**Mini Project: Mini ERP**

**MainActivity.kt**

package com.example.simpletodoapp  
  
import android.content.Context  
import android.content.SharedPreferences  
import android.graphics.Paint  
import android.os.Bundle  
import android.view.View  
import android.view.ViewGroup  
import android.widget.\*  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.content.ContextCompat  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.gson.Gson  
import com.google.gson.reflect.TypeToken  
import android.widget.Filter  
import android.widget.Filterable  
  
class MainActivity : AppCompatActivity() {  
  
 data class Todo(  
 val task: String,  
 val priority: String,  
 var isCompleted: Boolean  
 )  
  
 private lateinit var todoAdapter: TodoAdapter  
 private val todoList = mutableListOf<Todo>()  
  
 private lateinit var sharedPrefs: SharedPreferences  
 private val prefsKey = "TODO\_LIST"  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 sharedPrefs = getSharedPreferences("TodoPrefs", Context.MODE\_PRIVATE)  
 loadTodos()  
  
 val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)  
 todoAdapter = TodoAdapter(todoList)  
 recyclerView.layoutManager = LinearLayoutManager(this)  
 recyclerView.adapter = todoAdapter  
  
 val fabAdd = findViewById<ImageView>(R.id.fabAdd)  
 fabAdd.setOnClickListener { showAddTodoDialog() }  
  
 val searchView = findViewById<SearchView>(R.id.searchView)  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?) = false  
 override fun onQueryTextChange(newText: String?): Boolean {  
 todoAdapter.filter.filter(newText)  
 return true  
 }  
 })  
 }  
  
 private fun showAddTodoDialog() {  
 val dialogView = layoutInflater.inflate(R.layout.dialog\_add\_todo, null)  
 val editTextTodo = dialogView.findViewById<EditText>(R.id.editTextTodo)  
 val spinnerPriority = dialogView.findViewById<Spinner>(R.id.spinnerPriority)  
  
 ArrayAdapter.createFromResource(  
 this,  
 R.array.priority\_levels,  
 android.R.layout.simple\_spinner\_item  
 ).also { adapter ->  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 spinnerPriority.adapter = adapter  
 }  
  
 AlertDialog.Builder(this)  
 .setTitle("Add New Task")  
 .setView(dialogView)  
 .setPositiveButton("Add") { \_, \_ ->  
 val task = editTextTodo.text.toString()  
 val priority = spinnerPriority.selectedItem.toString()  
 if (task.isNotEmpty()) {  
 todoList.add(Todo(task, priority, false))  
 todoAdapter.updateList(todoList)  
 saveTodos()  
 Toast.makeText(this, "Task added", Toast.LENGTH\_SHORT).show()  
 } else {  
 Toast.makeText(this, "Task cannot be empty", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 .setNegativeButton("Cancel", null)  
 .show()  
 }  
  
 private fun saveTodos() {  
 val json = Gson().toJson(todoList)  
 sharedPrefs.edit().putString(prefsKey, json).apply()  
 }  
  
 private fun loadTodos() {  
 val json = sharedPrefs.getString(prefsKey, null)  
 if (json != null) {  
 val type = object : TypeToken<MutableList<Todo>>() {}.type  
 val loadedList: MutableList<Todo> = Gson().fromJson(json, type)  
 todoList.clear()  
 todoList.addAll(loadedList)  
 }  
 }  
  
 inner class TodoAdapter(private var todoList: MutableList<Todo>) :  
 RecyclerView.Adapter<TodoAdapter.TodoViewHolder>(), Filterable {  
  
 private var filteredList = todoList.toMutableList()  
  
 inner class TodoViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 val textViewTask: TextView = itemView.findViewById(R.id.textViewTask)  
 val textViewPriority: TextView = itemView.findViewById(R.id.textViewPriority)  
 val checkBox: CheckBox = itemView.findViewById(R.id.checkBox)  
 val imageViewDelete: ImageView = itemView.findViewById(R.id.imageViewDelete)  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TodoViewHolder {  
 val itemView = layoutInflater.inflate(R.layout.item\_todo, parent, false)  
 return TodoViewHolder(itemView)  
 }  
  
 override fun onBindViewHolder(holder: TodoViewHolder, position: Int) {  
 val currentItem = filteredList[position]  
  
 holder.textViewTask.text = currentItem.task  
 holder.textViewPriority.text = currentItem.priority  
 holder.checkBox.isChecked = currentItem.isCompleted  
  
 val priorityColor = when (currentItem.priority) {  
 "High" -> ContextCompat.getColor(this@MainActivity, R.color.priority\_high)  
 "Medium" -> ContextCompat.getColor(this@MainActivity, R.color.priority\_medium)  
 else -> ContextCompat.getColor(this@MainActivity, R.color.priority\_low)  
 }  
 holder.textViewPriority.setTextColor(priorityColor)  
  
 if (currentItem.isCompleted) {  
 holder.textViewTask.paintFlags = holder.textViewTask.paintFlags or Paint.STRIKE\_THRU\_TEXT\_FLAG  
 } else {  
 holder.textViewTask.paintFlags = holder.textViewTask.paintFlags and Paint.STRIKE\_THRU\_TEXT\_FLAG.inv()  
 }  
  
 holder.checkBox.setOnCheckedChangeListener { \_, isChecked ->  
 currentItem.isCompleted = isChecked  
 saveTodos()  
 notifyItemChanged(position)  
 }  
  
 holder.imageViewDelete.setOnClickListener {  
 val actualPosition = todoList.indexOf(currentItem)  
 todoList.removeAt(actualPosition)  
 filteredList.removeAt(position)  
 notifyDataSetChanged()  
 saveTodos()  
 Toast.makeText(this@MainActivity, "Task removed", Toast.LENGTH\_SHORT).show()  
 }  
 }  
  
 override fun getItemCount() = filteredList.size  
  
 fun updateList(newList: MutableList<Todo>) {  
 this.todoList = newList  
 this.filteredList = newList.toMutableList()  
 notifyDataSetChanged()  
 }  
  
 override fun getFilter(): Filter {  
 return object : Filter() {  
 override fun performFiltering(constraint: CharSequence?): FilterResults {  
 val query = constraint?.toString()?.lowercase()?.trim()  
 val results = if (query.isNullOrEmpty()) {  
 todoList  
 } else {  
 todoList.filter {  
 it.task.lowercase().contains(query)  
 }  
 }  
 val filterResults = FilterResults()  
 filterResults.values = results  
 return filterResults  
 }  
  
 override fun publishResults(constraint: CharSequence?, results: FilterResults?) {  
 filteredList = (results?.values as? List<Todo>)?.toMutableList() ?: mutableListOf()  
 notifyDataSetChanged()  
 }  
 }  
 }  
 }  
}

**Output**



