**Assignments: Core 2 Extention Spike report**

## **Dilni De Silva 103616345**

### Repository:

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

<https://github.com/Dilni-DS113/DS311-DilniCore2/tree/Core_2_Extention>

**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:

To further develop the travel journal application where certain views in the detail view can be edited, thus the data can be passed from detailed activity to main activity.

### Resources:

Skill gap 1 → How to validate an edit input

<https://www.geeksforgeeks.org/implement-form-validation-error-to-edittext-in-android/>

Skill gap 3 → Implementation of toasts

<https://developer.android.com/guide/topics/ui/notifiers/toasts>

### **Skills gaps and solutions:**

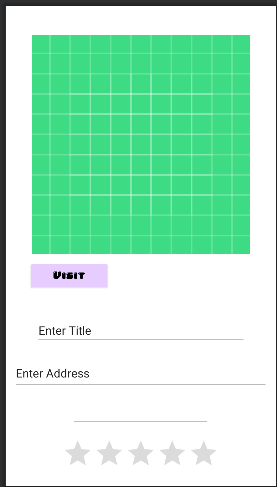
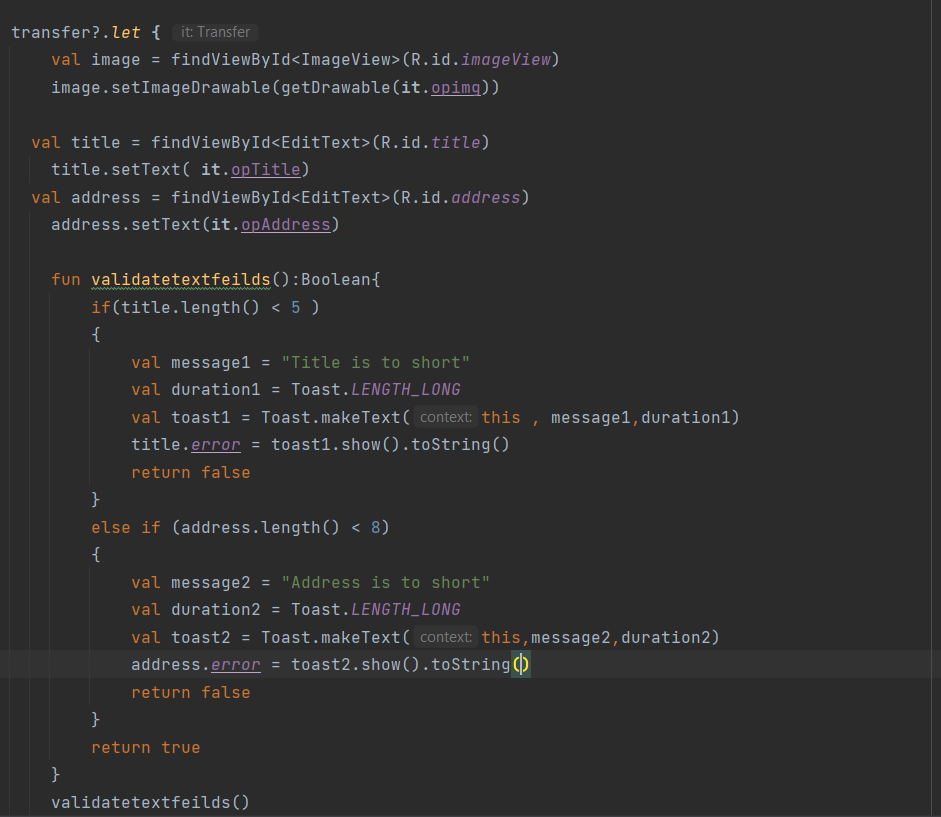
1. **Implement validation for text input:**

Validation for text input of users can have certain conditions set which have to be fulfiled to be able to submit the text.

* Be able to replace all text views with edited text views in the detailed layout
* Be able to create variables in Kotlin of edit text views.
* Be able to create a validation method.
* Be able to create a conditional *if else* statement in the validation method.

***References: How to validate an edit input***

<https://www.geeksforgeeks.org/implement-form-validation-error-to-edittext-in-android/>



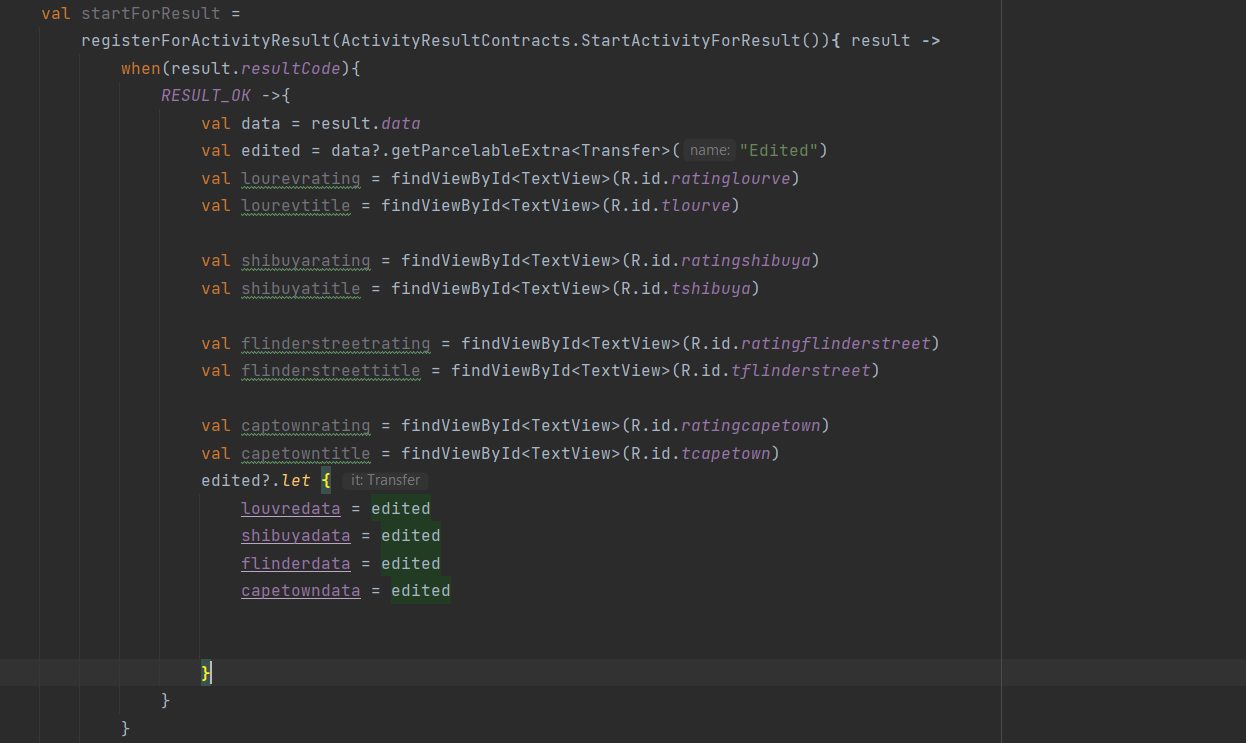
*Figure 1 & Figure 2: Kotlin validation method and new edit text variables and views from detail activity layout.*

1. **Activity Results contract**

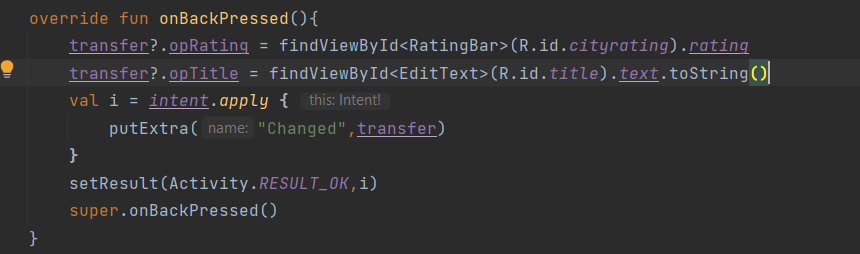
The activity results contract allows data from the detailed activity to be passed to the main activity when the back button is pressed.

Solution:

* Be able to an on-back pressed function.
* Be able to Initialise any data to be transferred by the parseable object.
* Be able to create an intent that transfers the data between classes.
* Be able to code in the main activity the start result function
* Be able to create all appropriate views in the start result function
* Be able to create a let function that updates the parcelable objects.



*Figure 3: Start for result function in the main activity*



*Figure 4: On back pressed method*

1. **Toast and snack bars**

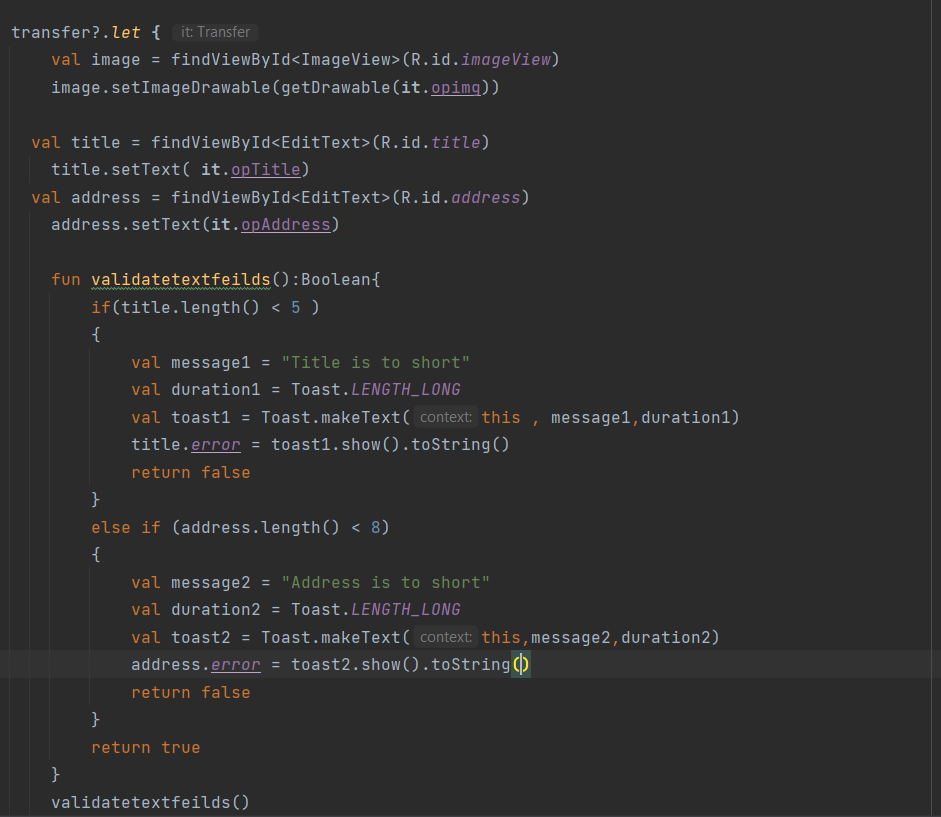
***References: How to implement toasts***

<https://developer.android.com/guide/topics/ui/notifiers/toasts>

Toasts are pop-up message which gives feedback to the user when a certain task is completed. Here I have implemented a toast as part of the validation when the title of a location is not long enough a toast message will appear for a long time.

Solution:

* Be able to create a message variable that holds a string (Message for toast)
* Be able to set a duration for the toast in a duration variable
* Be able to call the make toast function and use previously created message, and duration variables as the parameters.
* Be able to set toast when an error occurs in the view, using the show().

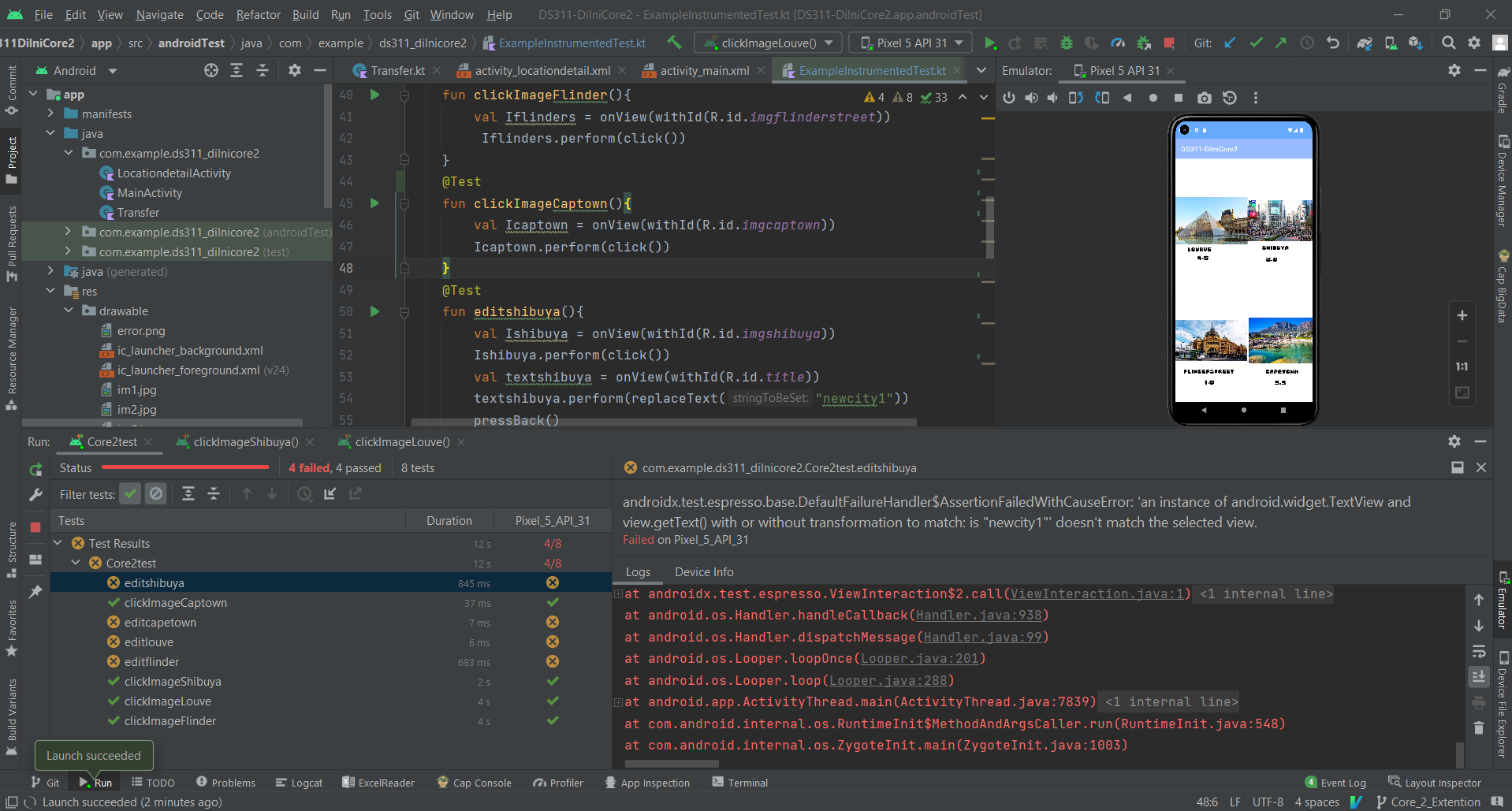


*Figure 5: Toast used in the application*

1. **Unit and UI tests**

To test certain functionality of the application 8 units tests were created to demonstrate to the programmer that the following functionality was implemented correctly.

* Be able to open Andriod test folder
* Be able to create a @LargeTest reference
* Be able to main activity rule
* Be able to create @Test reference to highlight test cases
* Be able to write testing function under each @Test reference
* Be able to write test cases which test for image click
* Be able to write test cases which test for editable location title
* Be able to run test cases using the Green arrow button next class core2test



*Figure 6: Running test cases for the application*