package com.example.exam

import android.app.Fragment

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.Box

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.padding

import androidx.compose.material3.MaterialTheme

import androidx.compose.material3.Scaffold

import androidx.compose.material3.Surface

import androidx.compose.material3.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.TextUnit

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.exam.ui.theme.ExamTheme

import com.google.gson.Gson

import android.content.ContentValues

import android.content.Context

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

import android.os.AsyncTask

import androidx.compose.foundation.layout.Spacer

import androidx.compose.foundation.layout.height

import androidx.compose.material3.Button

import androidx.compose.ui.Alignment

import androidx.lifecycle.lifecycleScope

import androidx.room.ColumnInfo

import androidx.room.Dao

import androidx.room.Database

import androidx.room.Delete

import androidx.room.Entity

import androidx.room.Insert

import androidx.room.PrimaryKey

import androidx.room.Query

import androidx.room.Room

import androidx.room.RoomDatabase

import androidx.room.TypeConverter

import androidx.room.TypeConverters

import java.lang.ref.WeakReference

import java.util.Date

import androidx.room.\*

import kotlinx.coroutines.CoroutineScope

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch

import androidx.compose.foundation.Image

import androidx.compose.ui.res.painterResource

@Entity(tableName = "users")

data class User (

@PrimaryKey val id: Int,

@ColumnInfo(name = "first\_name") val firstName: String?,

@ColumnInfo(name = "last\_name") val lastName: String?

)

@Dao

interface UserDao {

@Query("SELECT \* FROM users")

fun getAll(): List<User>

@Query("SELECT \* FROM users WHERE id IN (:userIds)")

fun loadAllByIds(userIds: IntArray): List<User>

@Query("SELECT \* FROM users WHERE first\_name LIKE :first AND " +

"last\_name LIKE :last LIMIT 1")

fun findByName(first: String, last: String): User

@Insert

fun insertAll(vararg users: User)

@Delete

fun delete(user: User)

}

@Database(entities = [User::class], version = 1, exportSchema = true)

abstract class AppDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

}

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

val usersText = StringBuilder()

// Use lifecycleScope to launch a coroutine

lifecycleScope.launch(Dispatchers.IO) **{**

// Ensure applicationContext is not null

val context: Context = applicationContext ?: return@launch

val db = Room.databaseBuilder(

context,

AppDatabase::class.java,

"database-name"

).build()

val userDao = db.userDao()

val user1 = User(id = 120, firstName = "Akhil", lastName = "Dominic")

// Perform the database insertion

userDao.insertAll(user1)

// Optionally, retrieve users from the database

val users: List<User> = userDao.getAll()

for (user in users) {

usersText.append("User ID: ${user.id}, Name: ${user.firstName} ${user.lastName}\n")

}

**}**

setContent **{**

ExamTheme **{**

// A surface container using the 'background' color from the theme

Surface(

modifier = Modifier.fillMaxSize(),

color = MaterialTheme.colorScheme.background

) **{**

Exam()

//Greeting(name = usersText.toString())

**}**

**}**

**}**

}

}

@Composable

fun Greeting(name: String, modifier: Modifier = Modifier) {

Text(

text = "Hello $name!",

modifier = modifier

)

}

@Composable

fun Exam(modifier: Modifier=Modifier)

{

Box(modifier= Modifier

.background(Color.Yellow)

.padding(60.dp)) **{**

Column(modifier= Modifier

.fillMaxSize()

.background(Color.Green)

.padding(20.dp),

horizontalAlignment = Alignment.CenterHorizontally)

**{**

Text("Hello",fontSize = 60.sp)

Text("How are yuo")

Image(painter = painterResource(R.drawable.dice\_1), contentDescription = "alte" )

Spacer(modifier = Modifier.height(120.dp))

Button(onClick = **{ }**) **{**

Text(text = "Roll")

**}**

**}**

**}**

}

Gradle file

plugins **{**

id("com.android.application")

id("org.jetbrains.kotlin.android")

id("com.google.devtools.ksp")

**}**

*android* **{**

namespace = "com.example.exam"

compileSdk = 34

defaultConfig **{**

applicationId = "com.example.exam"

minSdk = 24

targetSdk = 34

versionCode = 1

versionName = "1.0"

testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"

vectorDrawables **{**

useSupportLibrary = true

**}**

**}**

buildTypes **{**

*release* **{**

isMinifyEnabled = false

proguardFiles(

getDefaultProguardFile("proguard-android-optimize.txt"),

"proguard-rules.pro"

)

**}**

**}**

compileOptions **{**

sourceCompatibility = JavaVersion.*VERSION\_1\_8*

targetCompatibility = JavaVersion.*VERSION\_1\_8*

**}**

*kotlinOptions* **{**

jvmTarget = "1.8"

**}**

buildFeatures **{**

compose = true

**}**

composeOptions **{**

kotlinCompilerExtensionVersion = "1.5.1"

**}**

packaging **{**

resources **{**

excludes += "/META-INF/{AL2.0,LGPL2.1}"

**}**

**}**

**}**

*dependencies* **{**

*implementation*("androidx.room:room-runtime:2.6.1")

*ksp* ("androidx.room:room-compiler:2.6.1")

*implementation*("androidx.core:core-ktx:1.12.0")

*implementation*("com.google.code.gson:gson:2.8.7")

*implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")

*implementation*("androidx.activity:activity-compose:1.8.2")

*implementation*(platform("androidx.compose:compose-bom:2023.08.00"))

*implementation*("androidx.compose.ui:ui")

*implementation*("androidx.compose.ui:ui-graphics")

*implementation*("androidx.compose.ui:ui-tooling-preview")

*implementation*("androidx.compose.material3:material3")

*implementation*("androidx.room:room-common:2.6.1")

*implementation*("androidx.room:room-ktx:2.6.1")

*testImplementation*("junit:junit:4.13.2")

*androidTestImplementation*("androidx.test.ext:junit:1.1.5")

*androidTestImplementation*("androidx.test.espresso:espresso-core:3.5.1")

*androidTestImplementation*(platform("androidx.compose:compose-bom:2023.08.00"))

*androidTestImplementation*("androidx.compose.ui:ui-test-junit4")

*debugImplementation*("androidx.compose.ui:ui-tooling")

*debugImplementation*("androidx.compose.ui:ui-test-manifest")

**}**

App level dependency

// Top-level build file where you can add configuration options common to all sub-projects/modules.

plugins **{**

id("com.android.application") *version* "8.2.2" *apply* false

id("org.jetbrains.kotlin.android") *version* "1.9.0" *apply* false

id("com.google.devtools.ksp") *version* "1.9.22-1.0.16" *apply* false

**}**

Notifications

fun showNotification(context: Context, title: String, message: String) {

// NotificationManager

val notificationManager = context.getSystemService(Context.*NOTIFICATION\_SERVICE*) as NotificationManager

// Notification channel (for Android Oreo and above)

val channelId = "your\_channel\_id"

val channelName = "Your Channel Name"

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {

val channel = NotificationChannel(channelId, channelName, NotificationManager.*IMPORTANCE\_DEFAULT*)

notificationManager.createNotificationChannel(channel)

}

// Notification builder

val notificationBuilder = NotificationCompat.Builder(context, channelId)

.setContentTitle(title)

.setContentText(message)

.setSmallIcon(android.R.drawable.*ic\_dialog\_info*)

.setAutoCancel(true) // Dismisses the notification when tapped

// Show the notification

val notificationId = 1 // Change this ID if you have multiple notifications

notificationManager.notify(notificationId, notificationBuilder.build())

}