Angel college of engineering and technology

Money Matter: A personal Finance Management App

Team code :

Name :BHARATHKUMAR S

Nm id : 1EDC925EA88306DAD734CDFAEIA45364

Name :KUNGUMA SANJAI C

Nm id :C3BF96637E330958D3E46961575DACCF

Name :BIRITHIKA BHARATHI. R

Nm id. :21248130197CIFI89A447F51E270920

Name :NIVETHA R

Nm id :337D1EDF6F93974B400498F821EAFFCT

- Key Features:
- 1. User Registration and Authentication:
- Sign up/Login with email, phone number, or social media accounts.
- Secure authentication methods, including two-factor authentication (2FA).
- 2. Dashboard:
- Summary of accounts, transactions, budgets, and financial goals.
- Visual representations like graphs and charts for an overview of financial health.
- 3. Account Management:
- Link bank accounts, credit cards, loans, and investments.
- Automatic transaction import and categorization.

- 4. Expense Tracking:
- Manual and automatic transaction logging.
- Categorization of expenses.
- Customizable tags and categories.
- 5. Budgeting:
- Set monthly/annual budgets.
- Track spending against budgets.
- Alerts and notifications for budget limits.
- 6. Financial Goals:
- Set and track savings goals.
- Progress visualization.
- Suggestions for achieving goals.

- Design and Development Process:
- 1. Requirement Analysis:
- Identify and document all features and functionalities.
- Create user personas and scenarios.
- 2. Design:
- Wireframing and prototyping.
- UI/UX design focusing on simplicity and usability.
- 3. Development:
- Set up the development environment.
- Implement features incrementally, starting with core functionalities.
- Regular testing and quality assurance.
- 4. Deployment:
- Set up continuous integration/continuous deployment (CI/CD) pipelines.
- Deploy to app stores (Apple App Store, Google Play Store).
- 5. Maintenance:
- Regular updates and feature enhancements.
- Monitor user feedback and address issues promptly.

```
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, databaseHelpe    }          }       }   }}
@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.jav                )
                )
                //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
```

```
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 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 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 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 = {
```

```
                    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 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
mport 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

mport 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())
    }
```

```
.close()
        db.close()
        return users
    }

}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}”)
                    }
                }
            }
        }

    }
}
```

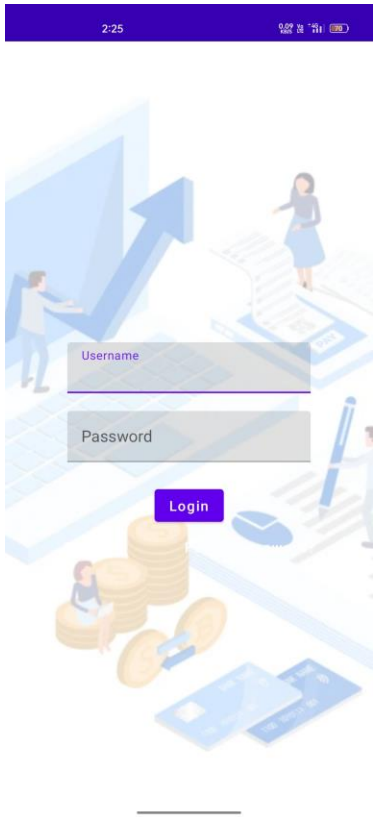
```
Package com.example.expensestracker.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)

val Purple500 = Color(0xFF6200EE)

val Purple700 = Color(0xFF3700B3)

val Teal200 = Color(0xFF03DAC5)
```

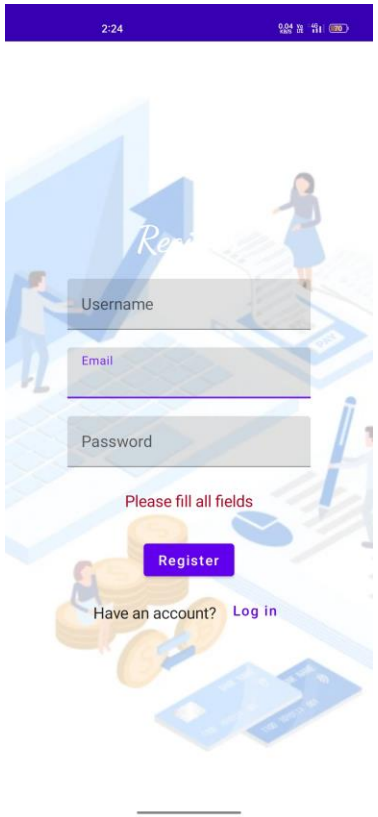
```
Package com.example.expensestracker.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.Shapes

import androidx.compose.ui.unit.dp

val Shapes = Shapes(

    small = RoundedCornerShape(4.dp),

    medium = RoundedCornerShape(4.dp),

    large = RoundedCornerShape(0.dp)

)
```

```
Package com.example.expensestracker.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme

import androidx.compose.material.MaterialTheme

import androidx.compose.material.darkColors

import androidx.compose.material.lightColors

import androidx.compose.runtime.Composable


private val DarkColorPalette = darkColors(

    primary = Purple200,

    primaryVariant = Purple700,

    secondary = Teal200

)

private val LightColorPalette = lightColors(

    primary = Purple500,

    primaryVariant = Purple700,

    secondary = Teal200
```

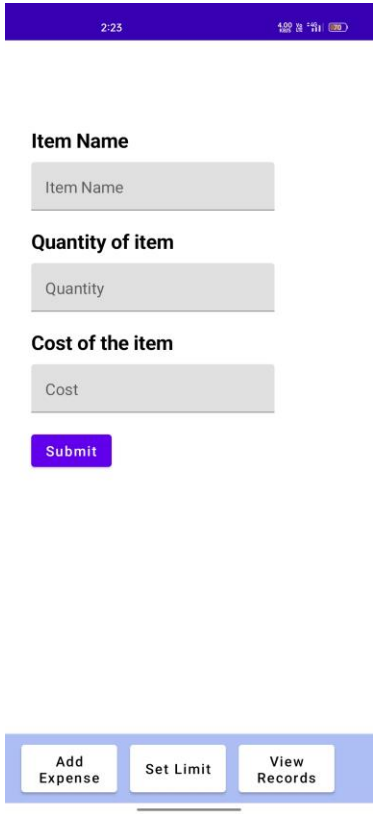
```
    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
)

@Composable
fun ExpensesTrackerTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

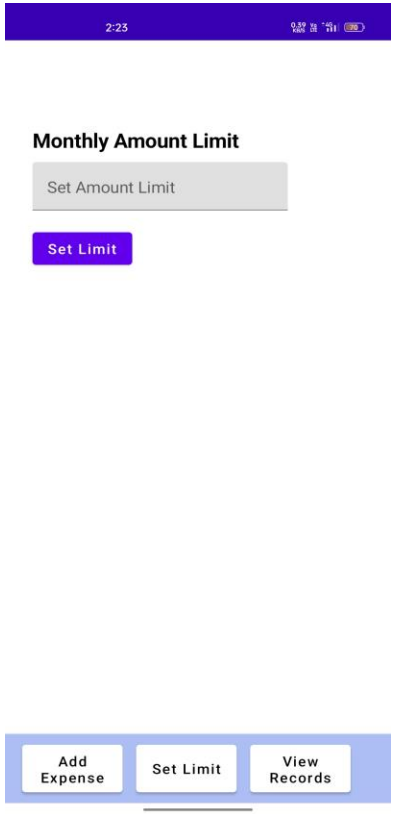
```
Package com.example.expensestracker.ui.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)
```


2:25
Username
Password
Login


2:24
Username
Email
Password
Please fill all fields
Register
Have an account?
Log in


2:23
Welcome To
Expense Tracker
Add
Expense
Set Limit
View
Records


2:23
Item Name
Item Name
Quantity of item
Quantity
Cost of the item
Cost
Submit
Add Expense
Set Limit
View Records


2:23
Monthly Amount Limit
Set Amount Limit
Set Limit
Add Expense
Set Limit
View Records


2:23
View Records
Add Expense
Set Limit
View Records