Task Core 2 – Spike: Communication between activities

Link to GitHub repository: https://github.com/aalexandros47/Software_Development_For_Mobile_Devices

Goals:

- Show that you can create an application that combines several different activities
- Use intents to make it easier for activities to share data in both directions.
- Include extra resources and graphics in the program.
- Add more complex user interface elements than in the last main task.
- Make good use of the sophisticated Kotlin features and code samples.

Tools and Resources Used

- Android Studio IDE
- Git and GitHub
- Kotlin programming language and XML files
- The course's modules
- Rating bar examples: https://www.javatpoint.com/android-rating-bar-example
- Implement scope functions: https://kotlinlang.org/docs/scope-functions.html
- Design custom themes: https://guides.codepath.com/android/developing-custom-themes
- Basic theme tutorial: https://www.geeksforgeeks.org/different-ways-to-change-or-add- themes- to-android-studio/

Knowledge Gaps and Solutions

Gap 1: Advanced UI widgets

I created a brand-new user interface widget called the RatingBar specifically for this task. Users can add input to the RatingBar by dragging or touching on it. Furthermore, customizations like the colors of picked and unselected sections can be used to produce distinctive styles; these will be covered in more detail in a later section.

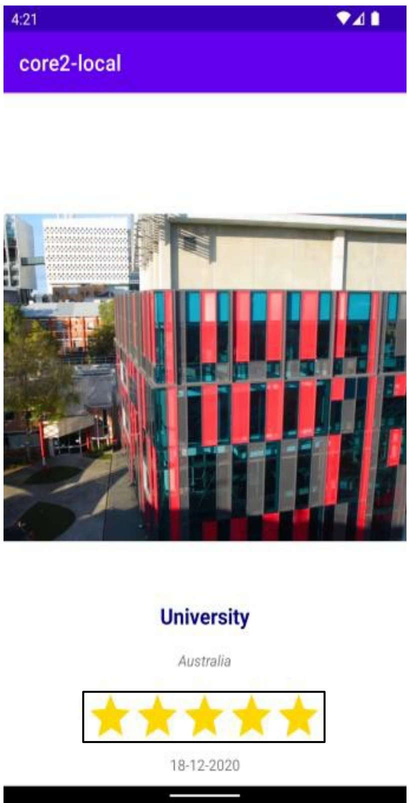
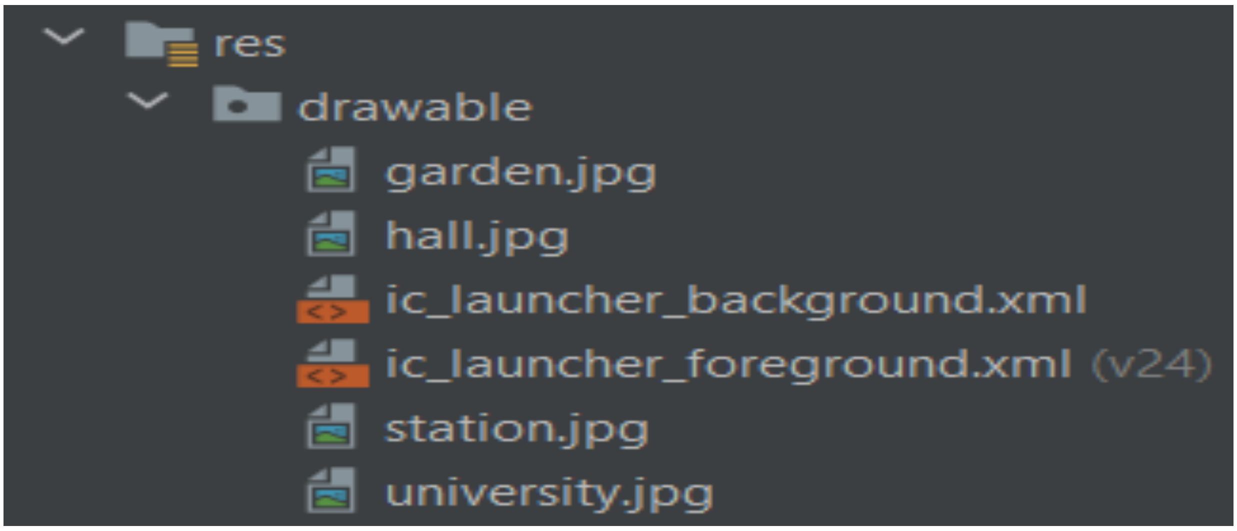


Figure 1: Rating bar highlighted in the squared box

Gap 2: Using available and addon resources

Additional images can be added for this assignment and stored in the /app/res/drawable folder. Although you may also right-click on the drawable folder to add files, I found that dragging and dropping the photos straight onto the Android Studio UI is the fastest way. Additionally, the photos are scaled correctly within the ImageView by using the android:scaleType="centerCrop" attribute, which improves the appearance of the images.



```
res
  drawable
    garden.jpg
    hall.jpg
    ic_launcher_background.xml
    ic_launcher_foreground.xml (v24)
    station.jpg
    university.jpg
```

Figure 2: File structure

Gap 3: Implementation of intents

When paired with the Parcelable interface, intents—which are used to move data across activities—improve efficiency by saving data on the disk rather than in memory (RAM).

Describe How I Plan to Implement It: Establishing the Parcelable Class

With characteristics like an image, name, location, date, and rating, I created a Parcelable class called Location. The rating was declared as a float because the RatingBar widget takes floating-point values, and the date was utilized to parse the Drawable image.

Making Use of Intents:

For each of the four clickable images, I made an Intent as a local variable inside the setOnClickListener event. Next, two parameters were passed to the putExtra function: nullable objects and a key name. The onCreate() override function was used to access these objects, which were declared in the MainActivity class.
Data Reception in the Detail Activity:

To receive the data supplied by the intent, I made a second nullable Location object in the DetailActivity class. When the activity was first started, this information was shown on the screen and assigned to the local Location object in DetailActivity.

```
// Main Activity
val universityImage = findViewById<ImageView>(R.id.university)
universityImage.setOnClickListener {
    val i = Intent(this, DetailActivity::class.java).apply {
        putExtra("image", university)
    }
    startActivity(i)
}

// Parcelable object
@Parcelize
data class Location(
    val image: Int, val name: String, val location: String, val date:
String, val rating: Float
) : Parcelable

// Detail Activity
location = intent.getParcelableExtra("image")
location?.let {
    // get data and assignments to attributes
}
```

Gap 4: Apply new styles to the application

The program's styles were centered on the RatingBar and the cover and header text that displayed the place name. To provide the user a recognizable and eye-catching appearance, the stars in the RatingBar of the detail activity were changed from light green to a gold-yellow hue, and the text color was changed to navy blue. I accomplished this by adding a new styles.xml file to the /app/res/values folder, defining the necessary themes, and allocating them to the appropriate layout file elements.

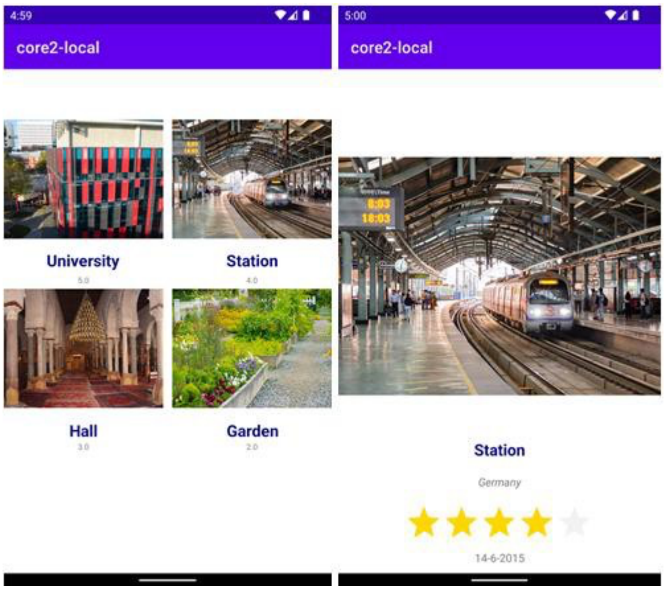


Figure 3: Blue navy text colour for headers, gold yellow for the stars on the Rating bar

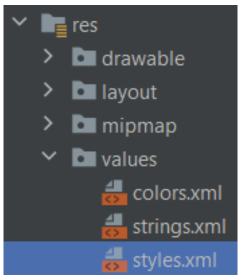


Figure 4: styles.xml added to the folder structure

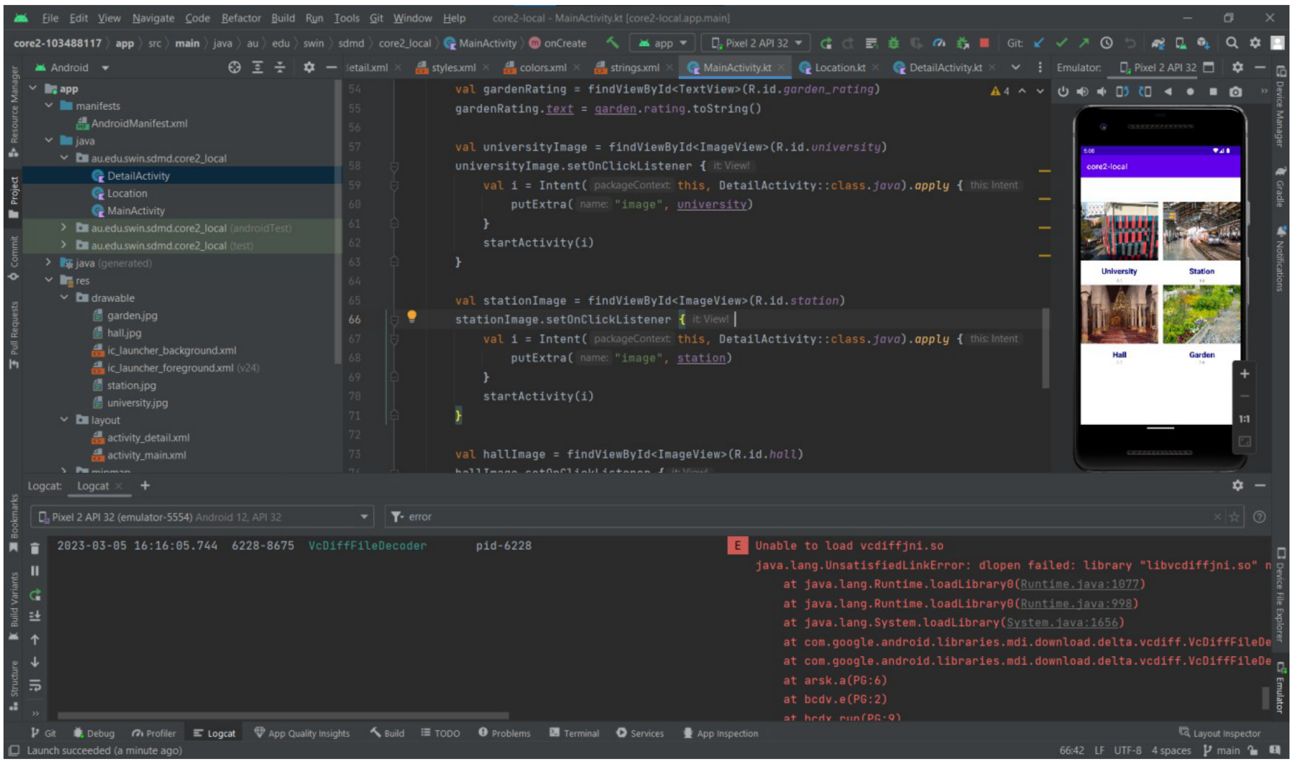
Code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="NavyText" parent="Theme.AppCompat">
        <item name="android:textColor">#000080</item>
    </style>
    <style name="RatingBar" parent="Theme.AppCompat">
        <item name="colorControlNormal">@color/grey</item>
        <item name="colorControlActivated">@color/yellow</item>
    </style>
</resources>
```

Code snippets to add to TextView and RatingBar elements in layout files: android:theme="@style/RatingBar" android:theme="@style/NavyText"

Gap 5: Take advantage of high-level features in the Android Studio IDE

In order to find and fix problems and enhance the program's functionality, I used Logcat and the debug console in this assignment to log and filter important error signals.



```
val gardenRating = findViewById<TextView>(R.id.garden_rating)
gardenRating.text = garden.rating.toString()

val universityImage = findViewById<ImageView>(R.id.university)
universityImage.setOnClickListener {
    val i = Intent(this, DetailActivity::class.java).apply {
        putExtra("image", university)
    }
    startActivity(i)
}

val stationImage = findViewById<ImageView>(R.id.station)
stationImage.setOnClickListener {
    val i = Intent(this, DetailActivity::class.java).apply {
        putExtra("image", station)
    }
    startActivity(i)
}

val hallImage = findViewById<ImageView>(R.id.hall)
```

```
Pixel 2 API 32 (emulator-5554) Android 12, API 32    error

2023-03-05 16:16:05.744  6228-8675  VcDiffFileDecoder    pid-6228    E  Unable to load vcdiffjni.so
    java.lang.UnsatisfiedLinkError: dlopen failed: library "libvcdiffjni.so" n
        at java.lang.Runtime.loadLibrary0(Runtime.java:1077)
        at java.lang.Runtime.loadLibrary0(Runtime.java:998)
        at java.lang.System.loadLibrary(System.java:1656)
        at com.google.android.libraries.mdi.download.delta.vcdiff.VcDiffFileDe
        at com.google.android.libraries.mdi.download.delta.vcdiff.VcDiffFileDe
        at arsk.a(PG:6)
        at bcdv.e(PG:2)
```

Launch succeeded (a minute ago)

Figure 5: Logcat and debugging

Gap 6: Kotlin advanced features

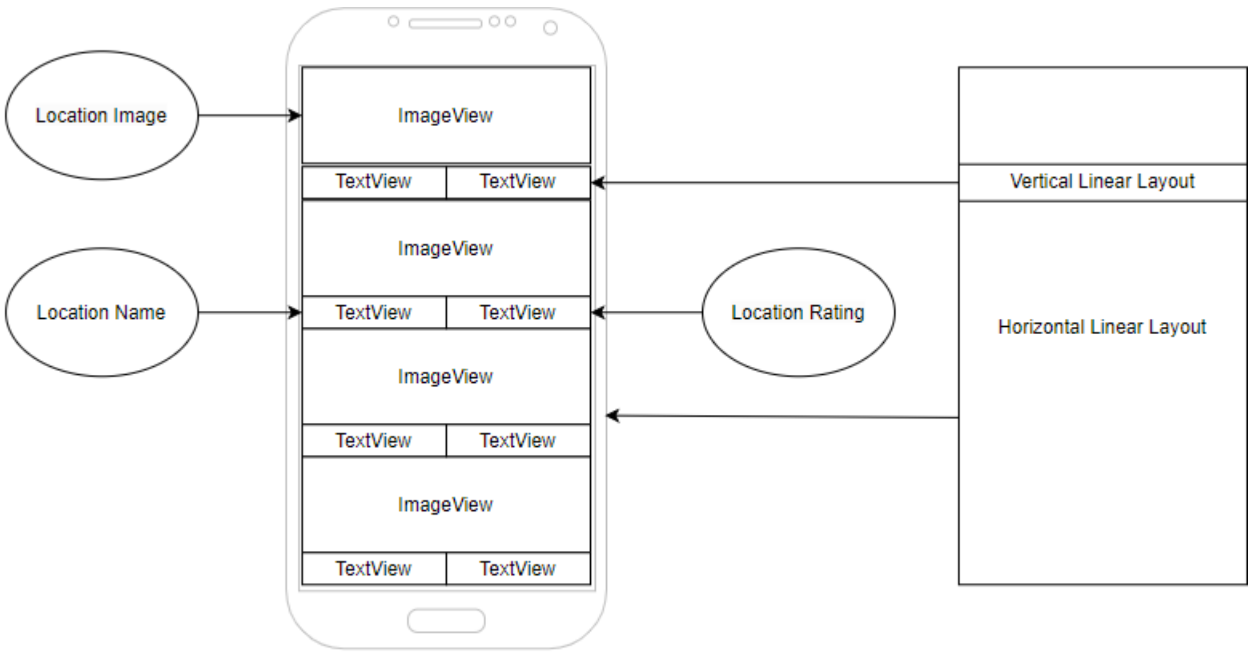
Including sophisticated Kotlin features is a crucial component of this main work. Scope functions were one of the features I used to add Location objects to Intents. The code sample following provides an illustration of this implementation:

```
val gardenImage = findViewById<ImageView>(R.id.garden)
        gardenImage.setOnClickListener {
            val i = Intent(this, DetailActivity::class.java).apply {
                putExtra("image", garden)
            }
            startActivity(i)
        }
```

This is faster than conventional coding using lazy codes and calling an object/variable multiple times.

Gap 7: Sketches of additional screen layouts

I utilized the Constraint Layout in the submission; here are some examples of other possible layouts.
(Please read the following two pages.)



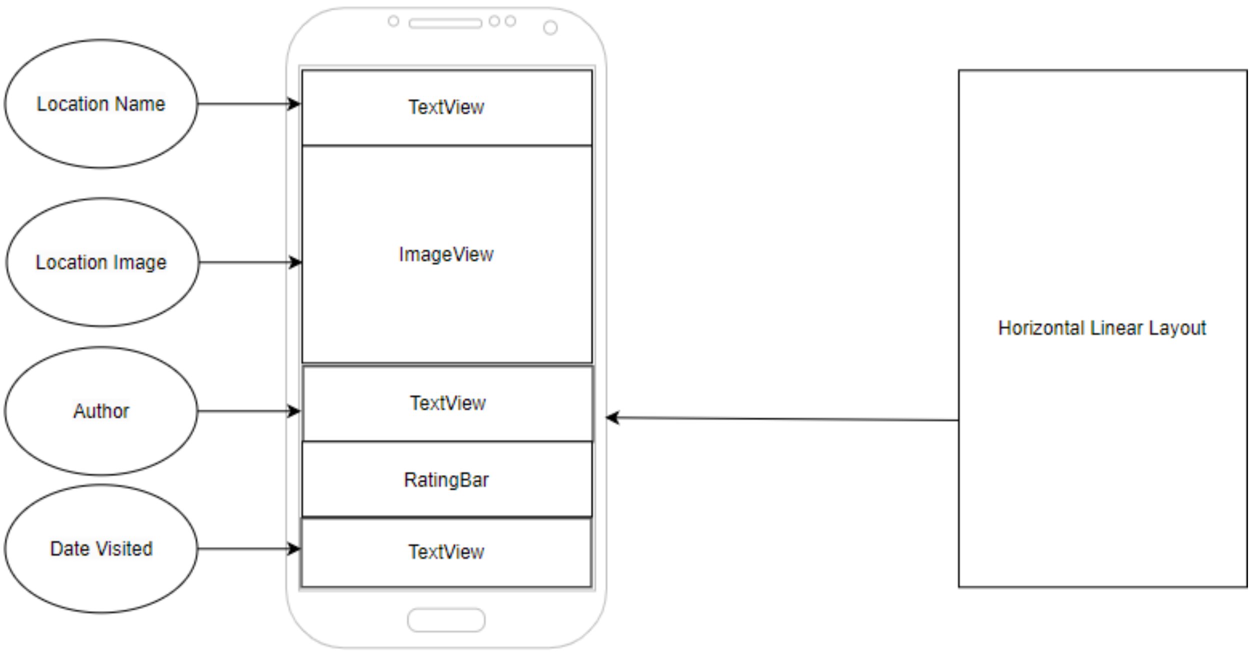


Figure 6: Linear Layout

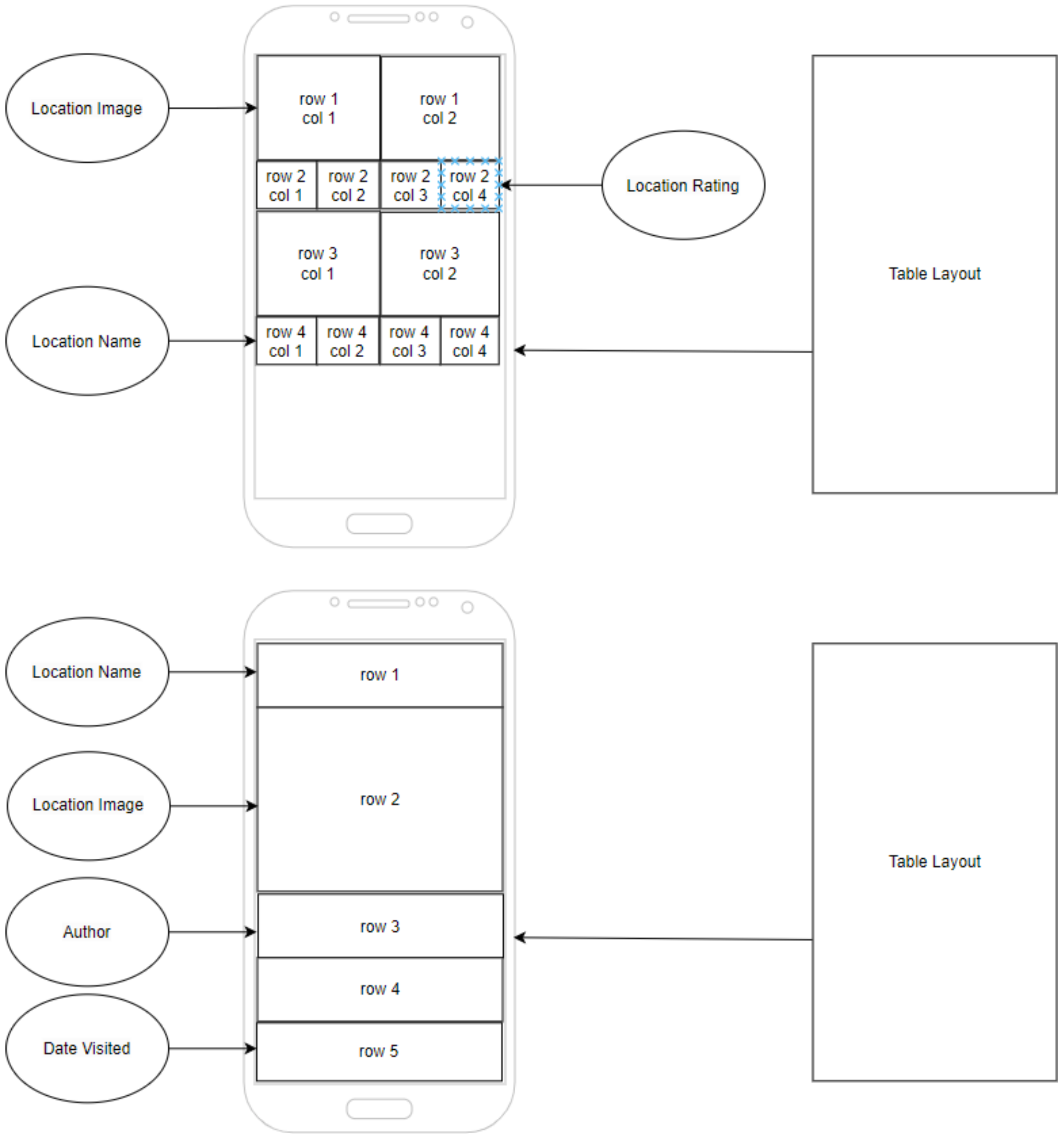
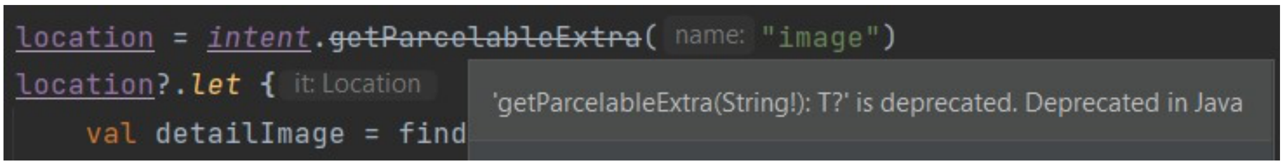


Figure 7: Table Layout

Open Issues and Recommendations

The deprecated getParcelableExtra() was the main problem, and I haven't been able to find a workaround for it. Although I haven't been successful yet, I think decreasing the API level can help. I'm eager to hear back from this core contribution so I can fix this issue.



```
location = intent.~~getParcelableExtra~~( name: "image")
location?.let { it: Location
    val detailImage = find
```

'getParcelableExtra(String!): T?' is deprecated. Deprecated in Java