**Mini Project: Mini ERP**

**MainActivity.kt**

package com.example.expensetracker  
  
import android.content.Intent  
import android.content.SharedPreferences  
import android.os.Bundle  
import android.view.Menu  
import android.view.MenuItem  
import android.view.View  
import android.widget.Button  
import android.widget.TextView  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import androidx.recyclerview.widget.ItemTouchHelper  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.gson.Gson  
import com.google.gson.reflect.TypeToken  
import java.text.SimpleDateFormat  
import java.util.\*  
import kotlin.collections.ArrayList  
const val *ADD\_EXPENSE\_REQUEST* = 1  
const val *PREFS\_NAME* = "ExpensePrefs"  
const val *EXPENSE\_LIST\_KEY* = "expense\_list"  
class MainActivity : AppCompatActivity(), ExpenseAdapter.OnItemDeleteListener {  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var adapter: ExpenseAdapter  
 private lateinit var totalExpenseText: TextView  
 private lateinit var addExpenseButton: Button  
 private lateinit var expenseChartButton: Button  
 private lateinit var sharedPreferences: SharedPreferences  
 private lateinit var emptyStateText: TextView // For showing empty message  
 private val expenseList = ArrayList<Expense>()  
 private var totalExpense = 0.0  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
 // Initialize SharedPreferences  
 sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE)  
 // Set up toolbar  
 setSupportActionBar(findViewById(R.id.toolbar))  
 supportActionBar?.title = "My Expenses"  
 // Initialize views  
 recyclerView = findViewById(R.id.expenseRecyclerView)  
 totalExpenseText = findViewById(R.id.totalExpenseText)  
 addExpenseButton = findViewById(R.id.addExpenseButton)  
 expenseChartButton = findViewById(R.id.expenseChartButton)  
 emptyStateText = findViewById(R.id.emptyStateText) // Initialize empty state view  
 // Setup RecyclerView  
 adapter = ExpenseAdapter(expenseList)  
 adapter.setOnItemDeleteListener(this)  
 recyclerView.layoutManager = LinearLayoutManager(this)  
 recyclerView.adapter = adapter  
 // Load saved expenses  
 loadExpenses()  
 // Add swipe to delete functionality  
 setupSwipeToDelete()  
 // Update UI based on whether list is empty  
 updateEmptyState()  
 // Button click listeners  
 addExpenseButton.setOnClickListener **{** val intent = Intent(this, AddExpenseActivity::class.java)  
 startActivityForResult(intent, ADD\_EXPENSE\_REQUEST)  
 **}** expenseChartButton.setOnClickListener **{** if (expenseList.isEmpty()) {  
 showEmptyListAlert()  
 } else {  
 val intent = Intent(this, ChartActivity::class.java)  
 intent.putExtra("totalExpense", totalExpense)  
 intent.putExtra("expenseList", expenseList)  
 startActivity(intent)  
 }  
 **}** }  
 private fun showEmptyListAlert() {  
 AlertDialog.Builder(this)  
 .setTitle("No Expenses")  
 .setMessage("There are no expenses to display. Please add some expenses first.")  
 .setPositiveButton("OK") **{** dialog, \_ **->** dialog.dismiss() **}** .create()  
 .show()  
 }  
 private fun updateEmptyState() {  
 if (expenseList.isEmpty()) {  
 recyclerView.visibility = View.GONE  
 emptyStateText.visibility = View.VISIBLE  
 expenseChartButton.isEnabled = false  
 } else {  
 recyclerView.visibility = View.VISIBLE  
 emptyStateText.visibility = View.GONE  
 expenseChartButton.isEnabled = true  
 }  
 }  
 private fun loadExpenses() {  
 try {  
 val json = sharedPreferences.getString(EXPENSE\_LIST\_KEY, null)  
 if (json != null) {  
 val type = object : TypeToken<ArrayList<Expense>>() {}.type  
 val savedList = Gson().fromJson<ArrayList<Expense>>(json, type)  
 expenseList.clear()  
 expenseList.addAll(savedList)  
 calculateTotalExpense()  
 adapter.notifyDataSetChanged()  
 }  
 } catch (e: Exception) {  
 // Handle corrupted preference data  
 AlertDialog.Builder(this)  
 .setTitle("Error")  
 .setMessage("Could not load saved expenses. Starting fresh.")  
 .setPositiveButton("OK") **{** dialog, \_ **->** dialog.dismiss() **}** .create()  
 .show()  
 }  
 }  
 private fun saveExpenses() {  
 try {  
 val editor = sharedPreferences.edit()  
 val json = Gson().toJson(expenseList)  
 editor.putString(EXPENSE\_LIST\_KEY, json)  
 editor.apply()  
 } catch (e: Exception) {  
 // Handle saving error  
 AlertDialog.Builder(this)  
 .setTitle("Error")  
 .setMessage("Failed to save expenses")  
 .setPositiveButton("OK") **{** dialog, \_ **->** dialog.dismiss() **}** .create()  
 .show()  
 }  
 }  
 private fun setupSwipeToDelete() {  
 val itemTouchHelper = ItemTouchHelper(object : ItemTouchHelper.SimpleCallback(  
 0, ItemTouchHelper.LEFT or ItemTouchHelper.RIGHT  
 ) {  
 override fun onMove(  
 recyclerView: RecyclerView,  
 viewHolder: RecyclerView.ViewHolder,  
 target: RecyclerView.ViewHolder  
 ): Boolean = false  
  
 override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {  
 val position = viewHolder.adapterPosition  
 showDeleteConfirmationDialog(position)  
 }  
 })  
 itemTouchHelper.attachToRecyclerView(recyclerView)  
 }  
 private fun showDeleteConfirmationDialog(position: Int) {  
 if (position < 0 || position >= expenseList.size) return  
  
 AlertDialog.Builder(this)  
 .setTitle("Delete Expense")  
 .setMessage("Are you sure you want to delete this expense?")  
 .setPositiveButton("Delete") **{** \_, \_ **->** expenseList.removeAt(position)  
 adapter.notifyItemRemoved(position)  
 calculateTotalExpense()  
 saveExpenses()  
 updateEmptyState() // Check if list is now empty  
 **}** .setNegativeButton("Cancel") **{** dialog, \_ **->** dialog.dismiss()  
 adapter.notifyItemChanged(position)  
 **}** .create()  
 .show()  
 }  
 override fun onItemDelete(position: Int) {  
 showDeleteConfirmationDialog(position)  
 }  
 private fun addSampleData() {  
 try {  
 val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())  
 expenseList.add(Expense("Food", 15.50, dateFormat.parse("10/05/2023"), "Lunch at cafe"))  
 expenseList.add(Expense("Transport", 5.20, dateFormat.parse("11/05/2023"), "Bus ticket"))  
 expenseList.add(Expense("Shopping", 45.99, dateFormat.parse("12/05/2023"), "New shoes"))  
 expenseList.add(Expense("Entertainment", 12.00, dateFormat.parse("13/05/2023"), "Movie ticket"))  
 calculateTotalExpense()  
 adapter.notifyDataSetChanged()  
 saveExpenses()  
 updateEmptyState()  
 } catch (e: Exception) {  
 AlertDialog.Builder(this)  
 .setTitle("Error")  
 .setMessage("Failed to add sample data")  
 .setPositiveButton("OK") **{** dialog, \_ **->** dialog.dismiss() **}** .create()  
 .show()  
 }  
 }  
 private fun calculateTotalExpense() {  
 try {  
 totalExpense = expenseList.sumOf **{** it.amount **}** totalExpenseText.text = String.format("Total: ₹%.2f", totalExpense)  
 } catch (e: Exception) {  
 totalExpenseText.text = "Total: ₹0.00"  
 }  
 }  
 override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
 menuInflater.inflate(R.menu.main\_menu, menu)  
 return true  
 }  
 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 super.onActivityResult(requestCode, resultCode, data)  
  
 if (requestCode == ADD\_EXPENSE\_REQUEST && resultCode == RESULT\_OK) {  
 try {  
 val newExpense = data?.getSerializableExtra("newExpense") as? Expense  
 newExpense?.let **{** expenseList.add(it)  
 calculateTotalExpense()  
 adapter.notifyDataSetChanged()  
 saveExpenses()  
 updateEmptyState()  
 **}** } catch (e: Exception) {  
 AlertDialog.Builder(this)  
 .setTitle("Error")  
 .setMessage("Failed to add new expense")  
 .setPositiveButton("OK") **{** dialog, \_ **->** dialog.dismiss() **}** .create()  
 .show()  
 }  
 }  
 }  
 override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 return when (item.itemId) {  
 R.id.menu\_settings -> true  
 R.id.menu\_about -> true  
 else -> super.onOptionsItemSelected(item)  
 }  
 }  
}

**ExpenseAdapter.kt**

package com.example.expensetracker  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageButton  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import java.text.SimpleDateFormat  
import java.util.\*  
  
class ExpenseAdapter(private val expenseList: ArrayList<Expense>) :  
 RecyclerView.Adapter<ExpenseAdapter.ExpenseViewHolder>() {  
 interface OnItemDeleteListener {  
 fun onItemDelete(position: Int)  
 }  
 private var deleteListener: OnItemDeleteListener? = null  
  
 fun setOnItemDeleteListener(listener: OnItemDeleteListener) {  
 this.deleteListener = listener  
 }  
 class ExpenseViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 val categoryText: TextView = itemView.findViewById(R.id.categoryText)  
 val amountText: TextView = itemView.findViewById(R.id.amountText)  
 val dateText: TextView = itemView.findViewById(R.id.dateText)  
 val descriptionText: TextView = itemView.findViewById(R.id.descriptionText)  
 val deleteButton: ImageButton = itemView.findViewById(R.id.deleteButton)  
 }  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ExpenseViewHolder {  
 val itemView = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.item\_expense, parent, false)  
 return ExpenseViewHolder(itemView)  
 }  
  
 override fun onBindViewHolder(holder: ExpenseViewHolder, position: Int) {  
 val currentItem = expenseList[position]  
 val dateFormat = SimpleDateFormat("dd MMM yyyy", Locale.getDefault())  
 holder.categoryText.*text* = currentItem.category  
 holder.amountText.*text* = String.*format*("₹%.2f", currentItem.amount)  
 holder.dateText.*text* = dateFormat.format(currentItem.date)  
 holder.descriptionText.*text* = currentItem.description  
  
 holder.deleteButton.setOnClickListener **{** deleteListener?.onItemDelete(position)  
 **}** }  
 override fun getItemCount() = expenseList.size  
}

**Expense.kt**

package com.example.expensetracker  
import java.io.Serializable  
import java.util.\*  
data class Expense(  
 val category: String,  
 val amount: Double,  
 val date: Date,  
 val description: String  
) : Serializable

**chartActivity.kt**

package com.example.expensetracker  
import android.graphics.Color  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import com.github.mikephil.charting.charts.PieChart  
import com.github.mikephil.charting.data.PieData  
import com.github.mikephil.charting.data.PieDataSet  
import com.github.mikephil.charting.data.PieEntry  
import com.github.mikephil.charting.utils.ColorTemplate  
  
class ChartActivity : AppCompatActivity() {  
 private lateinit var pieChart: PieChart  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_chart)  
 // Set up toolbar  
 setSupportActionBar(findViewById(R.id.toolbar))  
 supportActionBar?.title = "Expense Chart"  
 supportActionBar?.setDisplayHomeAsUpEnabled(true)  
 pieChart = findViewById(R.id.pieChart)  
 // Get data from intent (in a real app, you'd get from database)  
 val expenseList = intent.getSerializableExtra("expenseList") as ArrayList<Expense>  
 val totalExpense = intent.getDoubleExtra("totalExpense", 0.0)  
  
 setupPieChart(expenseList, totalExpense)  
 }  
 private fun setupPieChart(expenseList: ArrayList<Expense>, totalExpense: Double) {  
 // Group expenses by category  
 val categoryMap = *mutableMapOf*<String, Float>()  
 for (expense in expenseList) {  
 val currentAmount = categoryMap[expense.category] ?: 0f  
 categoryMap[expense.category] = currentAmount + expense.amount.toFloat()  
 }  
 // Create entries for the pie chart  
 val entries = ArrayList<PieEntry>()  
 for ((category, amount) in categoryMap) {  
 entries.add(PieEntry(amount, category))  
 }  
 val dataSet = PieDataSet(entries, "Expenses by Category")  
 dataSet.colors = ColorTemplate.MATERIAL\_COLORS.toList()  
 dataSet.valueTextColor = Color.*BLACK* dataSet.valueTextSize = 12f  
 val pieData = PieData(dataSet)  
 pieChart.data = pieData  
 pieChart.description.isEnabled = false  
 pieChart.centerText = "Total: $$totalExpense"  
 pieChart.animateY(1000)  
 pieChart.invalidate()  
 }  
 override fun onSupportNavigateUp(): Boolean {  
 finish()  
 return true  
 }  
}

**AddExpenseActivity.kt**

package com.example.expensetracker  
import android.app.DatePickerDialog  
import android.content.Intent  
import android.os.Bundle  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import androidx.appcompat.app.AppCompatActivity  
import java.text.SimpleDateFormat  
import java.util.\*  
const val *RESULT\_ADD* = 1  
const val *RESULT\_CANCEL* = 0  
class AddExpenseActivity : AppCompatActivity() {  
 private lateinit var categorySpinner: Spinner  
 private lateinit var amountEditText: EditText  
 private lateinit var dateEditText: EditText  
 private lateinit var descriptionEditText: EditText  
 private lateinit var saveButton: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_add\_expense)  
 // Set up toolbar  
 setSupportActionBar(findViewById(R.id.toolbar))  
 supportActionBar?.title = "Add New Expense"  
 supportActionBar?.setDisplayHomeAsUpEnabled(true)  
 // Initialize views  
 categorySpinner = findViewById(R.id.categorySpinner)  
 amountEditText = findViewById(R.id.amountEditText)  
 dateEditText = findViewById(R.id.dateEditText)  
 descriptionEditText = findViewById(R.id.descriptionEditText)  
 saveButton = findViewById(R.id.saveButton)  
 // Setup category spinner  
 val categories = *arrayOf*("Food", "Transport", "Shopping", "Entertainment", "Bills", "Other")  
 val adapter = ArrayAdapter(this, android.R.layout.*simple\_spinner\_item*, categories) adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 categorySpinner.*adapter* = adapter  
 // Set current date as default  
 val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())  
 dateEditText.setText(dateFormat.format(Date()))  
 dateEditText.setOnClickListener **{** showDatePicker()  
 **}** // Save button click listener  
 saveButton.setOnClickListener **{** if (validateInput()) {  
 val category = categorySpinner.*selectedItem*.toString()  
 val amount = amountEditText.*text*.toString().*toDouble*()  
 val date = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).parse(dateEditText.*text*.toString()) ?: Date()  
 val description = descriptionEditText.*text*.toString()  
 val newExpense = Expense(category, amount, date, description)  
 val resultIntent = Intent()  
 resultIntent.putExtra("newExpense", newExpense)  
 setResult(RESULT\_OK, resultIntent)  
 finish()  
 }  
 **}** }  
 private fun showDatePicker() {  
 val calendar = Calendar.getInstance()  
 val year = calendar.get(Calendar.*YEAR*)  
 val month = calendar.get(Calendar.*MONTH*)  
 val day = calendar.get(Calendar.*DAY\_OF\_MONTH*)  
 val datePickerDialog = DatePickerDialog(  
 this,  
 **{** \_, selectedYear, selectedMonth, selectedDay **->** val selectedDate = Calendar.getInstance()  
 selectedDate.set(selectedYear, selectedMonth, selectedDay)  
 dateEditText.setText(SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(selectedDate.*time*))  
 **}**,  
 year,  
 month,  
 day  
 )  
 datePickerDialog.show()  
 }  
 private fun validateInput(): Boolean {  
 var isValid = true  
 if (amountEditText.*text*.toString().*isEmpty*()) {  
 amountEditText.*error* = "Please enter amount"  
 isValid = false  
 } else {  
 try {  
 amountEditText.*text*.toString().*toDouble*()  
 } catch (e: NumberFormatException) {  
 amountEditText.*error* = "Please enter a valid number"  
 isValid = false  
 }  
 }  
 if (dateEditText.*text*.toString().*isEmpty*()) {  
 dateEditText.*error* = "Please select a date"  
 isValid = false  
 }  
 return isValid  
 }  
 override fun onSupportNavigateUp(): Boolean {  
 setResult(*RESULT\_CANCEL*)  
 finish()  
 return true  
 }  
}

**Output**



