P/C Task ## – Spike: Pass task 2A – Spike report

Two Sketches:

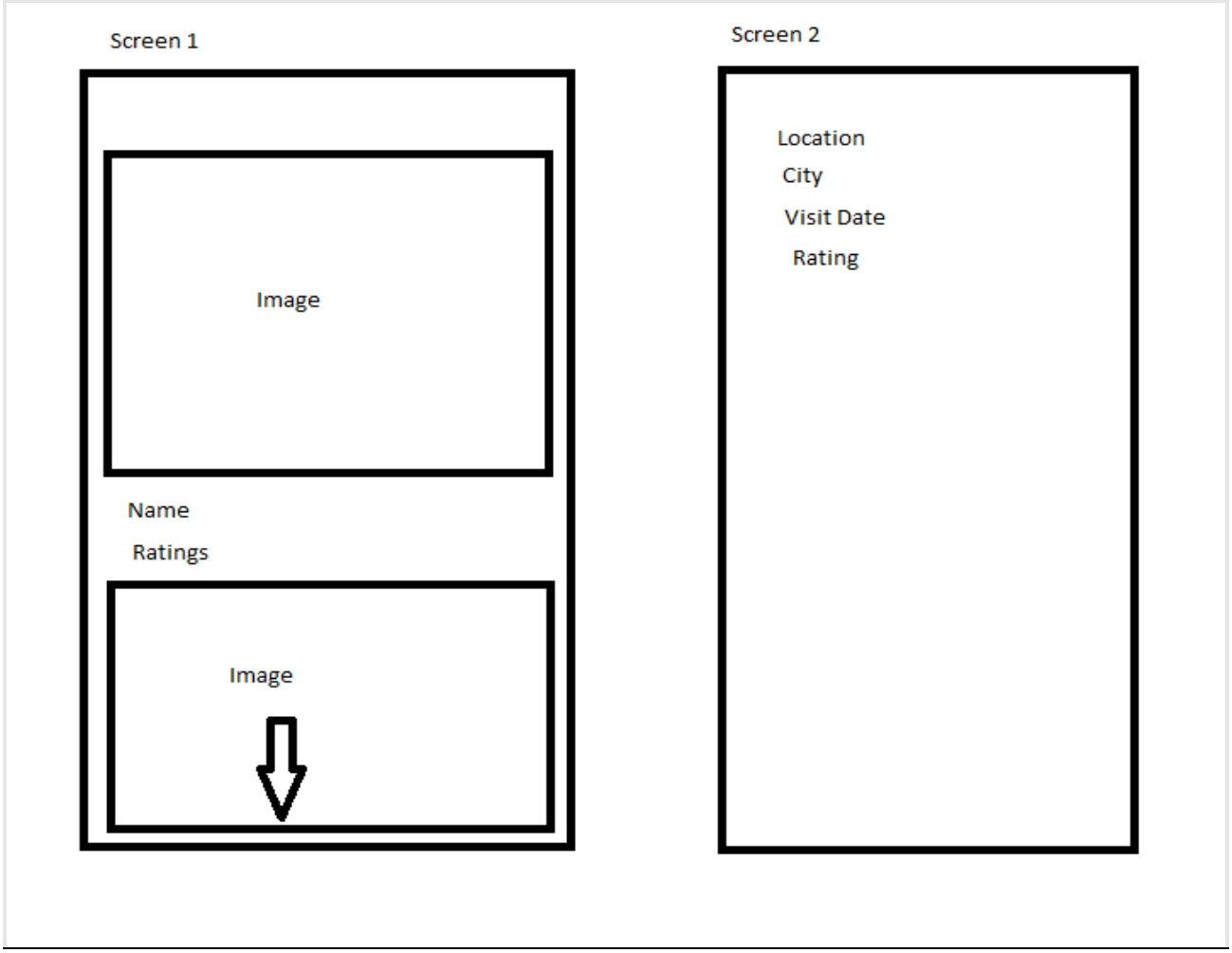

Figure 1 Sketches Done in paint

Tools and Resources Used

- Android Studio

Intent Components and Parcelable Objects

In our Android app development, we've utilized Intents along with the Parcelable interface to effectively manage and transfer data across activities. An Intent in Android serves as a communication tool that allows you to request actions from other app components. Specifically, we've employed explicit Intents to launch our DetailActivity. These Intents are set up with extra data that specifies the image details to be displayed, streamlining the communication between the MainActivity and DetailActivity.
The data passed through the Intent is bundled in the ImageDetails data class, which is marked with @Parcelize. This class adheres to the Parcelable interface, essential for transmitting complex data between activities. The Parcelable interface is favored over Serializable in Android because of its performance benefits. It is designed to parse data quickly, which is crucial for high-performance applications requiring rapid access and data retrieval. By adopting Parcelable, we've managed to cut down on runtime overhead significantly compared to Serializable, since Parcelable is specifically optimized for Android's native inter-

process communication and is fine-tuned for sending and receiving data packets. This approach not only ensures efficient data handling between activities but also boosts the overall performance of the app, leading to a smoother user experience.

Knowledge Gaps and Solutions

This section presents the listed knowledge gaps and their solutions with supporting images, screenshots and captions where appropriate/required.

Gap 1: Implementing Parcelable for Efficient Data Transfer

Gap: Understanding and implementing Parcelable efficiently can be complex due to its manual coding requirements, especially when dealing with multiple data types.

Solution:
Use the @Parcelize annotation which simplifies the implementation. The ImageDetails class is annotated with @Parcelize, automating the parcelable process without explicitly defining the writeToParcel and createFromParcel methods. This ensures efficient data passing with minimal boilerplate code.

```
@Parcelize
data class ImageDetails(
    val name: String,
    val location: String,
    val dateOfVisit: String,
    val rating: Float
) : Parcelable
```

Figure 2 Using @Parcelize to simplify Parcelable implementation.

Gap 2: Handling Click Events to Trigger Data Transfer

Gap: Developers new to Android may struggle with setting up click listeners on dynamically created UI elements, such as images in a scrollable list.

Solution:
In MainActivity, set up click listeners within a loop that iterates through a list of images. Each listener triggers an Intent that passes the ImageDetails object to DetailActivity. This approach demonstrates handling dynamic content and intents effectively.

```
imageDetails.forEachIndexed { index, details ->
    val imageView = findViewById<ImageView>(resources.getIdentifier( name: "imageView$index",  defType: "id", packageName))
    imageView.setImageResource(resources.getIdentifier( name: "image$index",  defType: "drawable", packageName))
    imageView.setOnClickListener { it: View!
        val intent = Intent( packageContext: this, DetailActivity::class.java).apply { this: Intent
            putExtra( name: "DETAILS", details)
        }
        startActivity(intent)
    }
```

Figure 3 Setting up click listeners on images to pass data with Intent.

Gap 3: Displaying Data Using a Parcelable Object in a New Activity

Gap: Displaying transferred data in a new activity can be confusing, particularly when it involves custom objects.

Solution:
In DetailActivity, retrieve the Parcelable object from the Intent and use it to update the UI. This ensures that the data passed from MainActivity is displayed properly.

```
class DetailActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_detail)

        val details = intent.getParcelableExtra<ImageDetails>( name: "DETAILS")
        val detailTextView = findViewById<TextView>(R.id.detailTextView)
        detailTextView.text = "Location: ${details?.name}\nCity: ${details?.location}\nVisit Date: ${details?.dateOfVisit}\nRatir
    }
}
```

Figure 4 Retrieving and displaying Parcelable data in DetailActivity.

Open Issues and Recommendations

Issue 1: Scalability of UI Layouts
Our current implementation uses a static, linear layout which may not scale well with a larger number of images or varying screen sizes. **Recommendation:** Transition to a **RecyclerView** with a **GridLayoutManager** for better performance and scalability.

Issue 2: Memory Management
Using high-resolution images directly in the app could lead to excessive memory usage and potential crashes. **Recommendation:** Implement image loading libraries like Glide or Picasso that handle image resizing and caching efficiently.

Issue 3: Parcelable Efficiency
Although **Parcelable** is efficient, manually managing parcelable data can be error-prone. **Recommendation:** Consider using a library like AutoValue with its Parcelable extension to automate and error-proof Parcelable implementations.

.

Resources

- Android Developers Documentation on Parcelable: This official guide provides a comprehensive overview of implementing Parcelable in Android, including examples and best practices.URL: https://developer.android.com/reference/android/os/Parcelable
- Android Click Events Tutorial: This tutorial from Vogella provides an in-depth look at handling click events in Android, covering various types of click listeners and their applications in real-world scenarios. URL: http://www.vogella.com/tutorials/AndroidEventHandling/article.html
- Android Intents and Data Passing: This section of the Android Developers documentation offers detailed information on how to use Intents for data passing between activities, specifically focusing on the use of Parcelable data. URL: https://developer.android.com/guide/components/intents-filters