Here's a step-by-step guide to add a testing module for an **Android Kotlin To-Do List App** that uses Room Database without ViewModel:

**1. Dependencies for Testing**

Add the required dependencies to your build.gradle (app-level):

groovy

CopyEdit

dependencies {

// Testing libraries

testImplementation 'junit:junit:4.13.2'

androidTestImplementation 'androidx.test.ext:junit:1.1.5'

androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

// Room testing

testImplementation "androidx.room:room-testing:2.5.0"

}

**2. Create a Test Database**

Room provides an in-memory database for testing, which does not persist data after the process finishes. Create a TestDatabase instance.

**3. Setup Database Testing**

Here’s an example of how to set up a testing module for your Room database:

**Example Test Class**

kotlin

CopyEdit

package com.example.todolistapp

import androidx.room.Room

import androidx.test.ext.junit.runners.AndroidJUnit4

import androidx.test.platform.app.InstrumentationRegistry

import com.example.todolistapp.data.TodoDao

import com.example.todolistapp.data.TodoDatabase

import com.example.todolistapp.data.TodoEntity

import org.junit.After

import org.junit.Assert.assertEquals

import org.junit.Before

import org.junit.Test

import org.junit.runner.RunWith

import java.io.IOException

@RunWith(AndroidJUnit4::class)

class TodoDatabaseTest {

private lateinit var database: TodoDatabase

private lateinit var todoDao: TodoDao

@Before

fun setup() {

// Create an in-memory database for testing

val context = InstrumentationRegistry.getInstrumentation().targetContext

database = Room.inMemoryDatabaseBuilder(context, TodoDatabase::class.java)

.allowMainThreadQueries() // For testing only

.build()

todoDao = database.todoDao()

}

@After

@Throws(IOException::class)

fun teardown() {

database.close()

}

@Test

fun insertTodoAndRetrieve() {

// Insert a to-do item

val todo = TodoEntity(id = 1, title = "Test Todo", description = "This is a test.")

todoDao.insert(todo)

// Retrieve the to-do item

val todos = todoDao.getAllTodos()

assertEquals(1, todos.size)

assertEquals("Test Todo", todos[0].title)

}

@Test

fun deleteTodo() {

// Insert a to-do item

val todo = TodoEntity(id = 1, title = "Test Todo", description = "This is a test.")

todoDao.insert(todo)

// Delete the to-do item

todoDao.delete(todo)

// Verify that the database is empty

val todos = todoDao.getAllTodos()

assertEquals(0, todos.size)

}

}

**4. Database Classes**

**Entity Class**

kotlin

CopyEdit

package com.example.todolistapp.data

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "todo\_table")

data class TodoEntity(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val title: String,

val description: String

)

**DAO**

kotlin

CopyEdit

package com.example.todolistapp.data

import androidx.room.\*

@Dao

interface TodoDao {

@Insert(onConflict = OnConflictStrategy.REPLACE)

fun insert(todo: TodoEntity)

@Query("SELECT \* FROM todo\_table")

fun getAllTodos(): List<TodoEntity>

@Delete

fun delete(todo: TodoEntity)

}

**Database**

kotlin

CopyEdit

package com.example.todolistapp.data

import androidx.room.Database

import androidx.room.RoomDatabase

@Database(entities = [TodoEntity::class], version = 1, exportSchema = false)

abstract class TodoDatabase : RoomDatabase() {

abstract fun todoDao(): TodoDao

}

**5. Run Tests**

Execute the test cases using Android Studio or the command line:

* **In Android Studio**: Right-click the test class and select "Run".
* **Command Line**: Use ./gradlew connectedAndroidTest.

This setup ensures a clean testing environment and validates your Room database's basic CRUD operations.