Task Core 2 – Spike: Communication between activities

Link to GitHub repository: https://github.com/SoftDevMobDevJan2023/core2-103488117.git

Goals:

- Provide evidence of the ability to create an app that contains multiple activities.
- Send data bidirectionally between activities with the help of intents.
- Use photos and other addon resources.
- Implement more advanced UI elements than the previous core task.
- Use Kotlin's advanced features and code snippets.

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

In this assignment, I have used a new UI widget, which is the rating bar. The rating bar can get input from the user through touching or dragging. New elements including the selected section and non-selected section colours can be used to apply new styles, which will be mentioned in one of the next sections.

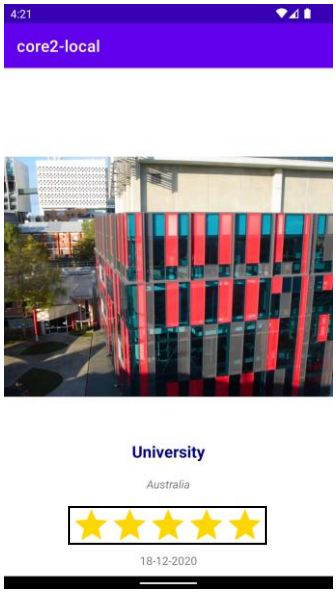

Figure 1: Rating bar highlighted in the squared box

Gap 2: Using available and addon resources

The assignment requires using additional images, which we can add to the `/app/res/drawable` folder. I found dragging the images directly to the UI of Android Studio was the fastest way, in addition to left-clicking on the `drawable` folder. Please note that you can also use `android:scaleType="centerCrop"` to scale the image to the `ImageView` and make it look better.

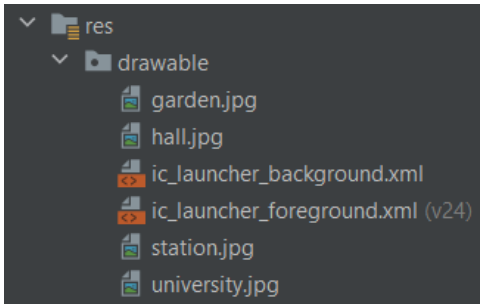

Figure 2: File structure

Gap 3: Implementation of intents

Intents are used to pass data from one activity to another, and together with Parcelable make the program efficient as it stores the data on the disk instead of memory (RAM).

Explanation of my intent:

- First, I created a Parcelable class named Location and assign necessary attributes like image, name, location, date, and rating (date was to parse the Drawable image and rating had to be float as the rating bar also accepted floating point ratings).
- Second, I created an intent, which is a local variable in the setOnClickListener event for every image that can be clicked on, of which there were 4. Then, I parse in the puExtra function 2 parameters: the key name and the nullable objects which were declared in the MainActivity class and then invoked in the onCreate() override function.
- Third, I created another nullable Location object in the DetailActivity class, which received the parse-in data, assign them to the local Location object in DetailActivity and displayed them on the screen when this activity is created.

```
// 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 styles which I applied to the program lay in the cover and header text which displayed the name of the location as well as the rating bar. The text colour was changed to navy blue and the stars on the rating bar in the detail activity were changed from light green to gold yellow, which may seem familiar to the user. To do this, we need to add a new styles.xml file to the /app/res/value folder, define our themes and assign them to the elements in the layout files.

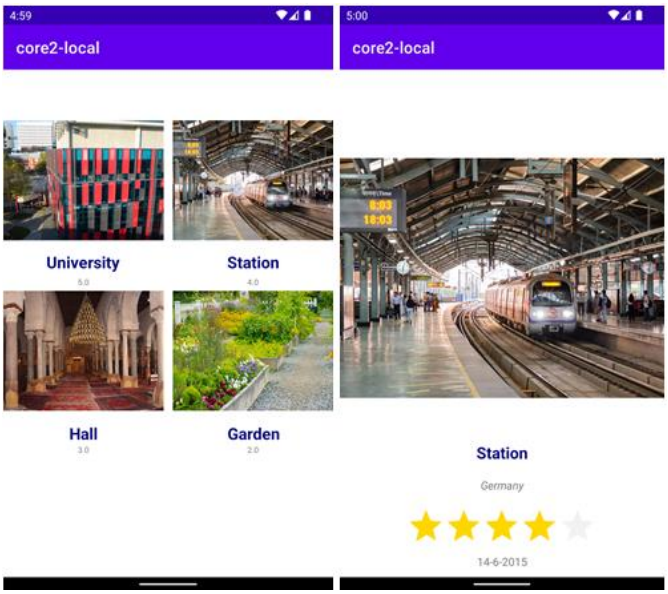

Figure 3: Blue navy text colour for headers, gold yellow for the stars on the Rating bar

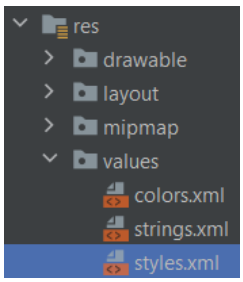

Figure 4: styles.xml added to the folder structure

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

For this assignment, I have used Logcat and the debug console to log and filter necessary error messages to fix and enhance the program.

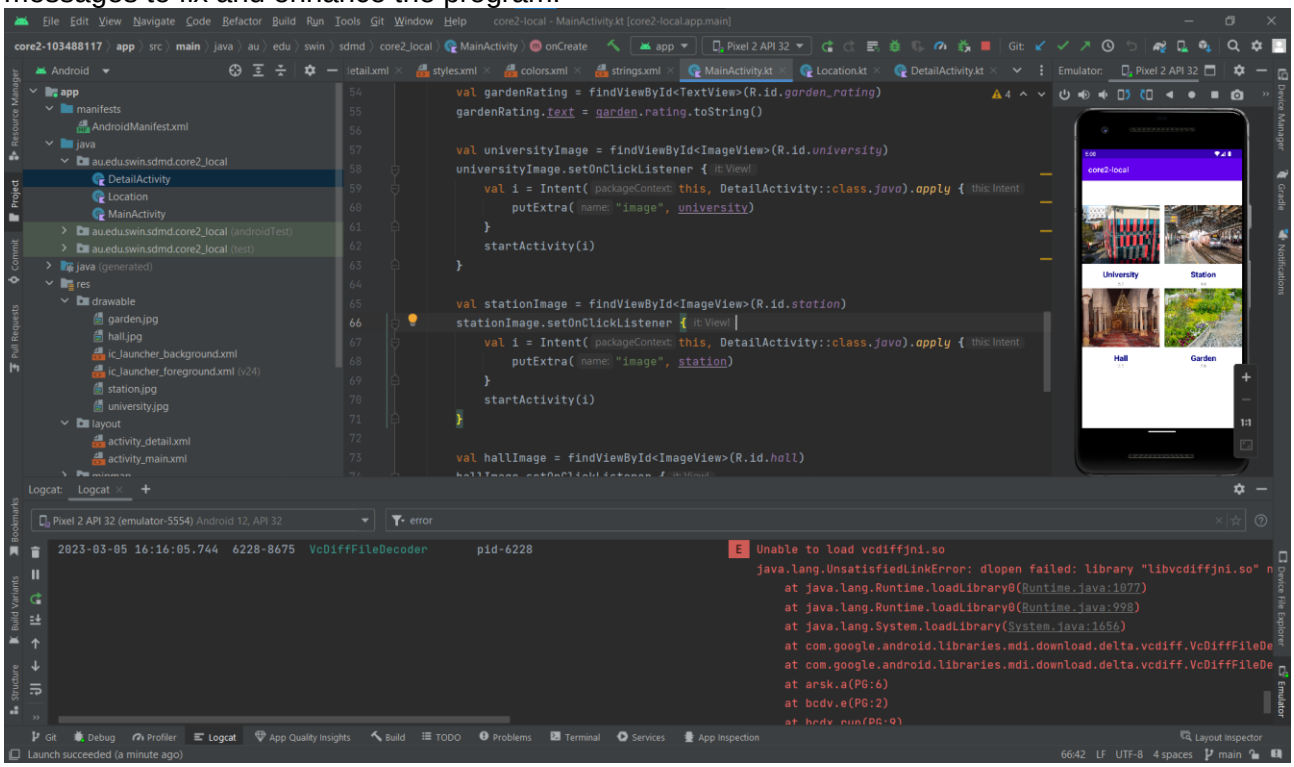

Figure 5: Logcat and debugging

Gap 6: Kotlin advanced features

One of the requirements of this core task is to use advanced features available in Kotlin. One of which I used was scope functions to put the Location objects into Intents, the example is shown in the code snippet below:

```
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

In the submission, I used Constraint Layout, here are some sketches for other layouts that can be implemented.
(Please view the next 2 pages)

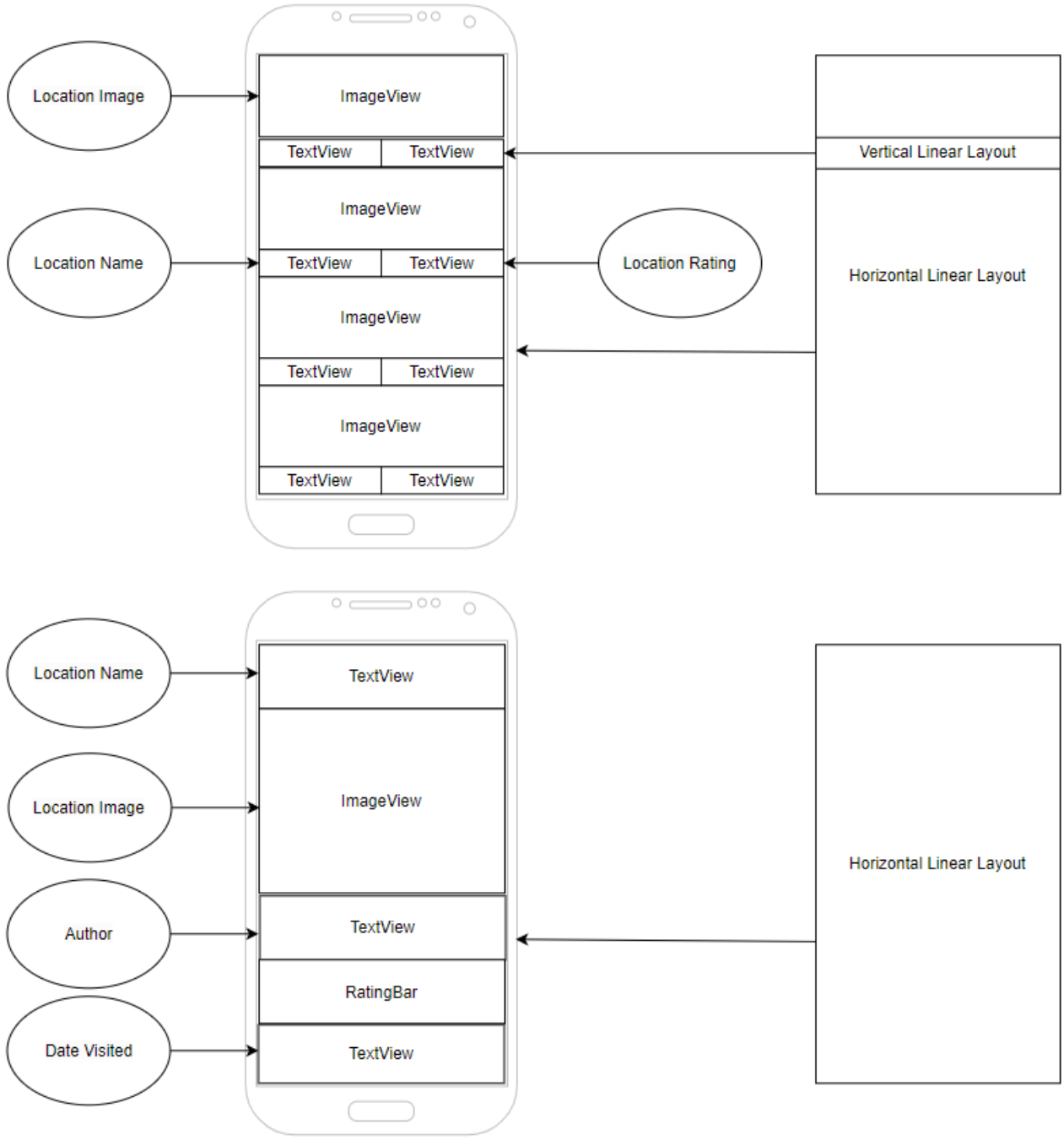

Figure 6: Linear Layout

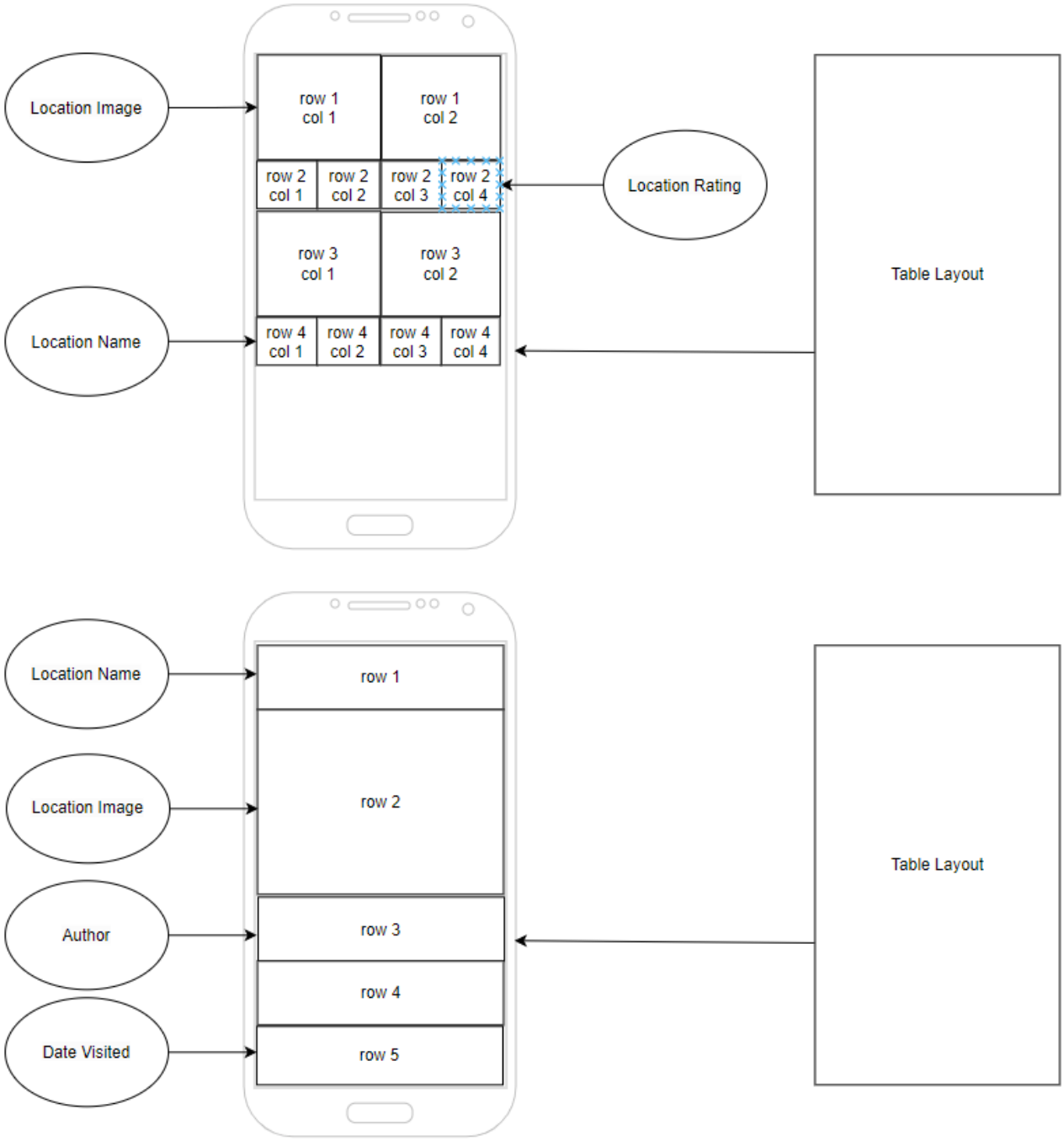

Figure 7: Table Layout

Open Issues and Recommendations

The only existing issue was that getParcelableExtra() is deprecated, which I have not found an alternative solution to. I believe changing the API level (lowering it) can help, but have not yet succeeded and I am looking forward to receiving the feedback from this core submission to patch this problem.