MainActivity.kt

package com.example.todolistapp  
  
import android.annotation.SuppressLint  
import android.app.AlarmManager  
import android.app.NotificationChannel  
import android.app.NotificationManager  
import android.app.PendingIntent  
import android.content.Context  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.os.Build  
import android.os.Bundle  
import android.os.Environment  
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.items  
import androidx.compose.foundation.text.KeyboardOptions  
import androidx.compose.material.\*  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.*Delete*import androidx.compose.material.icons.filled.*Edit*import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.text.input.ImeAction  
import androidx.compose.ui.text.style.TextDecoration  
import androidx.compose.ui.unit.dp  
import androidx.core.app.NotificationCompat  
import androidx.core.app.NotificationManagerCompat  
import androidx.core.content.ContextCompat  
import androidx.core.content.FileProvider  
import com.example.todolistapp.ui.theme.TodoListAppTheme  
import java.io.File  
import java.io.FileOutputStream  
import java.text.SimpleDateFormat  
import java.util.\*  
  
data class Task(  
 val title: String,  
 val description: String = "",  
 val dueDate: String,  
 val category: String,  
 val isDone: Boolean = false  
)  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 // Create notification channel  
 val channel = if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 NotificationChannel(  
 "reminder\_channel",  
 "Reminders",  
 NotificationManager.*IMPORTANCE\_HIGH* )  
 } else {  
 *TODO*("VERSION.SDK\_INT < O")  
 }  
 val manager = getSystemService(Context.*NOTIFICATION\_SERVICE*) as NotificationManager  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 manager.createNotificationChannel(channel)  
 }  
  
 *setContent* **{** TodoListAppTheme **{** Surface(modifier = Modifier.*fillMaxSize*()) **{** ToDoListApp()  
 **}  
 }  
 }** // Check and request permission for Android 13+  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 if (ContextCompat.checkSelfPermission(  
 this,  
 android.Manifest.permission.*POST\_NOTIFICATIONS* ) == PackageManager.*PERMISSION\_GRANTED* ) {  
 // Permission granted, proceed with functionality  
 } else {  
 // Request permission  
 requestPermissions(*arrayOf*(android.Manifest.permission.*POST\_NOTIFICATIONS*), 1001)  
 }  
 }  
 }  
}  
  
@Composable  
fun ToDoListApp() {  
 val context = *LocalContext*.current  
 var isDark by remember **{** *mutableStateOf*(false) **}** var taskTitle by remember **{** *mutableStateOf*("") **}** var taskDescription by remember **{** *mutableStateOf*("") **}** var dueDate by remember **{** *mutableStateOf*("") **}** var category by remember **{** *mutableStateOf*("") **}** var search by remember **{** *mutableStateOf*("") **}** val taskList = remember **{** *mutableStateListOf*<Task>() **}** val filteredList = if (search.*isBlank*()) taskList else taskList.*filter* **{  
 it**.title.*contains*(search, ignoreCase = true)  
 **}** MaterialTheme(colors = if (isDark) *darkColors*() else *lightColors*()) **{** Column(modifier = Modifier.*padding*(16.*dp*)) **{** TopAppBar(  
 title = **{** Text("\uD83D\uDCDD To-Do List") **}**,  
 actions = **{** Switch(  
 checked = isDark,  
 onCheckedChange = **{** isDark = **it }** )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(8.*dp*))  
  
 OutlinedTextField(  
 value = search,  
 onValueChange = **{** search = **it }**,  
 label = **{** Text("Search") **}**,  
 modifier = Modifier.*fillMaxWidth*()  
 )  
 OutlinedTextField(  
 value = taskTitle,  
 onValueChange = **{** taskTitle = **it }**,  
 label = **{** Text("Enter task") **}**,  
 modifier = Modifier.*fillMaxWidth*(),  
 keyboardOptions = KeyboardOptions.Default.copy(imeAction = ImeAction.Next)  
 )  
 OutlinedTextField(  
 value = taskDescription,  
 onValueChange = **{** taskDescription = **it }**,  
 label = **{** Text("Optional Description") **}**,  
 modifier = Modifier.*fillMaxWidth*(),  
 keyboardOptions = KeyboardOptions.Default.copy(imeAction = ImeAction.Next)  
 )  
 OutlinedTextField(  
 value = dueDate,  
 onValueChange = **{** dueDate = **it }**,  
 label = **{** Text("Due Date (yyyy-MM-dd HH:mm)") **}**,  
 modifier = Modifier.*fillMaxWidth*()  
 )  
 OutlinedTextField(  
 value = category,  
 onValueChange = **{** category = **it }**,  
 label = **{** Text("Category") **}**,  
 modifier = Modifier.*fillMaxWidth*()  
 )  
 Button(  
 onClick = **{** if (taskTitle.*isNotBlank*() && dueDate.*isNotBlank*()) {  
 val task = Task(taskTitle, taskDescription, dueDate, category)  
 taskList.add(task)  
 *setReminder*(context, task)  
 taskTitle = ""  
 taskDescription = ""  
 dueDate = ""  
 category = ""  
 }  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(top = 8.*dp*)  
 ) **{** Text("Add Task")  
 **}** Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 LazyColumn **{** *items*(filteredList) **{** task **->** Card(  
 elevation = 8.*dp*,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(vertical = 4.*dp*)  
 ) **{** Row(  
 modifier = Modifier.*padding*(16.*dp*),  
 verticalAlignment = Alignment.CenterVertically,  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Column **{** Text(  
 task.title,  
 style = MaterialTheme.typography.subtitle1,  
 textDecoration = if (task.isDone) TextDecoration.LineThrough else null  
 )  
 Text(  
 "\uD83D\uDCC5 ${task.dueDate} • \uD83D\uDCD2 ${task.category}",  
 style = MaterialTheme.typography.caption  
 )  
 **}** Row **{** Checkbox(  
 checked = task.isDone,  
 onCheckedChange = **{** val index = taskList.indexOf(task)  
 if (index >= 0) {  
 taskList[index] = task.copy(isDone = **it**)  
 }  
 **}** )  
 // Edit button  
 IconButton(  
 onClick = **{** taskTitle = task.title  
 taskDescription = task.description  
 dueDate = task.dueDate  
 category = task.category  
 **}** ) **{** Icon(Icons.Filled.*Edit*, contentDescription = "Edit", tint = MaterialTheme.colors.primary)  
 **}** // Delete button  
 IconButton(onClick = **{** taskList.remove(task)  
 **}**) **{** Icon(Icons.Filled.*Delete*, contentDescription = "Delete", tint = MaterialTheme.colors.error)  
 **}  
 }  
 }  
 }  
 }  
 }** Button(  
 onClick = **{** val exportText = taskList.*joinToString*("\n") **{** "${**it**.title} | ${**it**.dueDate} | ${**it**.category} | Done: ${**it**.isDone}"  
 **}** val fileName = "tasks\_${System.currentTimeMillis()}.txt"  
 val file = File(context.getExternalFilesDir(Environment.*DIRECTORY\_DOCUMENTS*), fileName)  
 FileOutputStream(file).*use* **{ it**.write(exportText.*toByteArray*()) **}** val uri = FileProvider.getUriForFile(  
 context,  
 "${context.*packageName*}.provider",  
 file  
 )  
 val shareIntent = Intent(Intent.*ACTION\_SEND*).*apply* **{** *type* = "text/plain"  
 putExtra(Intent.*EXTRA\_STREAM*, uri)  
 addFlags(Intent.*FLAG\_GRANT\_READ\_URI\_PERMISSION*)  
 **}** context.startActivity(Intent.createChooser(shareIntent, "Export tasks"))  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(top = 16.*dp*)  
 ) **{** Text("Backup / Export Tasks")  
 **}  
 }  
 }**}  
  
@SuppressLint("ScheduleExactAlarm")  
fun setReminder(context: Context, task: Task) {  
 if (task.dueDate.*isBlank*()) return  
  
 val alarmManager = context.getSystemService(Context.*ALARM\_SERVICE*) as AlarmManager  
 val intent = Intent(context, ReminderReceiver::class.*java*).*apply* **{** putExtra("task", task.title)  
 **}** val pendingIntent = PendingIntent.getBroadcast(  
 context,  
 task.title.hashCode(),  
 intent,  
 PendingIntent.*FLAG\_UPDATE\_CURRENT* or PendingIntent.*FLAG\_IMMUTABLE* )  
 try {  
 val formatter = SimpleDateFormat("yyyy-MM-dd HH:mm", Locale.getDefault())  
 val time = formatter.parse(task.dueDate)?.*time* ?: return  
 alarmManager.setExactAndAllowWhileIdle(AlarmManager.*RTC\_WAKEUP*, time, pendingIntent)  
 } catch (e: Exception) {  
 e.printStackTrace()  
 }  
}

SplashActivity.kt

{  
 val context = *LocalContext*.current  
  
 LaunchedEffect(Unit) **{** delay(2500) // 2.5 seconds  
 context.startActivity(Intent(context, MainActivity::class.*java*))  
 **}** Surface(  
 modifier = Modifier.*fillMaxSize*(),  
 color = MaterialTheme.colors.background  
 ) **{** Column(  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally,  
 modifier = Modifier.*fillMaxSize*()  
 ) **{** Image(  
 painter = painterResource(id = R.drawable.*icon*),  
 contentDescription = "Splash Logo",  
 modifier = Modifier.*size*(200.*dp*)  
 )  
 Spacer(modifier = Modifier.*height*(20.*dp*))  
 Text("To-Do List", style = MaterialTheme.typography.h5)  
 **}  
 }**}

AndrpidMainfest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.todolistapp">  
  
 <uses-permission android:name="android.permission.POST\_NOTIFICATIONS" />  
 <uses-permission android:name="android.permission.SCHEDULE\_EXACT\_ALARM" />  
 <uses-permission android:name="android.permission.WAKE\_LOCK" />  
  
 <application  
 android:allowBackup="true"  
 android:label="@string/app\_name"  
 android:icon="@mipmap/ic\_launcher"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:theme="@style/Theme.TodoListApp"  
 android:supportsRtl="true">  
  
 <!-- ✅ Broadcast receiver for Alarm -->  
 <receiver android:name=".ReminderReceiver" />  
  
 <!-- ✅ Splash Screen -->  
 <activity android:name=".SplashActivity" />  
  
 <!-- ✅ Main Activity -->  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
  
 </application>  
  
</manifest>

ReminderReceiver.kt

package com.example.todolistapp  
  
import android.content.BroadcastReceiver  
import android.content.Context  
import android.content.Intent  
import android.media.RingtoneManager  
import android.media.Ringtone  
import android.os.Build  
import androidx.core.app.NotificationCompat  
import androidx.core.app.NotificationManagerCompat  
import androidx.core.content.ContextCompat  
  
class ReminderReceiver : BroadcastReceiver() {  
 override fun onReceive(context: Context, intent: Intent) {  
 val taskTitle = intent.getStringExtra("task") ?: "Your task"  
  
 // ✅ Check if notification permission is granted (Android 13 and above)  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 if (ContextCompat.checkSelfPermission(  
 context,  
 android.Manifest.permission.*POST\_NOTIFICATIONS* ) == android.content.pm.PackageManager.*PERMISSION\_GRANTED* ) {  
 // 🔔 Play sound  
 val alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.*TYPE\_NOTIFICATION*)  
 val ringtone: Ringtone = RingtoneManager.getRingtone(context, alarmSound)  
 ringtone.play()  
  
 // 📣 Show notification  
 val notification = NotificationCompat.Builder(context, "reminder\_channel")  
 .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
 .setContentTitle("Task Reminder")  
 .setContentText(taskTitle)  
 .setPriority(NotificationCompat.*PRIORITY\_HIGH*)  
 .build()  
  
 *with*(NotificationManagerCompat.from(context)) **{** notify(taskTitle.hashCode(), notification)  
 **}** }  
 } else {  
 // 🔔 For Android below API 33, no runtime permission needed  
 val alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.*TYPE\_NOTIFICATION*)  
 val ringtone: Ringtone = RingtoneManager.getRingtone(context, alarmSound)  
 ringtone.play()  
  
 // 📣 Show notification  
 val notification = NotificationCompat.Builder(context, "reminder\_channel")  
 .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
 .setContentTitle("Task Reminder")  
 .setContentText(taskTitle)  
 .setPriority(NotificationCompat.*PRIORITY\_HIGH*)  
 .build()  
  
 *with*(NotificationManagerCompat.from(context)) **{** notify(taskTitle.hashCode(), notification)  
 **}** }  
 }  
}