WIA2007 Mobile Application Development
Semester 1, Session 2022/2023
Practical 10 (App Data Storage (Part 2))

Task 1: Understanding Today's Project and Preparation

This practical exercise will help you to understand the three major components of Room Persistence Library to work with a local database (SQLite). We will be creating a simple mobile application that adds and display notes. The exercise consists of two activities (with java class and XML layout), seven classes (including one interface class), and one layout file for individual item viewing:

Activity
- ☑ MainActivity (MainActivity.java and activity_main.xml)
- ☑ AddNoteActivity (AddNoteActivity.java and activity_add_note.xml)

Classes
- ☑ MoodNote
- ☑ MoodNoteDao (interface class)
- ☑ MoodNoteRoomDatabase
- ☑ MoodNoteRepository
- ☑ MoodNoteViewModel
- ☑ MoodNoteViewHolder
- ☑ MoodNoteListAdapter

Layout
- ☑ individual_item_view.xml

Other Resource files
- ☑ values/strings.xml
- ☑ values/styles.xml
- ☑ values/dimens.xml

In the MainActivity, the user can view all the Mood Notes (Your Android Project Name is MoodNote) and click on the Floating Action Button to navigate to AddNoteActivity to add a new Mood Note.

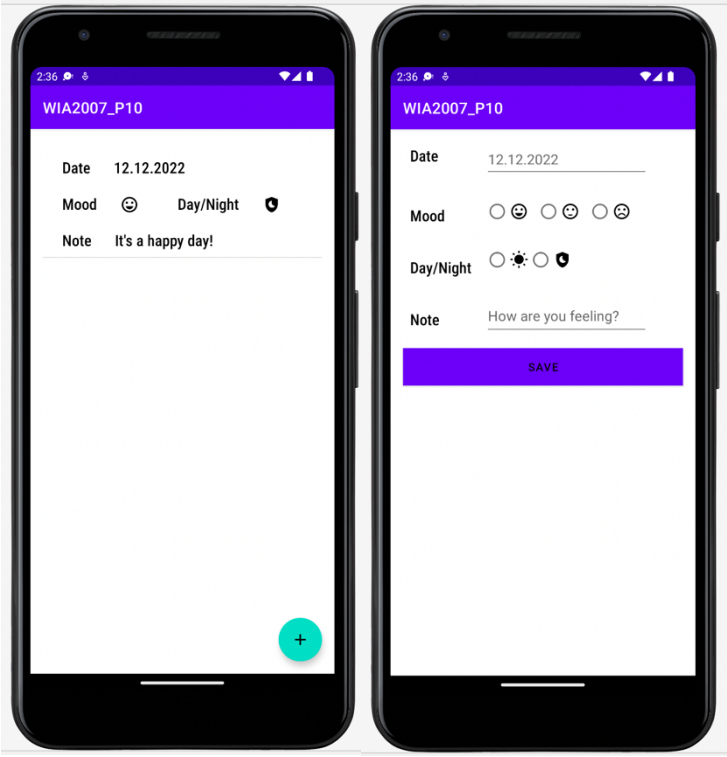

Figure 1(a) (left) and Figure 1(b) (right)

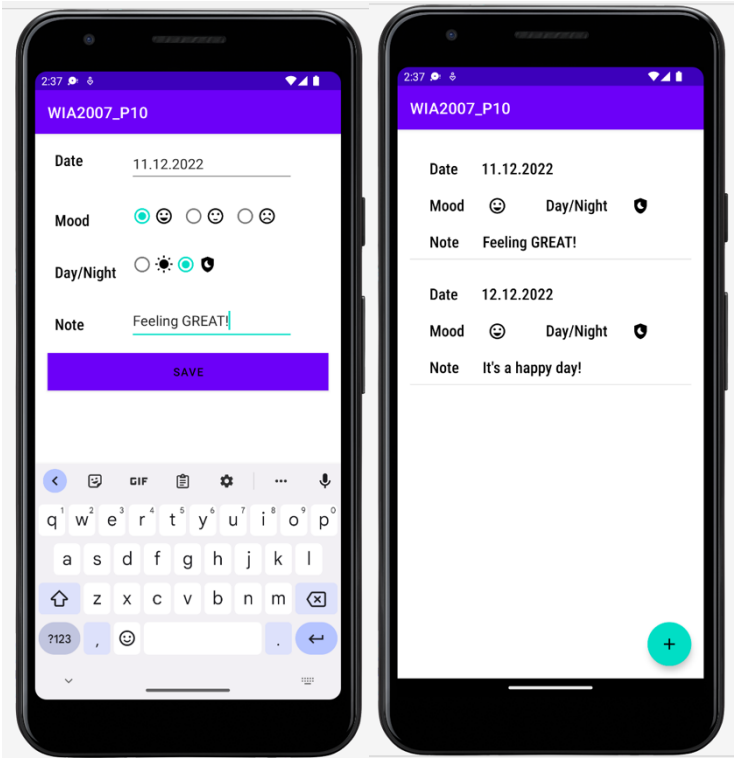

Figure 1(c) (left) and Figure 1(d) (right)

Figure 1: Subfigure (a) shows the default data is successfully entered to the database. When the floating action button (with add icon) is clicked, it will navigate to next activity – AddNoteActivity (refer to Subfigure (b)). After we fill in the mood note (Subfigure (c)), we can click Save (the button) and it will save the data and navigate back to MainActivity (Subfigure (d)) with the updated RecyclerView.

Now, let's first create two empty activities for MainActivity and AddNoteActivity. Then, we can move to the next task.

Task 1.1: Gradle Settings

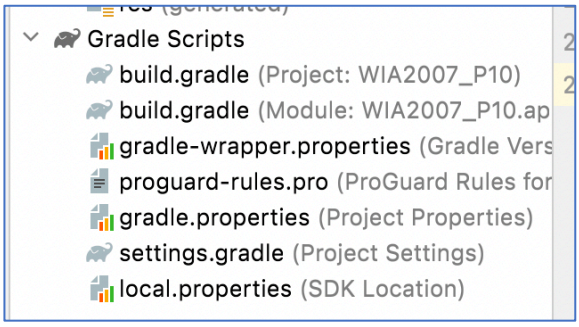

Figure 2: The Grade Scripts in the Project Panel.

In this task, we need to set up the dependencies that we will be using in this practical exercise. On your Gradle Scripts (build.gradle for Module), update your dependencies to these:

```
dependencies {
    implementation "androidx.appcompat:appcompat:$rootProject.appCompatVersion"

    // Dependencies for working with Architecture components
    // You'll probably have to update the version numbers in build.gradle
(Project)

    // Room components
    implementation "androidx.room:room-runtime:$rootProject.roomVersion"
    annotationProcessor "androidx.room:room-compiler:$rootProject.roomVersion"
    androidTestImplementation "androidx.room:room-
testing:$rootProject.roomVersion"

    // Lifecycle components
    implementation "androidx.lifecycle:lifecycle-
viewmodel:$rootProject.lifecycleVersion"
    implementation "androidx.lifecycle:lifecycle-
livedata:$rootProject.lifecycleVersion"
    implementation "androidx.lifecycle:lifecycle-common-
java8:$rootProject.lifecycleVersion"

    // UI
    implementation
"androidx.constraintlayout:constraintlayout:$rootProject.constraintLayoutVersion"
    implementation
"com.google.android.material:material:$rootProject.materialVersion"

    // Testing
    testImplementation "junit:junit:$rootProject.junitVersion"
    androidTestImplementation "androidx.arch.core:core-
testing:$rootProject.coreTestingVersion"
    androidTestImplementation ("androidx.test.espresso:espresso-
core:$rootProject.espressoVersion", {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    androidTestImplementation
"androidx.test.ext:junit:$rootProject.androidxJunitVersion"
}
```

There is no version stated in the dependencies above in the Module Gradle file. So, to declare and centralize the version control, let's put these extensions in your Project Gradle File (build.gradle for Project):

```
ext {
    appCompatVersion = '1.5.1'
    constraintLayoutVersion = '2.1.4'
    coreTestingVersion = '2.1.0'
    lifecycleVersion = '2.3.1'
    materialVersion = '1.7.0'
    roomVersion = '2.4.3'
    // testing
    junitVersion = '4.13.2'
    espressoVersion = '3.5.0'
    androidxJunitVersion = '1.1.2'
}
```

Task 1.2: Resource Preparation

There are six drawable files that we must create for this practical exercise:

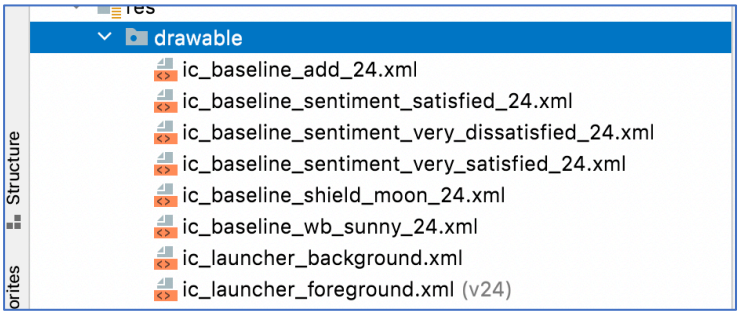

Figure 3: The drawable files for this project.

- Add icon for the floating action button (ic_baseline_add_24.xml)
- Neutral icon for the Mood Note (ic_baseline_sentiment_satisfied_24.xml)
- Unhappy icon for the Mood Note (ic_baseline_sentiment_very_dissatisfied_24.xml)
- Happy icon for the Mood Note (ic_baseline_sentiment_very_satisfied_24.xml)
- Night icon for the Mood Note (ic_baseline_shield_moon_24.xml)
- Day icon for the Mood Note (ic_baseline_wb_sunny_24.xml)

You may create the icon by selecting it when using "New Vector Asset" (refer to Figure 4) and choosing the clipart from the Android Resource Library (refer to Figure 5).

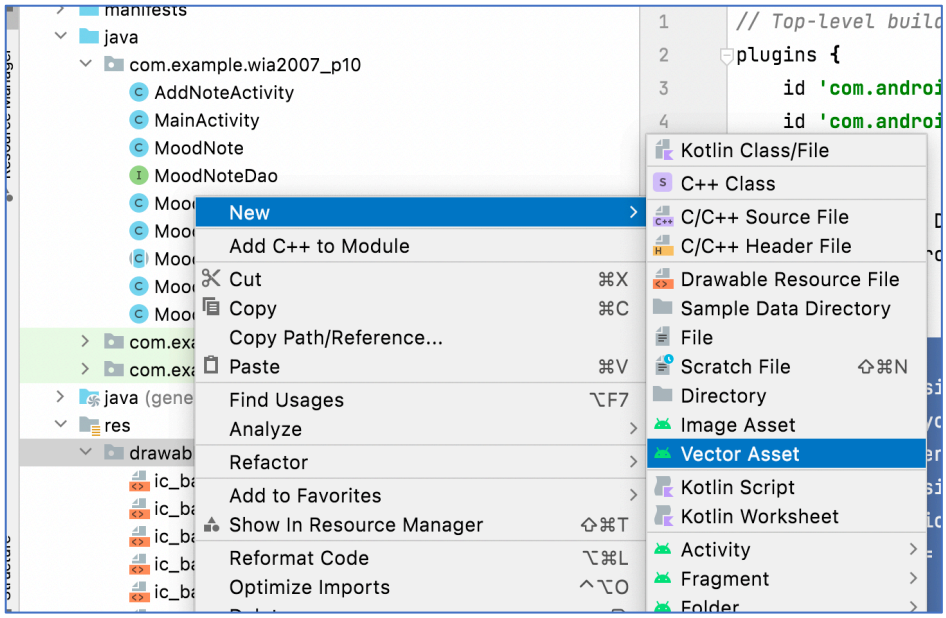

Figure 4: Create new vector asset.

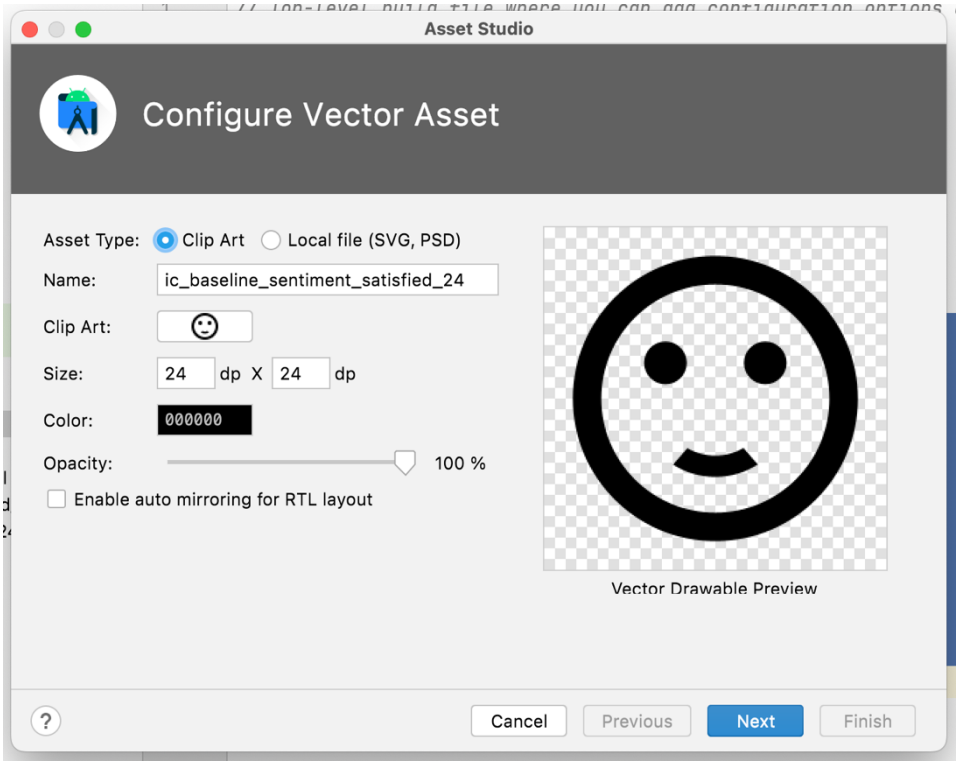

Figure 5: Create a new drawable using icon (Clip Art) from the Android Library.

Other than the drawable, you also need to create the resource value files:

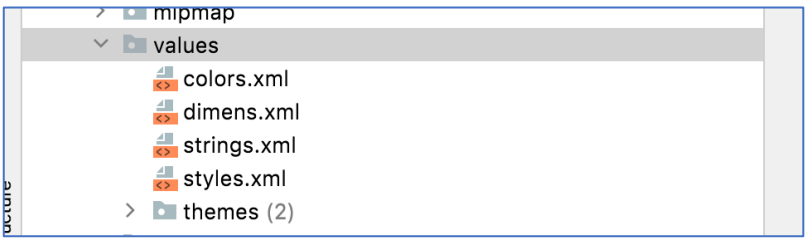

Figure 6: The resource value files.

In the dimens.xml, it consists of two dimensions:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">8dp</dimen>
    <dimen name="big_padding">16dp</dimen>
</resources>
```

In the strings.xml, it consists of three new string variables that we will use in the exercise:

```
<resources>
    <string name="app_name">WIA2007_P10</string>
    <string name="hint_word">Note…</string>
    <string name="button_save">Save</string>
    <string name="empty_not_saved">Note not saved because it is
empty.</string>
</resources>
```

In styles.xml, it contains the style for the title of the note (optional):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="note_title">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_marginBottom">8dp</item>
        <item name="android:paddingLeft">8dp</item>
        <item name="android:background">#CDDC39</item>
        <item
name="android:textAppearance">@android:style/TextAppearance.Large</it
em>
    </style>
</resources>
```

Task 2: Create Entity, Dao, and Room Database

There are three main components in Room Persistence Library, namely **Entity, Data Access Object (DAO),** and **Database** (please refer to Lecture 9 for recalling these items).

Hence, in this exercise, we will create an Entity called MoodNote (in **MoodNote.java class**), a DAO named MoodNoteDao (in **MoodNoteDao.java class**), and a Database called MoodNoteRoomDatabase (in **MoodNoteRoomDatabase.java class**):

Tips: To create the class file, just right click on the folder where it stores your usual MainActivity.java, click "New" and "Java Class" (refer to Figure 7). Then on the popped-up dialog, enter the class name and press Enter.

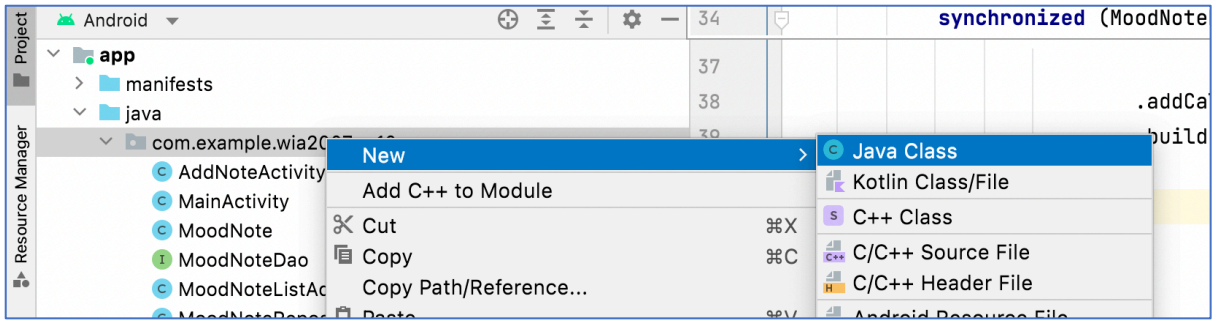

Figure 7: Create java class.

```java
import androidx.annotation.NonNull;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

//1. Room will use these data to create a table in sqlite database
@Entity
public class MoodNote {
    @PrimaryKey(autoGenerate = true)
    public int mNoteID;

    public String mNote;
    @NonNull
    public String mDate;
    @NonNull
    public int mMood;
    @NonNull
    public boolean mDayNight;
    public MoodNote(@NonNull String date, @NonNull int mood, @NonNull boolean
dayNight, String note) {
        this.mDate = date;
        this.mMood = mood;
        this.mDayNight = dayNight;
        this.mNote = note;
    }
    public String getmDate(){return this.mDate;}
    public int getmMood(){return this.mMood;}
    public boolean getmDayNight(){return this.mDayNight;}
    public String getmNote(){return this.mNote;}
}
```

```java
import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;

import java.util.List;

//2. Create the DAO to access to MoodNote database
@Dao
public interface MoodNoteDao {

    // allowing the insert of the same word multiple times by passing a
    // conflict resolution strategy
    // the convenience method - insert
    @Insert(onConflict = OnConflictStrategy.IGNORE)
    void insert(MoodNote note);

    // the query method
```

```
    @Query("DELETE FROM MoodNote")
    void deleteAll();

    // LiveData works with Room database to get instant update whenever there is
any changes
    @Query("SELECT * FROM MoodNote ORDER BY mDate ASC")
    LiveData<List<MoodNote>> getAscendingNote();
}
```

```
import android.content.Context;

import androidx.annotation.NonNull;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.sqlite.db.SupportSQLiteDatabase;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

// 3. Create a Room Database - a database abstraction layer on top of sqlite
@Database(entities = {MoodNote.class}, version = 1, exportSchema = false)
public abstract class MoodNoteRoomDatabase extends RoomDatabase {

    public abstract MoodNoteDao noteDao();

    private static volatile MoodNoteRoomDatabase INSTANCE;

    // We've created an ExecutorService with a fixed thread pool that you will
use to run database
    // operations asynchronously on a background thread.
    private static final int NUMBER_OF_THREADS = 4;
    static final ExecutorService databaseWriteExecutor =
            Executors.newFixedThreadPool(NUMBER_OF_THREADS);

    // getDatabase returns the singleton.
    // It'll create the database the first time it's accessed, using Room's
database builder to
    // create a RoomDatabase object in the application context from the
NoteRoomDatabase class
    // and names it "note_database".
    static MoodNoteRoomDatabase getDatabase(final Context context) {
        if (INSTANCE == null) {
            synchronized (MoodNoteRoomDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                                    MoodNoteRoomDatabase.class, "note_database")
                            .addCallback(sRoomDatabaseCallback)
                            .build();
                }
            }
        }
        return INSTANCE;
    }

    private static RoomDatabase.Callback sRoomDatabaseCallback = new
RoomDatabase.Callback() {
        @Override
        public void onCreate(@NonNull SupportSQLiteDatabase db) {
            super.onCreate(db);

            // If you want to keep data through app restarts,
            // comment out the following block
            databaseWriteExecutor.execute(() -> {
                // Populate the database in the background.
```

```
                // If you want to start with more notes, just add them.
                MoodNoteDao dao = INSTANCE.noteDao();
                dao.deleteAll();

                // insert a default instance
                MoodNote note = new MoodNote("12.12.2022", 1, false, "It's a
happy day!");
                dao.insert(note);
            });
        }
    };
}
```

Task 3: Create Support Classes

In this task, we will create various support classes, including the Repository class and the ViewModel.

The Repository class is a data abstract class used to communicate with various data resources (in one mobile app, we might have multiple app data storages). Meanwhile, the ViewModel stands between the Repository and the UI. It separates the data and the UI to enable flexibility, thus better controlling various screen sizes (refer to Introduction to ViewModel for more explanation).

For Repository, we will name it as **MoodNoteRepository** class, and for ViewMode, we will name it as **MoodNoteViewModel**:

```
import android.app.Application;
import androidx.lifecycle.LiveData;
import java.util.List;

// 4. Create Repository class to access multiple data sources (in this case, we
have only one).
class MoodNoteRepository {

    private MoodNoteDao mNoteDao;
    private LiveData<List<MoodNote>> mAllNotes;

    // The DAO is passed into the repository constructor as opposed to the whole
database. This is
    // because you only need access to the DAO, since it contains all the
read/write methods for
    // the database. There's no need to expose the entire database to the
repository.
    MoodNoteRepository(Application application) {
        MoodNoteRoomDatabase db = MoodNoteRoomDatabase.getDatabase(application);
        mNoteDao = db.noteDao();
        mAllNotes = mNoteDao.getAscendingNote();
    }

    // The getAllNotes method returns the LiveData list of notes from Room; we
can do this because
    // of how we defined the getAscendingNote() method to return LiveData in the
MoodNoteDao.
    // Room executes all queries on a separate thread. Then observed LiveData
will notify the
    // observer on the main thread when the data has changed.
    LiveData<List<MoodNote>> getAllNotes() {
        return mAllNotes;
    }
```

```
    //We need to not run the insert on the main thread, so we use the
ExecutorService we created
    // in the MoodNoteRoomDatabase to perform the insert on a background thread.
    // You must call this on a non-UI thread or your app will throw an exception.
Room ensures
    // that you're not doing any long running operations on the main thread,
blocking the UI.
    void insert(MoodNote note) {
        MoodNoteRoomDatabase.databaseWriteExecutor.execute(() -> {
            mNoteDao.insert(note);
        });
    }
}
```

```
import android.app.Application;

import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import java.util.List;

// 5. Created a class called MoodNoteViewModel that gets the Application as a
parameter and
// extends AndroidViewModel.
public class MoodNoteViewModel extends AndroidViewModel {

    //Added a private member variable to hold a reference to the repository.
    private MoodNoteRepository mRepository;
    private final LiveData<List<MoodNote>> mAllNotes;

    //Implemented a constructor that creates the MoodNoteRepository.
    //In the constructor, initialized the allNotes LiveData using the repository.
    public MoodNoteViewModel (Application application) {
        super(application);
        mRepository = new MoodNoteRepository(application);
        mAllNotes = mRepository.getAllNotes();
    }
    //Added a getAllNotes() method to return a cached list of words.
    LiveData<List<MoodNote>> getAllNotes() { return mAllNotes; }

    // Created a wrapper insert() method that calls the Repository's insert()
method.
    // In this way, the implementation of insert() is encapsulated from the UI.
    public void insert(MoodNote note) { mRepository.insert(note); }
}
```

Task 4: The Layout

We are using RecyclerView to show a list of Mood Notes. To create a single view for each item, we must create a layout resource file called **individual_item_view.xml**:

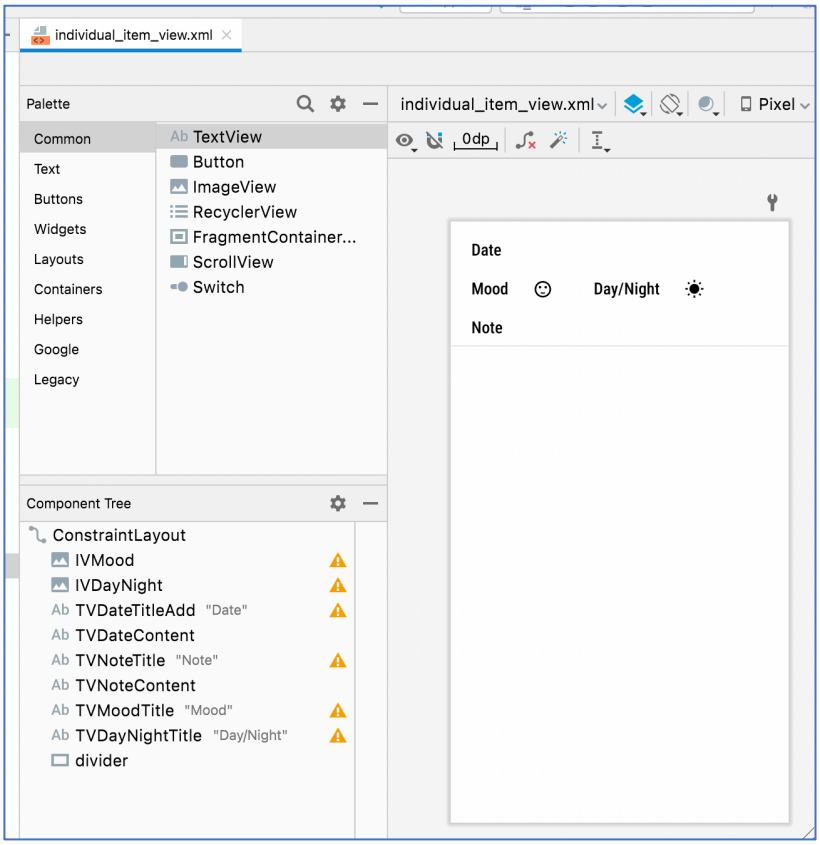

Figure 8: The design view for individual_item_view.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/IVMood"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        app:layout_constraintBottom_toBottomOf="@+id/TVMoodTitle"
        app:layout_constraintStart_toEndOf="@+id/TVMoodTitle"
        app:srcCompat="@drawable/ic_baseline_sentiment_satisfied_24" />

    <ImageView
        android:id="@+id/IVDayNight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        app:layout_constraintBottom_toBottomOf="@+id/TVDayNightTitle"
        app:layout_constraintStart_toEndOf="@+id/TVDayNightTitle"
        app:srcCompat="@drawable/ic_baseline_wb_sunny_24" />

    <TextView
        android:id="@+id/TVDateTitleAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="25dp"
        android:layout_marginTop="20dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Date"
```

```
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/TVDateContent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="@+id/TVDateTitleAdd"
        app:layout_constraintStart_toEndOf="@+id/TVDateTitleAdd" />

    <TextView
        android:id="@+id/TVNoteTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Note"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/TVMoodTitle"
        app:layout_constraintTop_toBottomOf="@+id/TVMoodTitle" />

    <TextView
        android:id="@+id/TVNoteContent"
        android:layout_width="275dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:paddingBottom="20dp"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toEndOf="@+id/TVNoteTitle"
        app:layout_constraintTop_toTopOf="@+id/TVNoteTitle" />

    <TextView
        android:id="@+id/TVMoodTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Mood"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/TVDateTitleAdd"
        app:layout_constraintTop_toBottomOf="@+id/TVDateTitleAdd" />

    <TextView
        android:id="@+id/TVDayNightTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Day/Night"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="@+id/TVMoodTitle"
        app:layout_constraintStart_toEndOf="@+id/IVMood" />

    <View
        android:id="@+id/divider"
        android:layout_width="409dp"
        android:layout_height="1dp"
```

```
        android:layout_marginBottom="10dp"
        android:background="?android:attr/listDivider"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Now, let's modify the **activity_main.xml** file to have one RecyclerView and one Floating Action Button. We will use the ***listitem*** attribute in the RecyclerView to bind with the individual_item_view.xml. After binding, you should see the following:

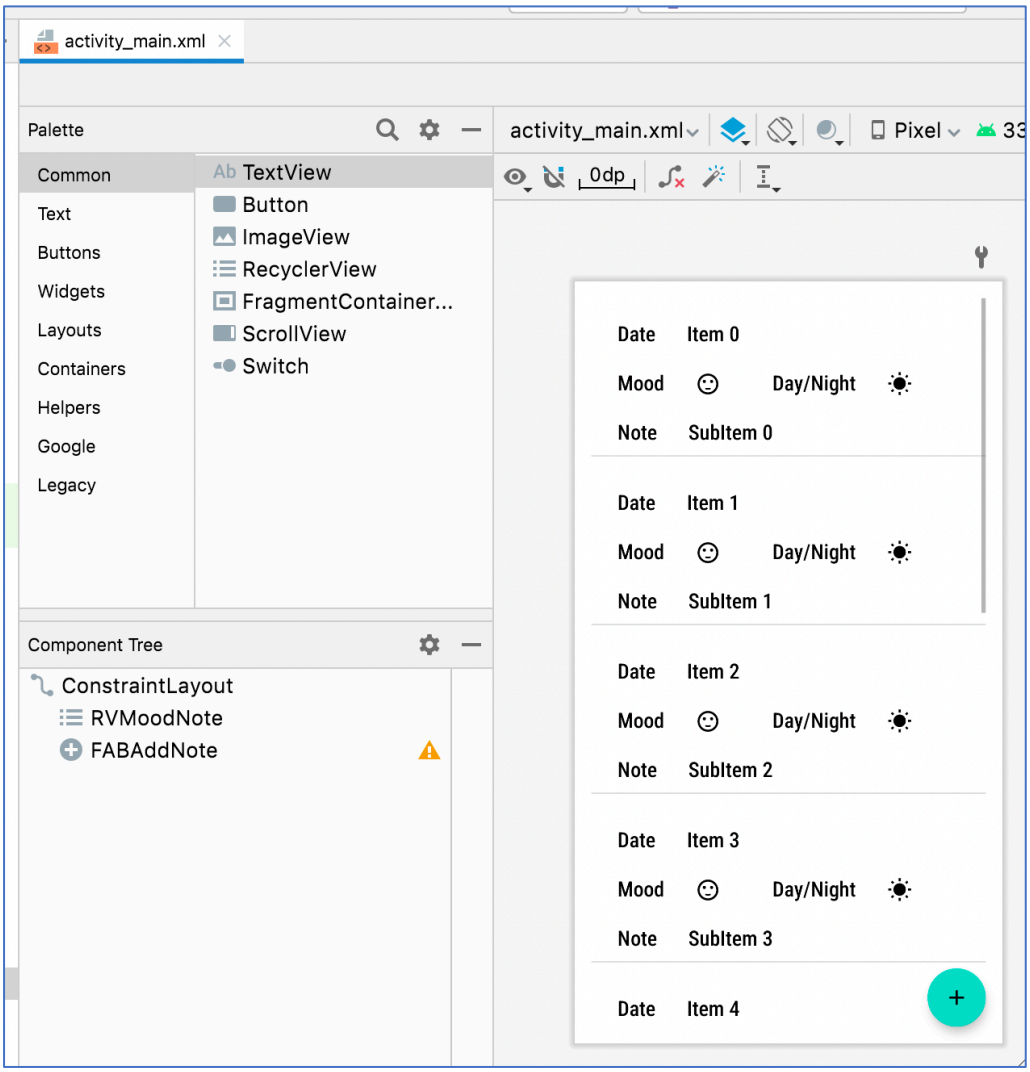

Figure 9: The design view for activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/RVMoodNote"
```

```
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:padding="@dimen/big_padding"
        android:scrollbars="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:listitem="@layout/individual_item_view" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/FABAddNote"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:contentDescription="Add Note"
        android:src="@drawable/ic_baseline_add_24"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Lastly, prep the last layout file – the **activity_add_note.xml**:

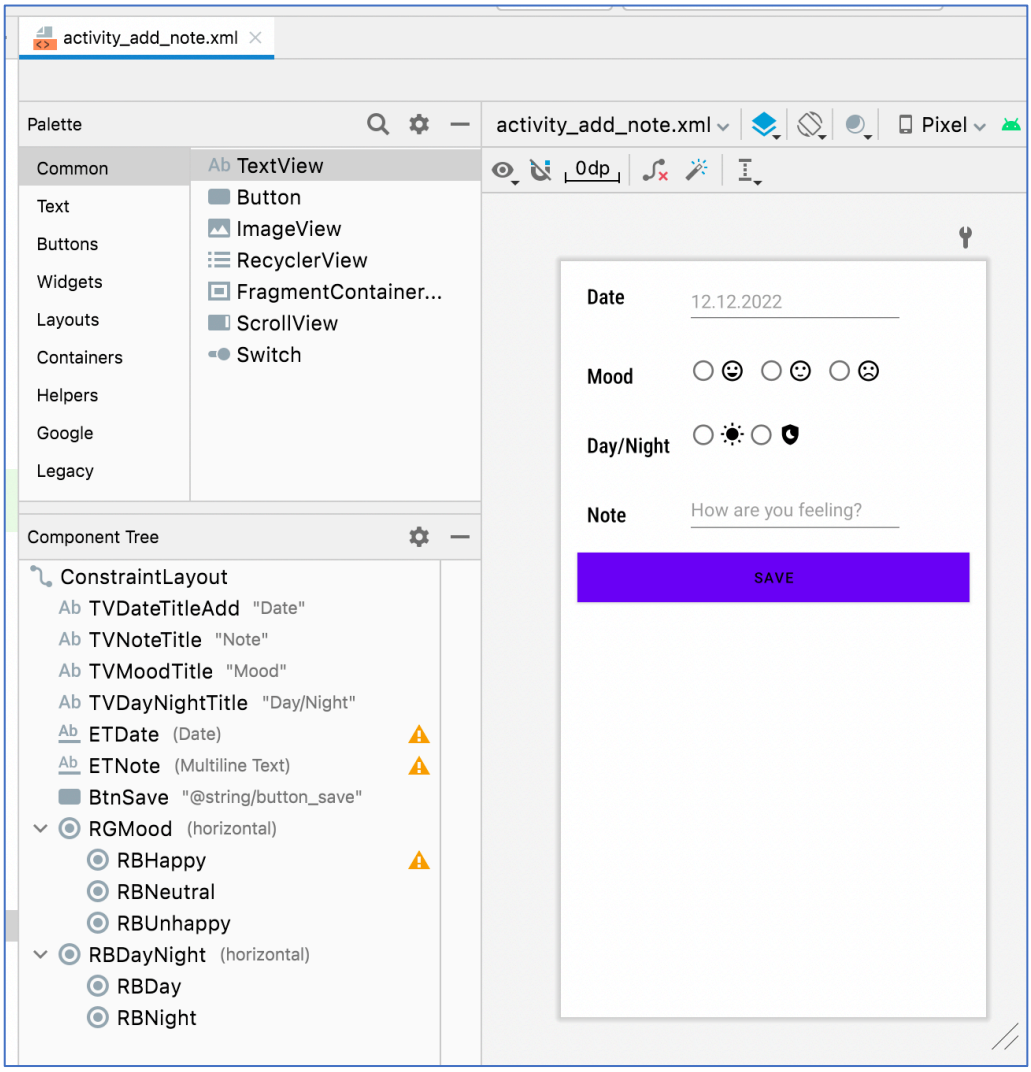

Figure 10: The design view for activity_add_note.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddNoteActivity">

    <TextView
        android:id="@+id/TVDateTitleAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="25dp"
        android:layout_marginTop="20dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Date"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/TVNoteTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Note"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/TVDayNightTitle"
        app:layout_constraintTop_toBottomOf="@+id/TVDayNightTitle" />

    <TextView
        android:id="@+id/TVMoodTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Mood"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/TVDateTitleAdd"
        app:layout_constraintTop_toBottomOf="@+id/TVDateTitleAdd" />

    <TextView
        android:id="@+id/TVDayNightTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:fontFamily="sans-serif-condensed-medium"
        android:text="Day/Night"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/TVMoodTitle"
        app:layout_constraintTop_toBottomOf="@+id/TVMoodTitle" />

    <EditText
        android:id="@+id/ETDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="60dp"
        android:layout_marginTop="15dp"
        android:ems="10"
        android:hint="12.12.2022"
        android:inputType="date"
        android:minHeight="48dp"
```

```
        app:layout_constraintStart_toEndOf="@+id/TVDateTitleAdd"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/ETNote"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:ems="10"
        android:gravity="start|top"
        android:hint="How are you feeling?"
        android:inputType="textMultiLine"
        android:minHeight="48dp"
        app:layout_constraintStart_toStartOf="@+id/ETDate"
        app:layout_constraintTop_toBottomOf="@+id/RBDayNight" />

    <Button
        android:id="@+id/BtnSave"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/big_padding"
        android:layout_marginTop="50dp"
        android:background="@color/purple_200"
        android:text="@string/button_save"
        android:textColor="@color/black"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ETNote" />

    <RadioGroup
        android:id="@+id/RGMood"
        android:layout_width="205dp"
        android:layout_height="46dp"
        android:layout_marginTop="20dp"
        android:orientation="horizontal"
        app:layout_constraintStart_toStartOf="@+id/ETDate"
        app:layout_constraintTop_toBottomOf="@+id/ETDate">

        <RadioButton
            android:id="@+id/RBHappy"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

android:drawableLeft="@drawable/ic_baseline_sentiment_very_satisfied_24"
            tools:ignore="TouchTargetSizeCheck" />

        <RadioButton
            android:id="@+id/RBNeutral"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:drawableLeft="@drawable/ic_baseline_sentiment_satisfied_24"
            tools:ignore="TouchTargetSizeCheck" />

        <RadioButton
            android:id="@+id/RBUnhappy"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"

android:drawableLeft="@drawable/ic_baseline_sentiment_very_dissatisfied_24"
            tools:ignore="TouchTargetSizeCheck" />
    </RadioGroup>

    <RadioGroup
        android:id="@+id/RBDayNight"
        android:layout_width="164dp"
        android:layout_height="59dp"
```

```
        android:layout_marginTop="15dp"
        android:orientation="horizontal"
        app:layout_constraintStart_toStartOf="@+id/ETDate"
        app:layout_constraintTop_toBottomOf="@+id/RGMood">

        <RadioButton
            android:id="@+id/RBDay"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:drawableLeft="@drawable/ic_baseline_wb_sunny_24" />

        <RadioButton
            android:id="@+id/RBNight"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:drawableLeft="@drawable/ic_baseline_shield_moon_24" />
    </RadioGroup>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Task 5: Add RecyclerView

To use RecyclerView, we are creating two more classes here – the **MoodNoteViewHolder** class and the **MoodNoteList** Adapter class, that will be in charge of obtaining the data from the database and displaying the data to RecyclerView.

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

class MoodNoteViewHolder extends RecyclerView.ViewHolder {
    private final TextView noteDate;
    private final TextView noteContent;
    private final ImageView noteMood;
    private final ImageView noteDayNight;

    private MoodNoteViewHolder(View itemView) {
        super(itemView);

        noteDate = itemView.findViewById(R.id.TVDateContent);
        noteContent = itemView.findViewById(R.id.TVNoteContent);
        noteMood = itemView.findViewById(R.id.IVMood);
        noteDayNight = itemView.findViewById(R.id.IVDayNight);
    }

    public void bind(String date, int mood, boolean daynight, String note) {
        noteDate.setText(date);
        noteContent.setText(note);
        if (mood == 1)

noteMood.setImageResource(R.drawable.ic_baseline_sentiment_very_satisfied_24);
        else if (mood == 2)

noteMood.setImageResource(R.drawable.ic_baseline_sentiment_satisfied_24);
        else

noteMood.setImageResource(R.drawable.ic_baseline_sentiment_very_dissatisfied_24);

        if (daynight == true)
```

```
            noteDayNight.setImageResource(R.drawable.ic_baseline_wb_sunny_24);
        else
            noteDayNight.setImageResource(R.drawable.ic_baseline_shield_moon_24);
    }

    static MoodNoteViewHolder create(ViewGroup parent) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.individual_item_view, parent, false);
        return new MoodNoteViewHolder(view);
    }
}
```

```
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.DiffUtil;
import androidx.recyclerview.widget.ListAdapter;

public class MoodNoteListAdapter extends ListAdapter<MoodNote,
MoodNoteViewHolder> {

    public MoodNoteListAdapter(@NonNull DiffUtil.ItemCallback<MoodNote>
diffCallback) {
        super(diffCallback);
    }

    @Override
    public MoodNoteViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
        return MoodNoteViewHolder.create(parent);
    }

    @Override
    public void onBindViewHolder(MoodNoteViewHolder holder, int position) {
        MoodNote current = getItem(position);
        holder.bind(current.getmDate(), current.getmMood(),
current.getmDayNight(), current.getmNote());
    }

    static class NoteDiff extends DiffUtil.ItemCallback<MoodNote> {

        @Override
        public boolean areItemsTheSame(@NonNull MoodNote oldItem, @NonNull
MoodNote newItem) {
            return oldItem == newItem;
        }

        @Override
        public boolean areContentsTheSame(@NonNull MoodNote oldItem, @NonNull
MoodNote newItem) {
            return oldItem.getmNote().equals(newItem.getmNote());
        }
    }
}
```

Task 6: The Backend

Lastly, let's complete our backend codes for **MainActivity.java** and **AddNoteActivity.java**:

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;
```

```
import androidx.recyclerview.widget.RecyclerView;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

public class MainActivity extends AppCompatActivity {

    public static final int NEW_NOTE_ACTIVITY_REQUEST_CODE = 1;
    private MoodNoteViewModel mNoteViewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RecyclerView recyclerView = findViewById(R.id.RVMoodNote);
        final MoodNoteListAdapter adapter = new MoodNoteListAdapter(new
MoodNoteListAdapter.NoteDiff());
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        mNoteViewModel = new
ViewModelProvider(this).get(MoodNoteViewModel.class);

        mNoteViewModel.getAllNotes().observe(this, notes -> {
            // Update the cached copy of the words in the adapter.
            adapter.submitList(notes);
        });

        FloatingActionButton fab = findViewById(R.id.FABAddNote);
        fab.setOnClickListener( view -> {
            Intent intent = new Intent(MainActivity.this, AddNoteActivity.class);
            startActivityForResult(intent, NEW_NOTE_ACTIVITY_REQUEST_CODE);
        });
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == NEW_NOTE_ACTIVITY_REQUEST_CODE && resultCode ==
RESULT_OK) {
            int mood =
Integer.parseInt(data.getStringExtra(AddNoteActivity.ExtraMood));
            boolean daynight =
Boolean.parseBoolean(data.getStringExtra(AddNoteActivity.ExtraDayNight));
            MoodNote note = new
MoodNote(data.getStringExtra(AddNoteActivity.ExtraDate), mood,
                    daynight, data.getStringExtra(AddNoteActivity.ExtraNote));
            mNoteViewModel.insert(note);
        } else {
            Toast.makeText(
                    getApplicationContext(),
                    R.string.empty_not_saved,
                    Toast.LENGTH_LONG).show();
        }
    }

}
```

```
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Button;
import android.widget.EditText;
```

```
import android.widget.RadioButton;

public class AddNoteActivity extends AppCompatActivity {

    public static final String ExtraDate = "date";
    public static final String ExtraMood = "mood";
    public static final String ExtraDayNight = "daynight";
    public static final String ExtraNote = "note";

    private EditText ETDate, ETNote;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_note);

        RadioButton RBHappy = findViewById(R.id.RBHappy);
        RadioButton RBNeutral = findViewById(R.id.RBNeutral);
        RadioButton RBUnhappy = findViewById(R.id.RBUnhappy);
        RadioButton RBDay = findViewById(R.id.RBDay);
        RadioButton RBNight = findViewById(R.id.RBNight);

        ETDate = findViewById(R.id.ETDate);
        ETNote = findViewById(R.id.ETNote);

        final Button button = findViewById(R.id.BtnSave);
        button.setOnClickListener(view -> {
            Intent replyIntent = new Intent();
            String date = ETDate.getText().toString();
            String note = ETNote.getText().toString();
            int mood;
            if (RBHappy.isChecked())
                mood = 1;
            else if (RBUnhappy.isChecked())
                mood = 3;
            else
                mood = 2;

            boolean dayNight = true;
            if (RBNight.isChecked())
                dayNight = false;
            if (TextUtils.isEmpty(ETDate.getText())) {
                setResult(RESULT_CANCELED, replyIntent);
            }
            else if (TextUtils.isEmpty(ETNote.getText())){
                setResult(RESULT_CANCELED, replyIntent);
            }
            else {
                replyIntent.putExtra(ExtraDate, date);
                replyIntent.putExtra(ExtraNote, note);
                replyIntent.putExtra(ExtraMood, Integer.toString(mood));
                replyIntent.putExtra(ExtraDayNight, Boolean.toString(dayNight));
                setResult(RESULT_OK, replyIntent);
            }
            finish();
        });
    }
}
```

After completing all the tasks, you should be able to add and view notes easily using this MoodNote app.

Submission

You are required to complete the exercise and submit the Android Studio Project to Spectrum before the next Tutorial class.