**Repository:**

**Own repository which has no emulator running issues and pushing issues**

[**https://github.com/Dilni-DS113/DS311-DilniCore2**](https://github.com/Dilni-DS113/DS311-DilniCore2)

**Classroom link respiratory which doesn't run correctly in an emulator**

[**https://github.com/SDMD-2022/core-2-Dilni-DS113**](https://github.com/SDMD-2022/core-2-Dilni-DS113)

**Goal:**

The goal of the development of this application is to create a travel journal application that uses two activities and passes data between those two activities using a intents parseable object.

Resources: <https://www.geeksforgeeks.org/how-to-send-image-file-from-one-activity-to-another-activity/>

Skill Gap 2 → *How to import images and use them in application*

<https://developer.android.com/studio/write/resource-manager#:~:text=To%20import%20image%20resources%20into,that%20you%20want%20to%20import>.

Skill Gap 4 → *How to edit colours in the application in the themes XML file and where to find colour styles and themes*

<https://developer.android.com/develop/ui/views/theming/themes>

Skill Gap 4 → *How to edit colours in the application in the themes XML file and where to find colour styles and themes*

<https://developer.android.com/develop/ui/views/text-and-emoji/fonts-in-xml>

<https://material.io/resources/color/#!/?view.left=0&view.right=0&primary.color=64B5F6&secondary.color=BBDEFB&primary.text.color=263238&secondary.text.color=263238>

Skill Gap 4 → *Enable the addition of new font and style the font*

<https://developer.android.com/develop/ui/views/text-and-emoji/fonts-in-xml>

### Skills gaps and solutions:

1. Use a broader range of UI elements

UI elements are the components of an android app that have specific functionality. In this travel journal app.

Solution:

I used multiple different types of UI elements: a rating bar, text view, a toggle button and image views.

* Be able to add an image view from the widget section of the palette to both detail and main activities.
* Be able to add a rating bar from the widget section of the palette into the detailed activity.
* Be able to add text views from the text section from the palette to both detailed and main activities.
* Be able to add a toggle button from the button section of the palette.

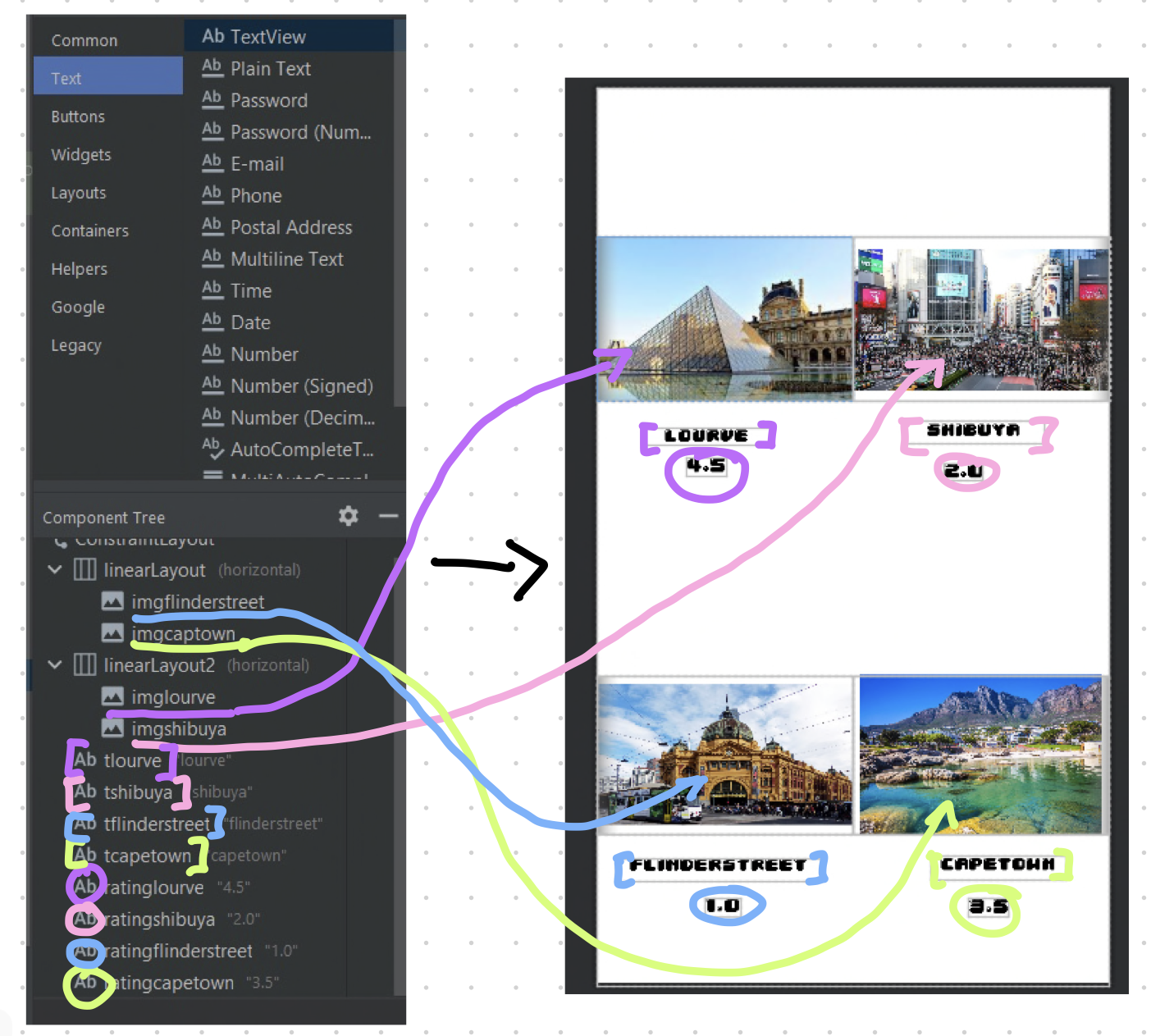
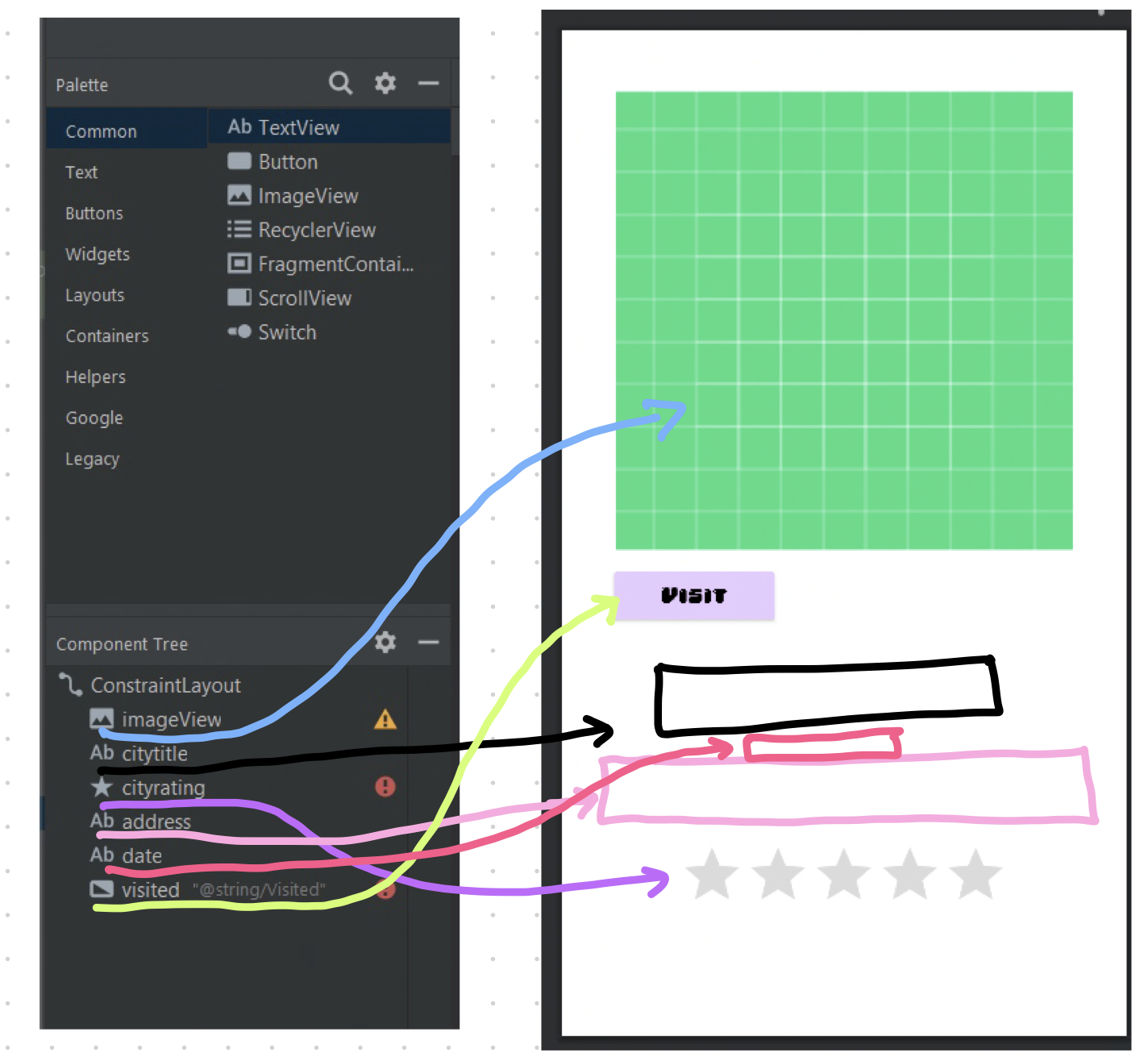


Figure 1 : Activity\_locationdetail

Figure 2: Activity\_main

1. Work with resources

Import resource files into correct folders hence the application can access them and use them for certain functionality and features such as adding image resources to image views.

Solution:

*Reference: How to import images and use them in application*

<https://developer.android.com/studio/write/resource-manager#:~:text=To%20import%20image%20resources%20into,that%20you%20want%20to%20import>.

* Be able to select and download 4 images from google.
* Be able to save the 4 images into the android app folder.
* Be able to navigate to view on the top navigation bar in android studio.
* Be able to open the tool window from the view menu option.
* Be able to select and open resource manager from the tool window
* Be able to import images but add them by pressing the + button and pressing import drawable from the drop-down menu which will open the app files where a developer be able to access downloaded images and add them to the drawable files.
* Be able to add image resources to the image view from the imported resources.

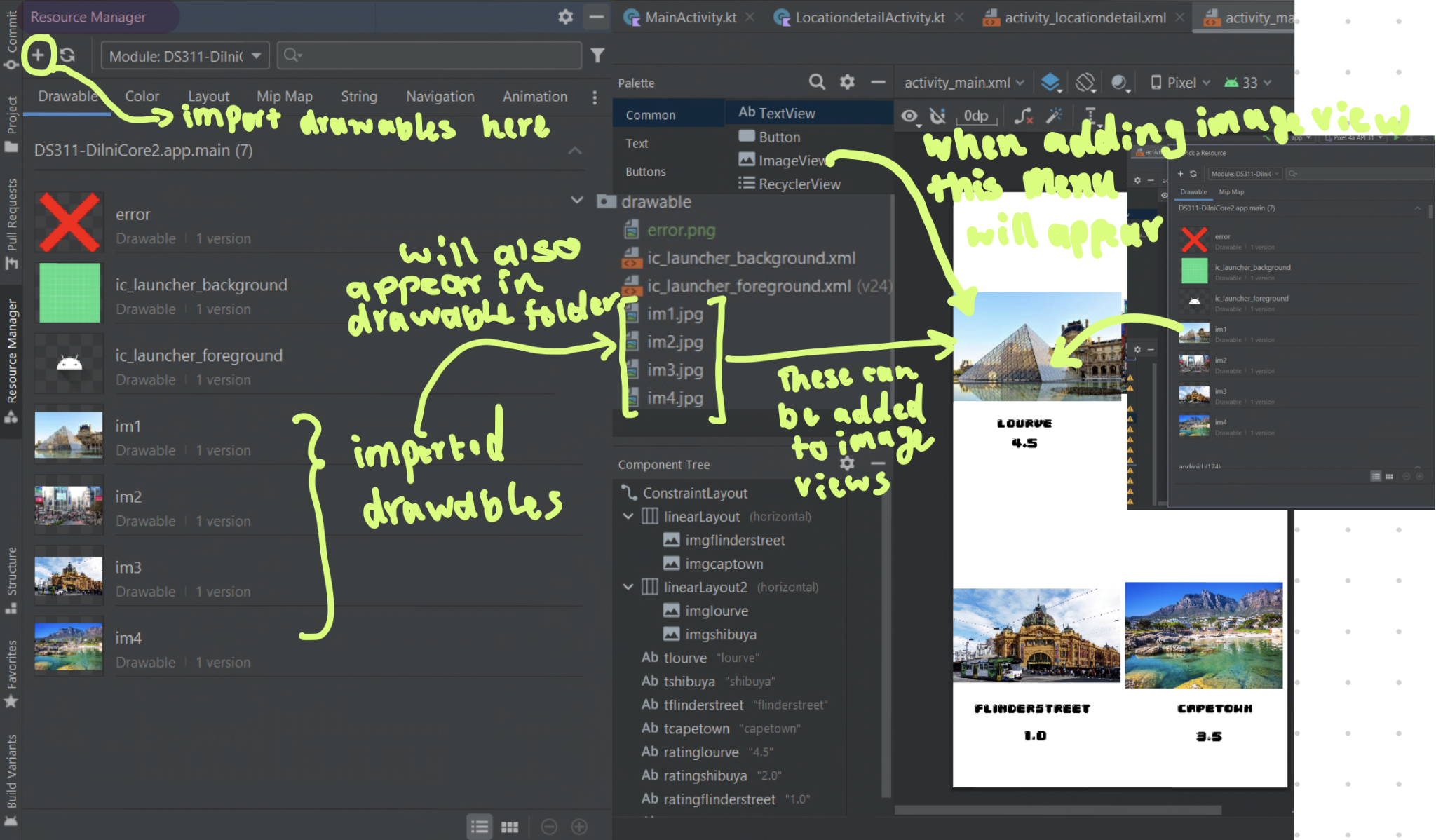


Figure 3: Demonstrating the process of importing and adding image resource

1. Correctly implement intents

This application requires the inclusion of intents. This enables the opening of detailed activity from the main activity in addition to the transfer of data between activities which can be displayed in the detailed activity. Furthermore, a practicable data class was added

So that the intent can pass multiple pieces of data using a parciable object.

Solution:

* Be able to create a new second detailed activity by right-clicking the com.example.ds311\_dilnicore2 and clicking New → Activity → Empty activity.
* Be able to add the correct view and layouts to the detailed activity so data from the main activity can be correctly placed into the detailed activity.
* Be able to create a practicable data class and create variables for certain data to pass from a parciable object that contains multiple instances of data.
* Be able to initialise any practicable objects with data into variables which will allow data to be passed when an image view is clicked.
* Be able to create a click listener for the images.
* Be able to add intents to each click listeners which allow users to open respective travel journal locations with detailed clicks on images and transport data into specified views such as text view and rating bar widgets.
* Be able to program detailed activity to create a practicable object which will transfer all specified data points to the selected view in the detailed activity.

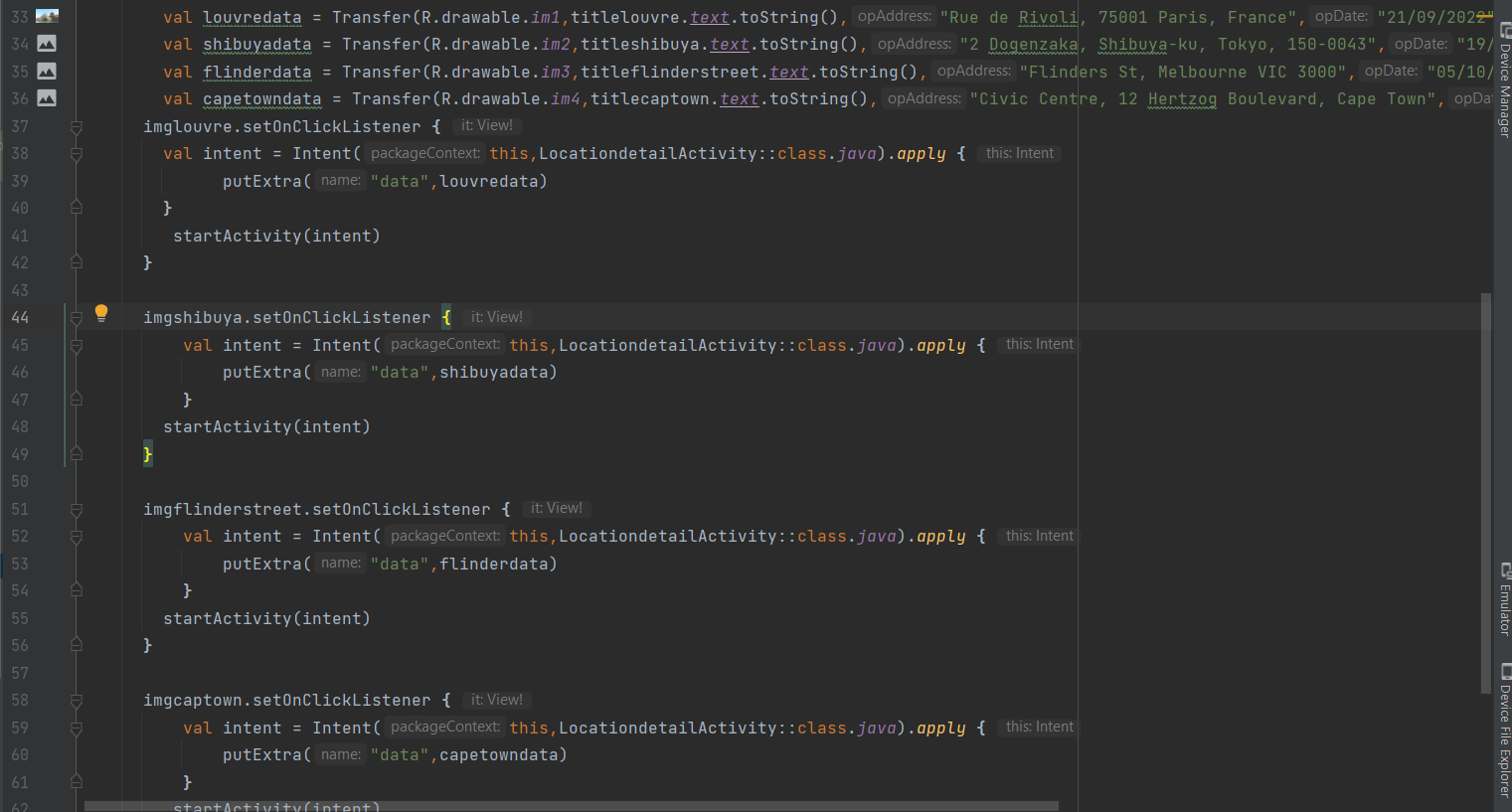


Figure 4: Intents and parciable object initialisation in the main activity



Figure 5: Creating the find views for detail activity so data from parciable object can be passed somewhere

1. Use styles in an app

The design can be modified programmatically in the app's code by downloading certain resources which can be enabled in the app XML code. This application has two styles added such as the detailed and main activity. Such as the fonts and the colour of certain attributes such as buttons or the background and features.

Solution:

*Reference: How to edit colours in the application in the themes XML file and where to find colour styles and themes*

[*https://developer.android.com/develop/ui/views/theming/themes*](https://developer.android.com/develop/ui/views/theming/themes)

<https://material.io/resources/color/#!/?view.left=0&view.right=0&primary.color=64B5F6&secondary.color=BBDEFB&primary.text.color=263238&secondary.text.color=263238>

Colour:

* Be able to find a material design website that gives a palette of colours for parts of the application layout.
* Be able to select preferred primary and secondary colours from the colour palette.
* Be able to export the android XML file.
* Be able to open the XML files and copy all colour elements labelled anything with secondary or primary into the colour XML file in the values folder in application files.
* Be able to change which colours appear in the application by changing the colour application name in the themes XML file to new copied colour labels.

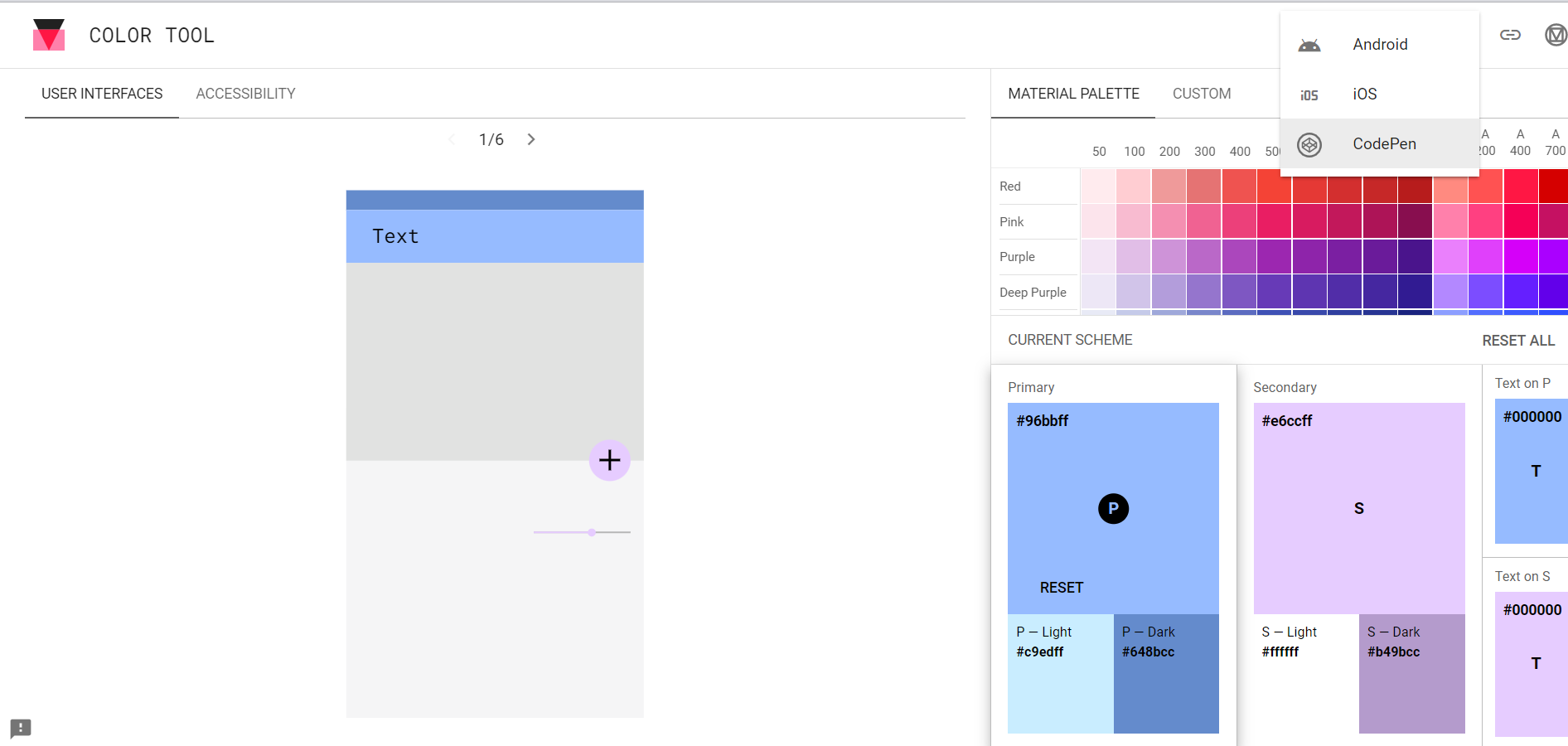


Figure 6: The website where the XML colour elements will be chosen and exported



Figure 7: Colour XML file which has new copied colour elements.

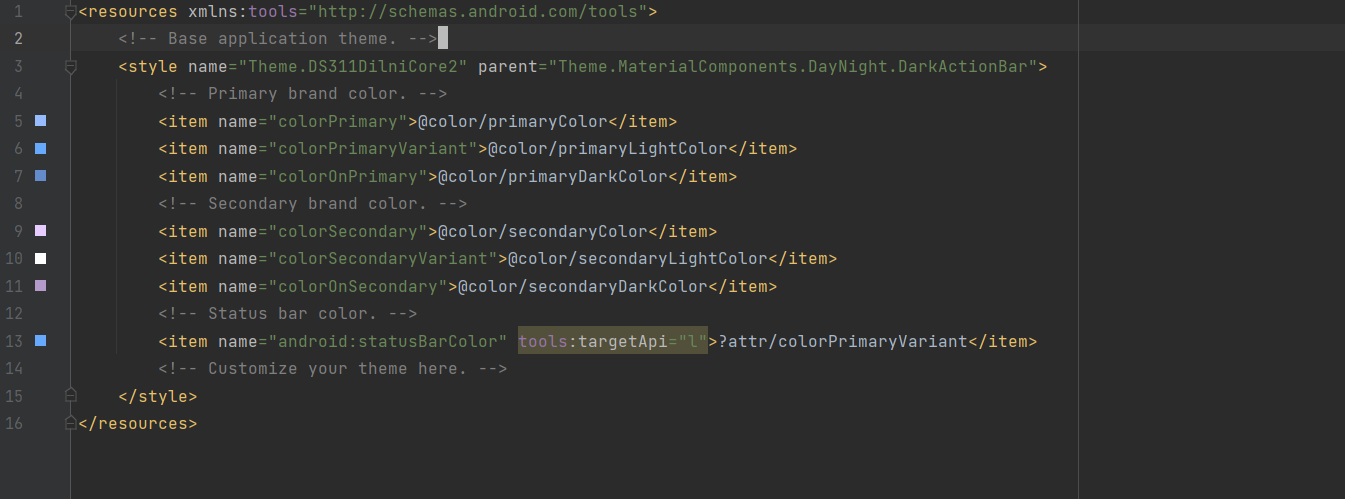


Figure 8: Themes files that have the changed primary and secondary colour items.

*Reference: Enable the addition of new font and style the font*

<https://developer.android.com/develop/ui/views/text-and-emoji/fonts-in-xml>

Fonts:

* Be able to download a font file.
* Be able to create a font folder
* Be able to add font file into the fount folder
* Be able to create a font family file called text-style to set certain specifications for text such as setting the colour, size and font style of the text (remember to set the text to the downloadable font.)
* Be able to add a text-style file into the XML code of each layout file text view.
* Be able to change the colour of any fonts in the XML code of each layout if not done in the font family.

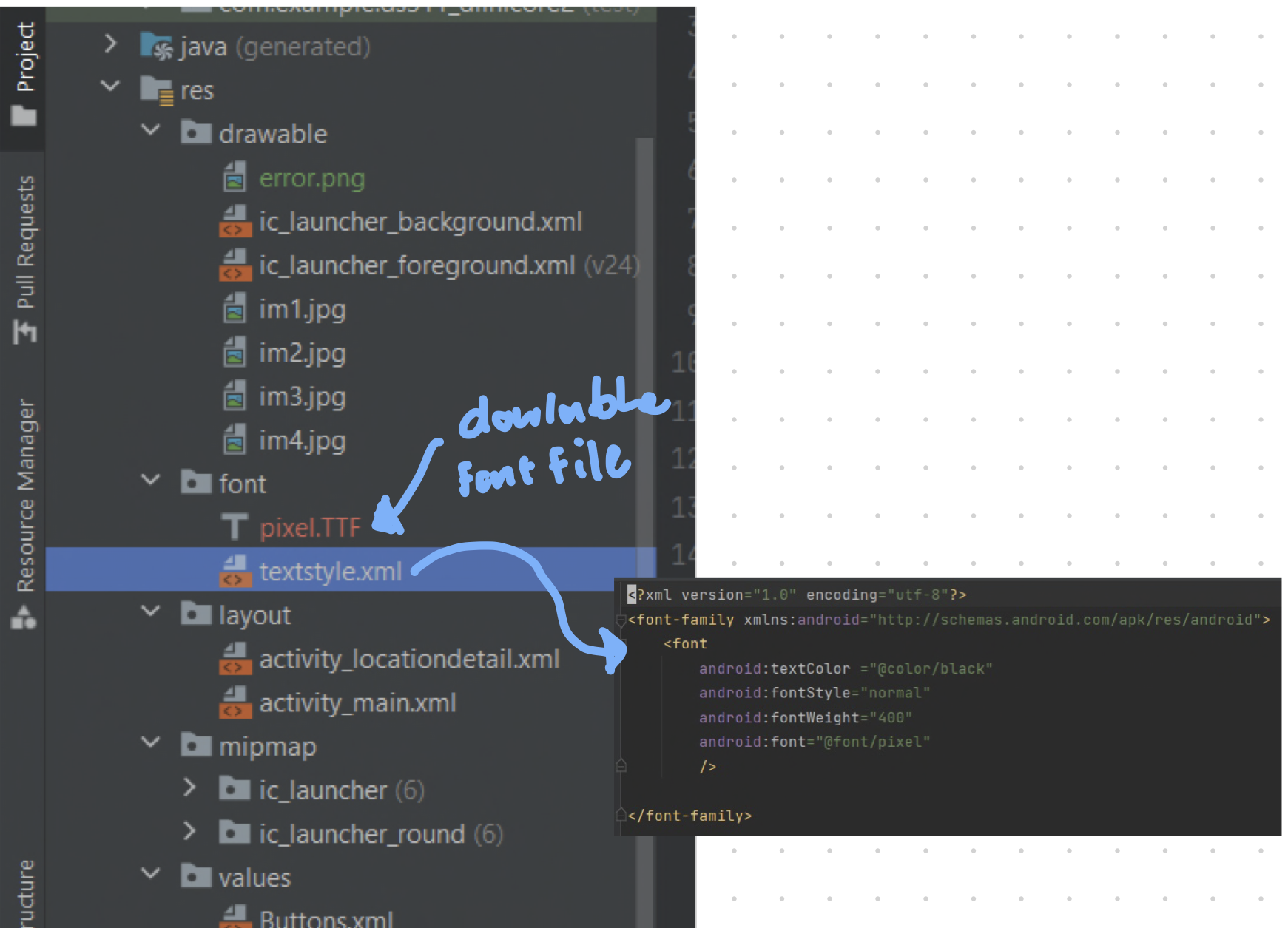


Figure 9: Font folder and font file importing and creating a font family file where text can be spesifction can be changed.



Figure 10: Demonstrates the change in font of text to downloaded pixel font

1. Command of IDE

IDE build displays the application as running on a mobile phone, where a developer can view and test out the functionality of the application.

Solution:

* Be able to select a device from the device manager and run the device.
* Be able to view the device on the emulator tab.
* Be able to power up the emulator.
* Be able to press the green arrow button to run the application.
* Be able to use and view applications on the emulator.

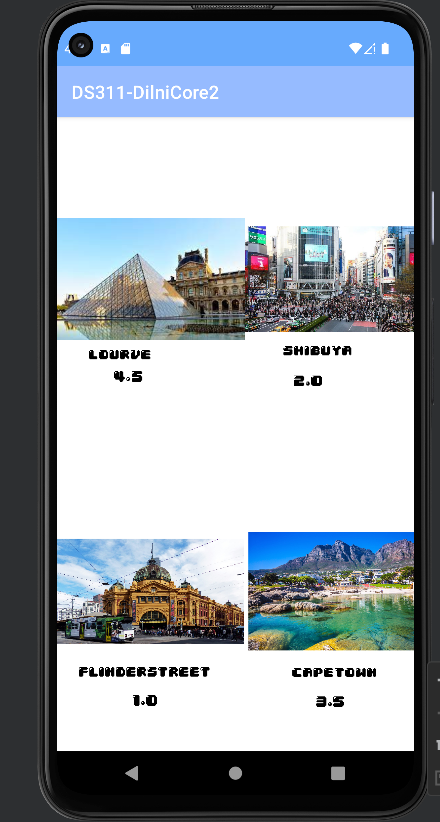


Figure 11: Main activity running



Figure 12: detailed activity running

1. Essential Kotlin

Kotlin is a programming language that the application will program in where all functionality will be executed, hence the code shown should demonstrate good Kotlin syntax.

* Be able to use certain statements such as find view model and access a view in the Kotlin version.
* Be able to initialise internets in the specified Kotlin version.
* Be able to create a Kotlin data class for the Parciable object.

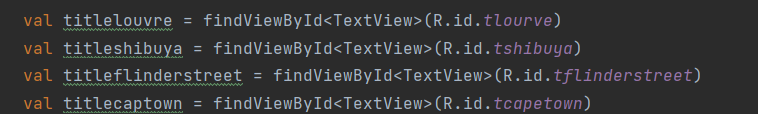


Figure 13: Variables to initialise views

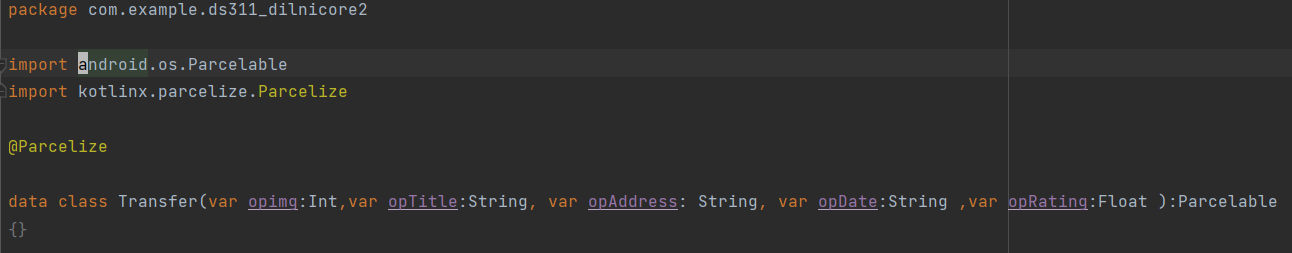


Figure 14: Parcelable Kotlin data class

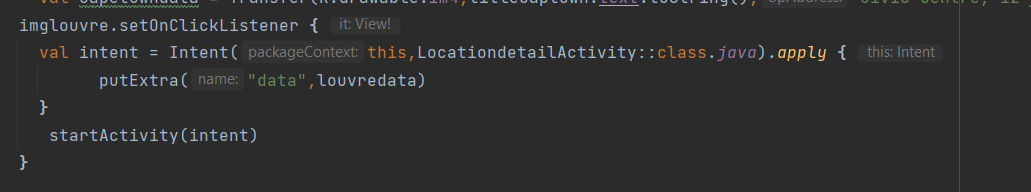


Figure 15: starting an Intent using a click listener

1. Sketch screen layouts (A larger certain of the sketch will be included at the end of the report in the appendix )

Sketching the application will give a design and an understanding of how this application will look giving a simple design of each activity layout.

Solutions:

* Be able to open a drawing application on a computer or tablet or even a physical sketchbook.
* Be able to sketch out how each layout will appear.
* Be able to add correct UX fundamentals to sketch such as adding representation of text views, toggle buttons, and rating bar.
* Be able to add labels to sketch all attributes and views.

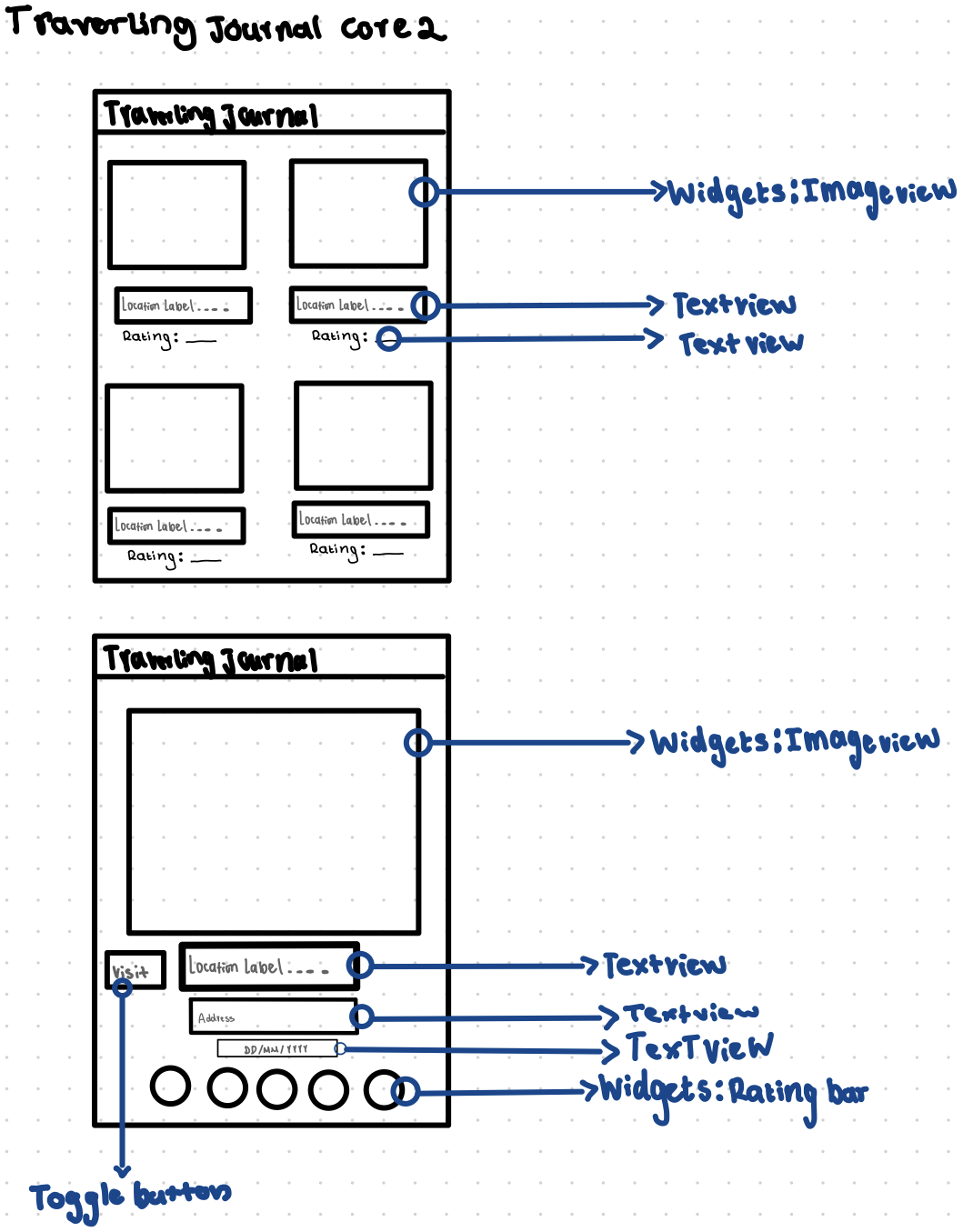


Figure 16: Sketch of layouts

1. Implement simple screens

Simple screens are screen layouts that are created from the sketches of the screen. Which take reference from the designs.

* Be able to refer to the screen sketches and copy the same views.
* Be able to add liner and constraint layouts to add different views to them.
* Be able to constrain all views to the correct position.

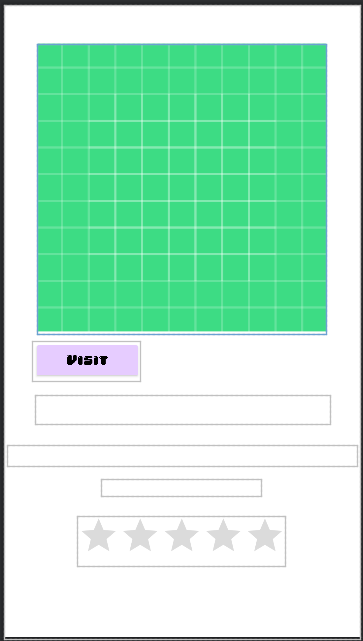
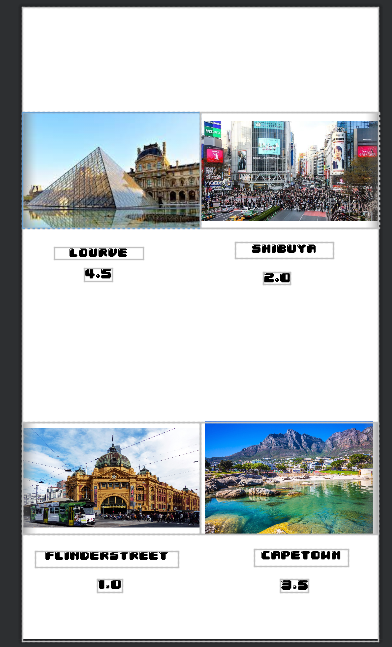


Figure 17 & 18: simples layouts