEPARTMENT**
Mrs. A. SARANYA M.E.,	**Dr. G. SELVAVINAYAGAM Ph.D.,**
ASSISTANT PROFESSOR	**HEAD OF THE DEPARTMENT**
DEPT. COMPUTER SCIENCE AND ENGINEERING	DEPT. COMPUTER SCIENCE AND ENGINEERING
INFO INSTITUTE OF ENGINEERING,	INFO INSTITUTE OF ENGINEERING,
KOVILPALAYAM COIMBATORE - 641107	KOVILPALAYAM COIMBATORE - 641107

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely thank to Tamil Nadu Skill Development Corporation (TNSDC), Naan Mudhalvan" Platform and ANDROID STUDIO – EXPERIENCE BASED PROJECT LEARNING (EBPL) for encouragement towards our project work for providing necessary skill training.

We sincerely thank our Principal Dr. N. KOTTISWARAN, M.E., Ph.D., and Head of the Department Dr. G. SELVAVINAYAGAM, M.E., Ph.D., and also Staff Coordinator Mrs. A. SARANYA M.E for her encouragement towards our project works.

We also thank our project guide and our parents for the complete and whole hearted support, motivation guidance and help in making our project activities.

ABSTRACT

This Android application is developed to help users efficiently track their expenses, providing an intuitive platform for managing both personal and financial data. The app allows users to register, log in, and maintain a comprehensive record of their spending habits. It utilizes Room Database and SQLite to store and manage crucial data, such as user information, purchased items, and expense records, ensuring quick access and smooth performance.

The user management system enables individuals to create accounts and log in securely with their credentials. Users can also update their personal details as needed. This functionality is supported by a secure authentication system that verifies the login credentials by comparing the inputted values with the records stored in the database.

The expense-tracking feature enables users to log items they purchase, along with relevant details like item names, quantities, and costs. This helps users keep track of their spending and analyze where their money is going. By utilizing Room Database, the app ensures that the data is stored efficiently, allowing for easy retrieval and management.

The app also employs SQLite for a more flexible database management approach. Users can access all the data related to their expenses and items, providing a centralized and organized system for personal finance management.

Overall, this app aims to offer a seamless and effective solution for users to monitor and manage their finances. With its simple user interface and secure backend system, it helps individuals stay organized, track their purchases, and make better financial decisions.

INTRODUCTION

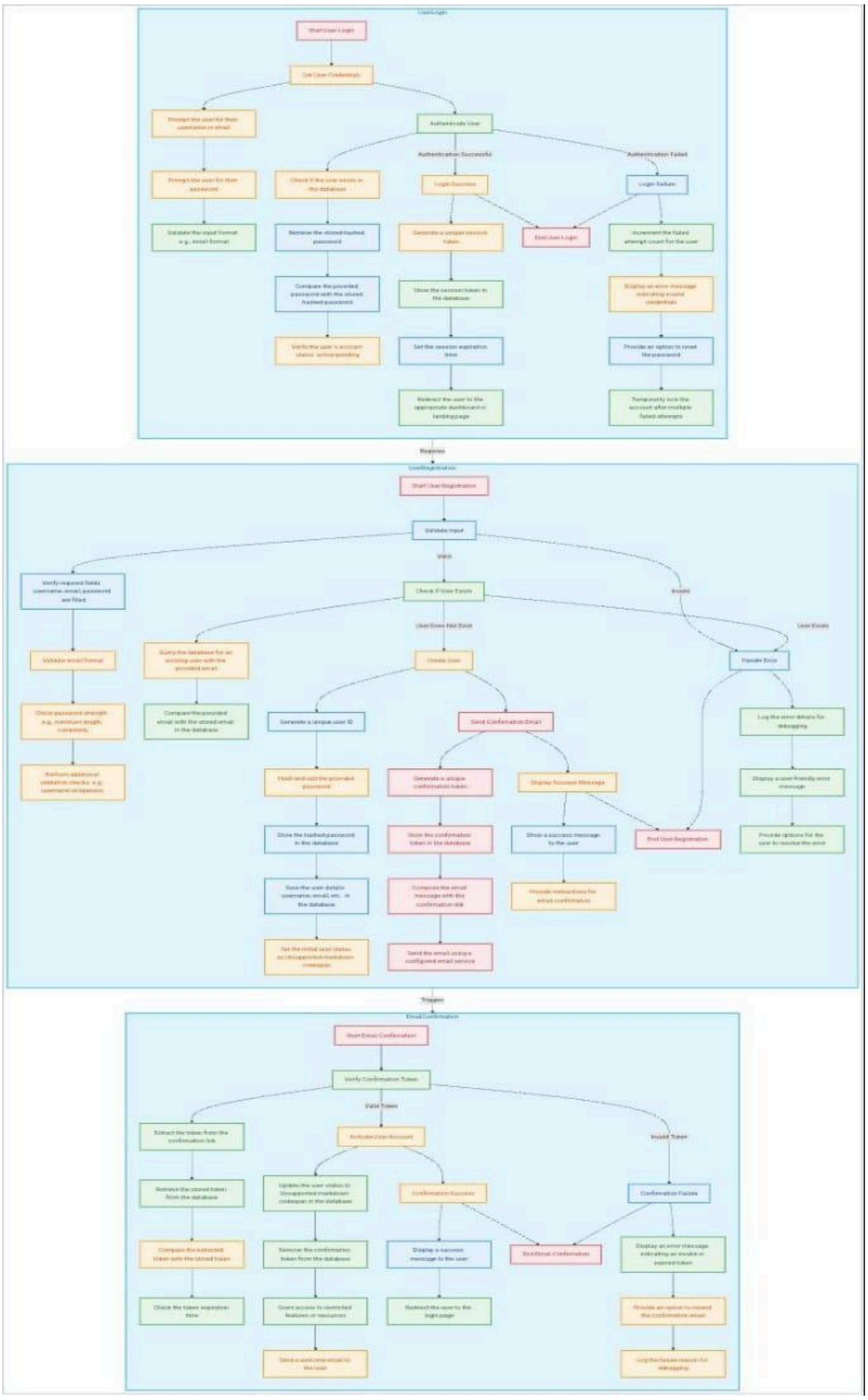
The Expenses Tracker app is designed to offer users a seamless experience in managing their personal finances. By combining a simple interface with powerful features, it makes the process of tracking spending both easy and efficient. The core functionality of the app revolves around tracking expenses by allowing users to input details such as the name of the item, the quantity purchased, and the total cost. This data is stored securely in the app's database, ensuring privacy and ease of access.

In addition to recording daily expenses, the app allows users to view their financial history through a clear and organized interface. With the ability to categorize expenses, users can gain insights into where their money is going and identify areas where they might be able to save. The Room Database ensures that all records are stored securely and efficiently, while the app's integration with SQLite allows for smooth and fast data retrieval, even when the user is offline.

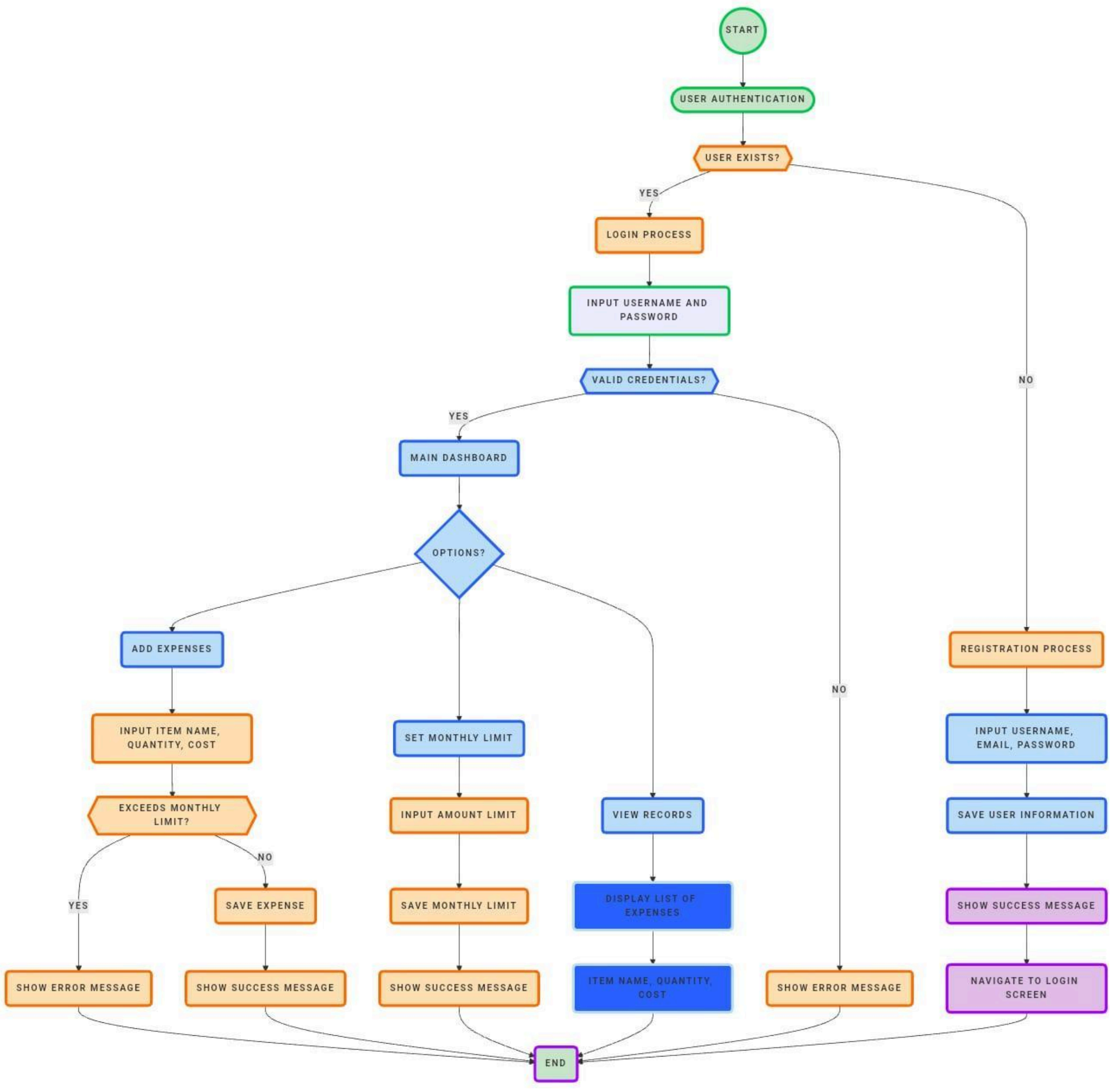
Security is a priority, and the app provides secure user authentication with options for storing and updating personal details. By using Android Jetpack Compose, the app is able to deliver an aesthetically pleasing and easy-to-navigate layout, giving users an enjoyable experience while managing their financial transactions.

Moreover, the app offers features like real-time updates and reminders to help users stay on track with their financial goals. Whether you are managing monthly expenses, planning for a specific purchase, or simply looking to get a better understanding of your spending patterns, the Expenses Tracker app is an essential tool for anyone seeking financial discipline and awareness. By keeping everything organized and easily accessible, it empowers users to make informed decisions, stick to their budget, and ultimately achieve greater financial stability.

DATA FLOW DIAGRAM


User Login
Start User Login
Get User Credentials
Prompt the user for their username or email
Authenticate User
Authentication Successful
Authentication Failed
Prompt the user for their password
Check if the user exists in the database
Login Success
Login Failure
Validate the input format e.g., email format
Retrieve the stored hashed password
Generate a unique session token
End User Login
Increment the failed attempt count for the user
Compare the provided password with the stored hashed password
Store the session token in the database
Display an error message indicating invalid credentials
Verify the user's account status: active/pending
Set the session expiration time
Provide an option to reset the password
Redirect the user to the appropriate dashboard or landing page
Temporarily lock the account after multiple failed attempts
Requires
User Registration
Start User Registration
Validate Input
Valid
Invalid
Verify required fields (username, email, password) are filled
Check if User Exists
User Does Not Exist
User Exists
Validate email format
Query the database for an existing user with the provided email
Create User
Handle Error
Check password strength e.g., minimum length, complexity
Compare the provided email with the stored email in the database
Generate a unique user ID
Send Confirmation Email
Log the error details for debugging
Perform additional validation checks e.g. username uniqueness
Hash and salt the provided password
Generate a unique confirmation token
Display Success Message
Display a user-friendly error message
Store the hashed password in the database
Store the confirmation token in the database
Show a success message to the user
End User Registration
Provide options for the user to resolve the error
Save the user details (username, email, etc.) in the database
Compose the email message with the confirmation link
Provide instructions for email confirmation
Set the initial user status
Send the email using a configured email service
Triggers
Email Confirmation
Start Email Confirmation
Verify Confirmation Token
Valid Token
Invalid Token
Extract the token from the confirmation link
Activate User Account
Retrieve the stored token from the database
Update the user status in the database
Confirmation Success
Confirmation Failure
Compare the extracted token with the stored token
Remove the confirmation token from the database
Display a success message to the user
End Email Confirmation
Display an error message indicating an invalid or expired token
Check the token expiration time
Grant access to restricted features or resources
Redirect the user to the login page
Provide an option to resend the confirmation email
Send a welcome email to the user
Log the failure reason for debugging

USER CASE DIAGRAM



SOFTWARE REQUIREMENT

Operating Systems:
- Android

Programming Languages:
- Java
- Swift
- Kotlin
- JavaScript

Integrated Development Environments (IDEs):
- Android Studio
- Xcode
- Visual Studio

Frameworks:
- Flutter
- Xamarin

Databases:
- MySQL
- MongoDB
- Firebase

Testing Tools:
- TestNG
- Appium

Version Control Systems:
- Git

Emulators:
- Android Emulator
- iOS Simulator

PROGRAM CODE

```
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?,

   )
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)
}
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
            }
        }
    }
}
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>()
Join        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
    }
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?,
)
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)
}
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
            }
        }
    }
}
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
    }
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?,
)
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)
}
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
            }
        }
    }
}
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
    }



}
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)
    }
    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()
            }
        }
    }
}

@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))

    }
}
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")
    }

  }
}
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")
```

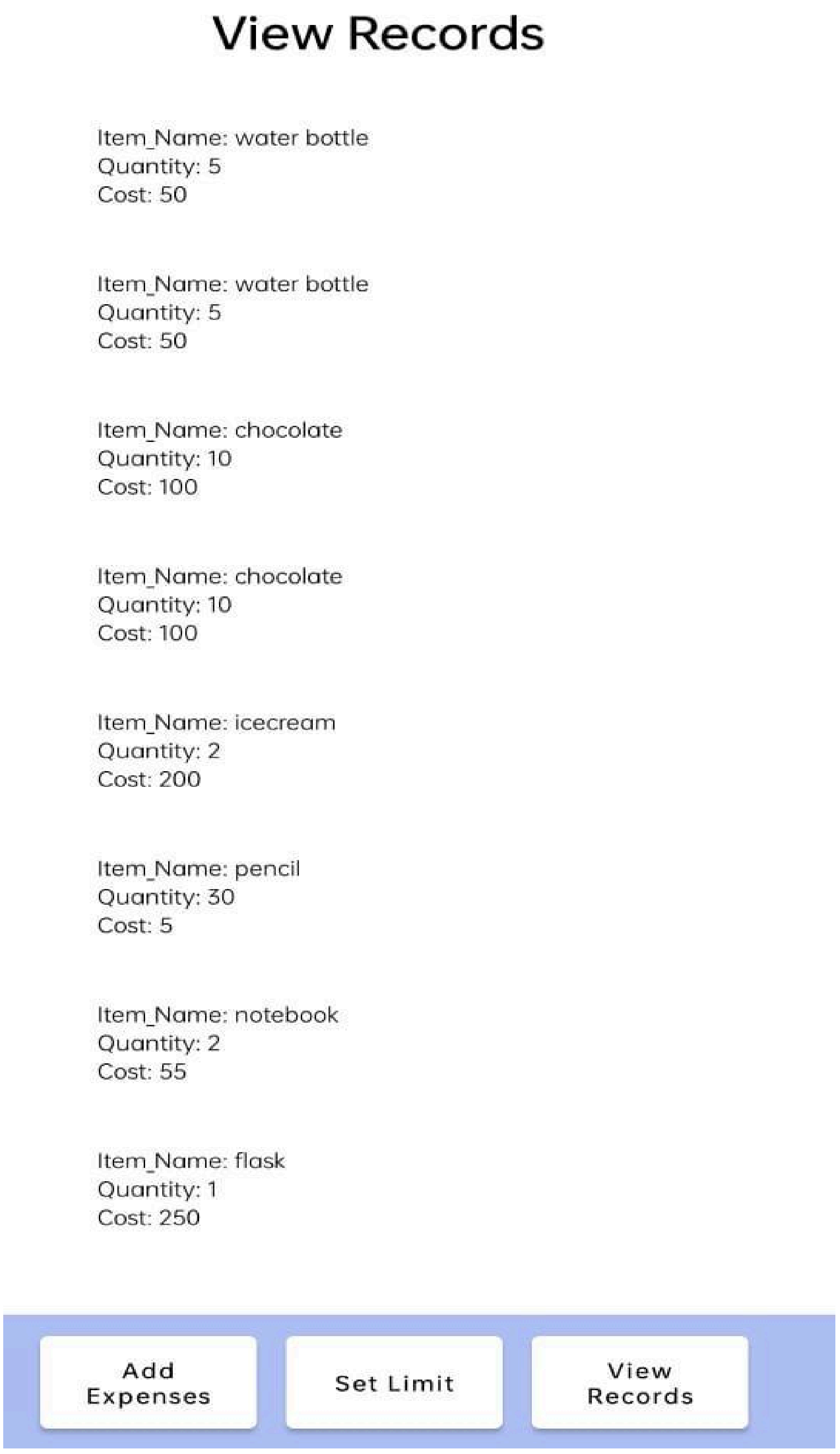
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    expenseDatabaseHelper = ExpenseDatabaseHelper(this)
    setContent {
        Scaffold(
            // in scaffold we are specifying top bar.
            bottomBar = {
                // inside top bar 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}")
//                    }
//                }
//            }
//        }
//
```

```
//    }
//}
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))
```

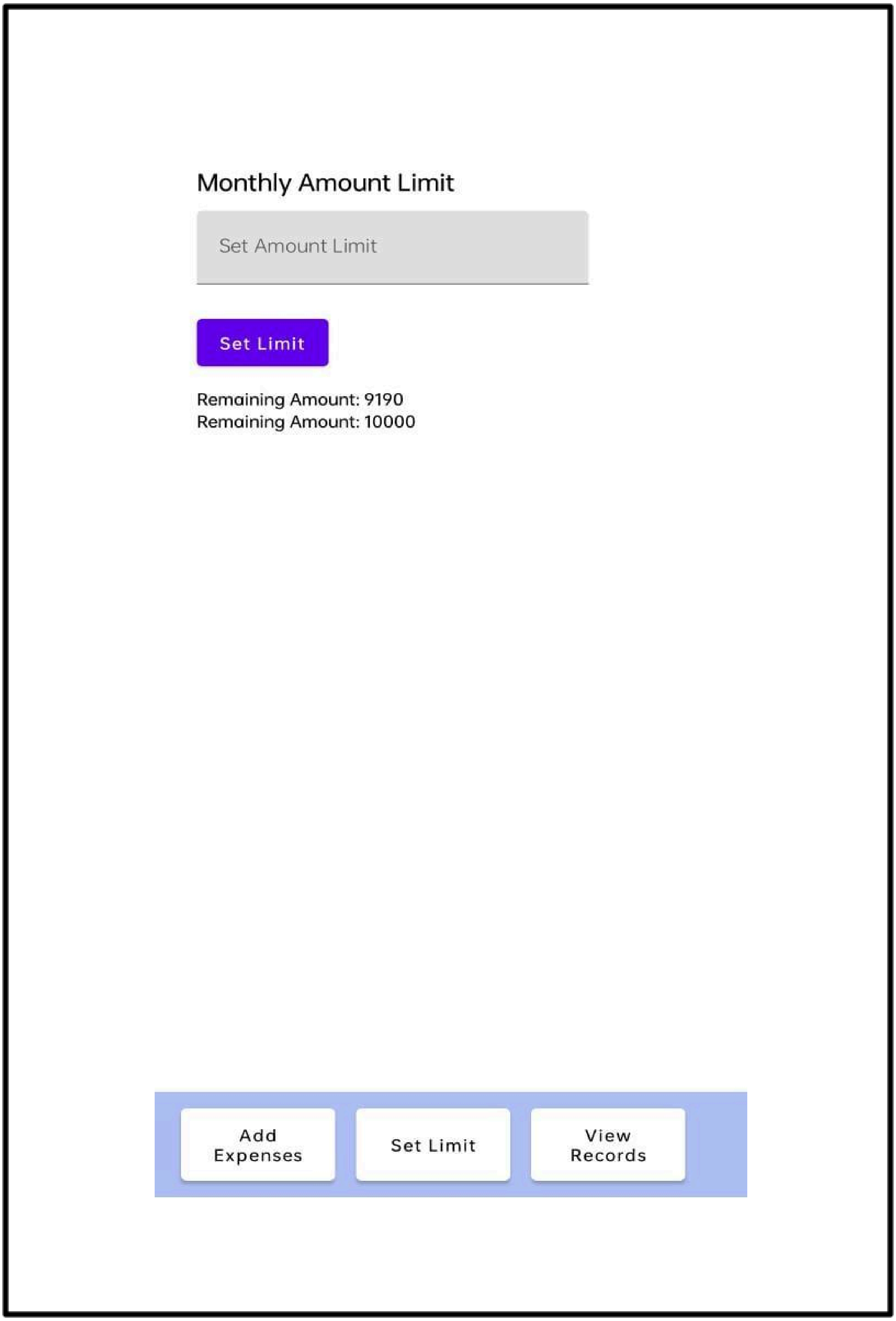
```
Button(
    onClick = {
        startActivity(
            Intent(
                applicationContext,
                AddExpensesActivity::class.java
            )
        )
    },
    colors = ButtonDefaults.buttonColors(backgroundColor = Color.White),
    modifier = Modifier.size(height = 55.dp,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

  }
}
<?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>
```

OUTPUT



View Records

```
Item_Name: water bottle
Quantity: 5
Cost: 50

Item_Name: water bottle
Quantity: 5
Cost: 50

Item_Name: chocolate
Quantity: 10
Cost: 100

Item_Name: chocolate
Quantity: 10
Cost: 100

Item_Name: icecream
Quantity: 2
Cost: 200

Item_Name: pencil
Quantity: 30
Cost: 5

Item_Name: notebook
Quantity: 2
Cost: 55

Item_Name: flask
Quantity: 1
Cost: 250
```



Remaining Amount: 9190
Remaining Amount: 10000

CONCLUSION

The Expenses Tracker Application is designed to provide users with a seamless experience in managing their expenses. Using Room Database and SQLite, the app efficiently handles CRUD operations for key entities: User, Items, and Expense.The User table manages user credentials with fields such as firstName, lastName, email, and password. The Items table stores details like itemName, quantity, and cost, while the Expense table tracks monetary expenses. The use of Data Access Objects (DAOs) such as UserDao, ItemsDao, and ExpenseDao optimizes database operations, ensuring smooth querying and modification.In terms of UI, the app features a secure login system built with Jetpack Compose, enhancing the user experience with modern design and seamless authentication. The UserDatabaseHelper, ItemsDatabaseHelper, and ExpenseDatabaseHelper provide the foundational database support, using SQLite for those who prefer traditional relational database management.With its efficient data management, scalability, and user-friendly interface, the Expenses Tracker Application offers a comprehensive solution for individuals or small businesses to track and analyze daily expenses. Whether you're adding new users, managing items or tracking costs, the application is designed to handle all aspects of financial data efficiently

FUTURE ENHANCEMENT

1. Budget Planning and Alerts: Set budgets and get notifications when limits are reached.

2. Data Visualization: Use graphs and charts for expense tracking and trends.

3. Multi-Currency Support: Track expenses in different currencies with real-time conversion.

4. Recurring Expense Tracking: Automatically track recurring expenses like subscriptions.

5. Receipt Scanning: Use OCR to scan and log expenses from receipts.

6. Expense Sharing: Split and share expenses with others.

7. Cloud Sync: Sync data across devices for backup and access.

8. AI Insights: Get personalized financial advice based on spending patterns.

9. Bank Integration: Import transactions directly from bank accounts or payment services.

10. Export Reports: Export data as CSV or PDF for reporting purposes.