Họ tên: Lê Phúc Hưng

MSSV: 20215276

Mã lớp: 151902

Môn học: Phát triển ứng dụng cho thiết bị di động (IT4785)

Source Code: https://github.com/lephuchung/HustMobile

Bài: Lesson 14 – Build bus schedule app

Contents

1. Before you begin

2. Download the starter code

3. Add dependencies

Add the following dependencies to the app:
build.gradle.kts

```
    implementation("androidx.room:room-ktx:${rootProject.extra["room_version"]}")
    implementation("androidx.room:room-runtime:${rootProject.extra["room_version"]}")
    ksp("androidx.room:room-compiler:${rootProject.extra["room_version"]}")
```

app/build.gradle.kts

```
set("room_version", "2.5.2")
```

4. Create a room entity

Convert the current Bus Schedule data class into a Room Entity.

The following image shows a sample of what the final data table looks like, including the schema and Entity property

```
@Entity(tableName = "Schedule")
data class BusSchedule(
    @PrimaryKey
    val id: Int,
    @NonNull
    @ColumnInfo(name = "stop_name")
    val stopName: String,
    @NonNull
    @ColumnInfo(name = "arrival_time")
    val arrivalTimeInMillis: Int
)
```

5. Create a data access object

Create a data access object (DAO) to access the database. The DAO provides a method to retrieve all the items in the database and a method to retrieve a single item with the name of the bus stop. Make sure to order the schedule by arrival time.

```kotlin
@Dao
interface BusScheduleDao {
    @Query(
        """
        SELECT * FROM schedule
        ORDER BY arrival_time ASC
        """
    )
    fun getAll(): Flow<List<BusSchedule>>

    @Query(
        """
        SELECT * FROM schedule
        WHERE stop_name = :stopName
        ORDER BY arrival_time ASC
        """
    )
    fun getByStopName(stopName: String): Flow<List<BusSchedule>>
}
```

6. Create a database instance

Create a Room database that uses the Entity and your DAO. The database initializes itself with data from the assets/database/bus_schedule.db file in the starter code

```kotlin
@Database(entities = arrayOf(BusSchedule::class), version = 1)
abstract class AppDatabase: RoomDatabase() {
    abstract fun busScheduleDao(): BusScheduleDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                Room.databaseBuilder(
                    context,
                    AppDatabase::class.java,
                     name: "app_database"
                )
                    .createFromAsset( databaseFilePath: "database/bus_schedule.db")
                    // Wipes and rebuilds instead of migrating if no Migration object.
                    // Migration is not part of this codelab.
                    .fallbackToDestructiveMigration()
                    .build()
                    .also {
                        INSTANCE = it
                    }
            }
        }
    }
}
```

7. Update the ViewModel

Update the ViewModel to retrieve data from the DAO and provide it to the UI instead of supplying sample data. Make sure to leverage both of your DAO methods to supply data for the list and for individual stops

```
class BusScheduleViewModel(private val busScheduleDao: BusScheduleDao): ViewModel() {
    // Get full bus schedule from Room DB
    fun getFullSchedule(): Flow<List<BusSchedule>> = busScheduleDao.getAll()
    // Get bus schedule based on the stop name from Room DB
    fun getScheduleFor(stopName: String): Flow<List<BusSchedule>> =
        busScheduleDao.getByStopName(stopName)

    companion object {
        val factory : ViewModelProvider.Factory = viewModelFactory {
            initializer {
                val application = (this[APPLICATION_KEY] as BusScheduleApplication)
                BusScheduleViewModel(application.database.busScheduleDao())
            }
        }
    }
}
```

8. Solution code