Task Core 2 – Spike: Sharing is caring

Link to GitHub: https://github.com/SoftDevMobDev-2023-Classrooms/core2-Dank143

Goals:

- Create an app with multiple activities.
- Pass back data from intents using ActivityResultContracts.
- Use addon resources like images.
- Apply styles in the app.
- Understand and correctly use intent.
- Use UI tests for the app.
- Use a variety of UI elements.
- Use Toast or Snackbar in app.
- Implementation of validation for user input.

Tools and Resources Used

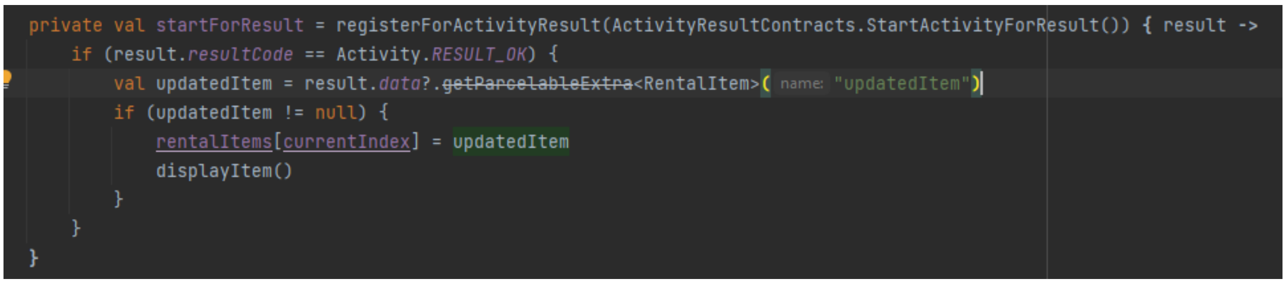
- Android Studio IDE
- Canvas documentation
- Git and GitHub
- Kotlin programming language
- XML files
- Use of RatingBar: https://www.javatpoint.com/android-rating-bar-example
- Styles and Themes: https://developer.android.com/develop/ui/views/theming/themes

Knowledge Gaps and Solutions

Gap 1: Pass back data from intents using ActivityResultContracts

This can be found in the MainActivity.kt by importing the AndroidX library with the following code:

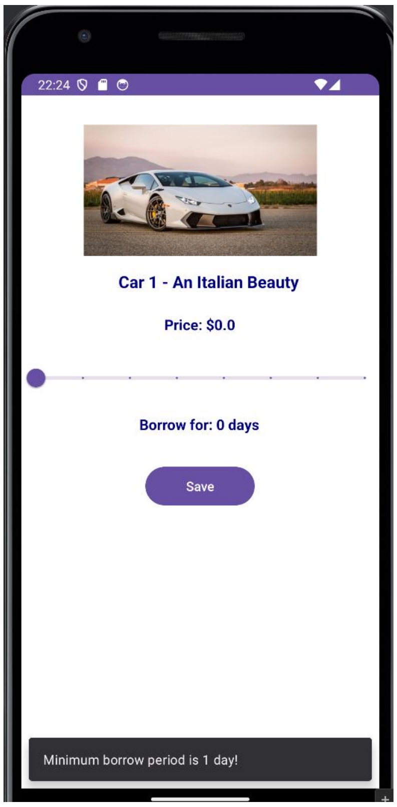
```
import androidx.activity.result.contract.ActivityResultContracts
```



```
private val startForResult = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
    if (result.resultCode == Activity.RESULT_OK) {
        val updatedItem = result.data?.~~getParcelableExtra~~<RentalItem>( name: "updatedItem")
        if (updatedItem != null) {
            rentalItems[currentIndex] = updatedItem
            displayItem()
        }
    }
}
```

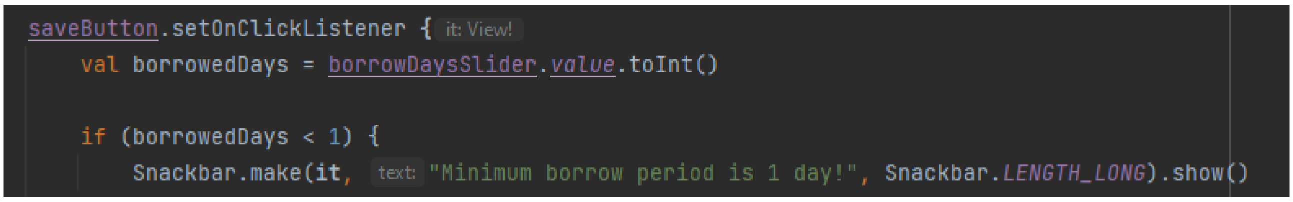
Gap 2: Validation for user input & Snackbar for error message

Should the user borrow the item for 0 days (which is impossible), there will be an error message to indicate that this action cannot be done, as a form of input validator shown in the picture below:



Minimum borrow period is 1 day!

The error message can be created using Snackbar. Using a Snackbar is a common practice in Android app development to display transient messages or notifications to the user (in this case a validation message). I set the validation message to appear if the borrow day is less than one:

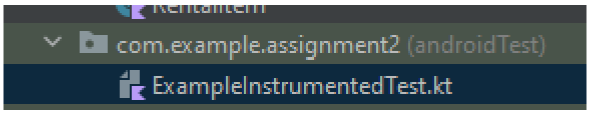


```
saveButton.setOnClickListener { it: View!
    val borrowedDays = borrowDaysSlider.value.toInt()

    if (borrowedDays < 1) {
        Snackbar.make(it,  text: "Minimum borrow period is 1 day!", Snackbar.LENGTH_LONG).show()
```

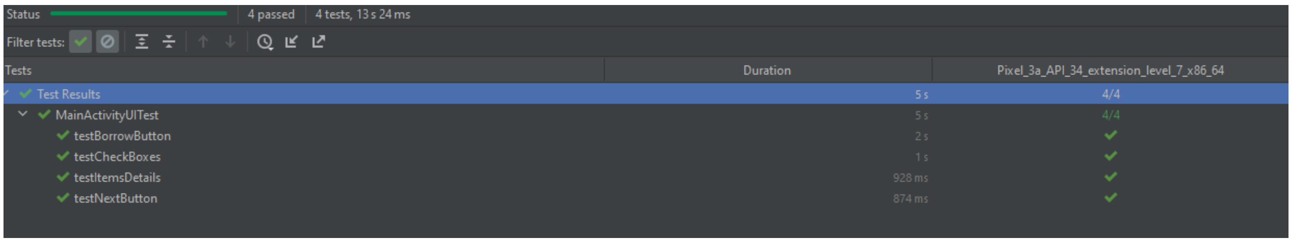
Gap 3: Implementation of Unit and UI tests

I have created several tests for the app that can be found in the ExampleInstrumentedTest.kt. These tests include:

- Test whether the borrow action is working correctly (Borrow button – Set days of borrow using slider – Save button – Due date is not “Not borrowed”).
- Test whether pressing the Next button will go through different items.
- Test whether the 2 checkboxes (Used and Brand New) are working correctly.
- Test whether the description of the item is a match.


RentalItem
com.example.assignment2 (androidTest)
ExampleInstrumentedTest.kt

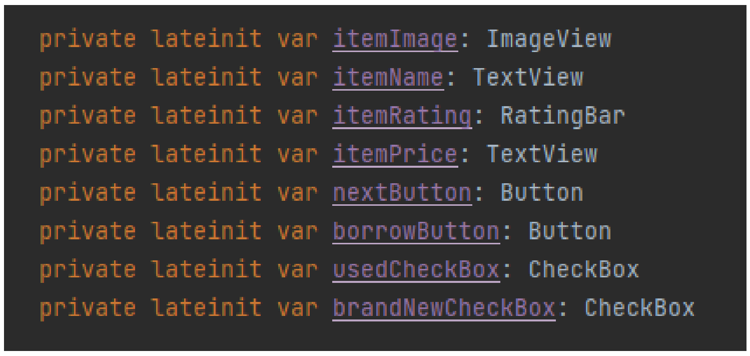
All 4 tests have passed in the Espresso test.



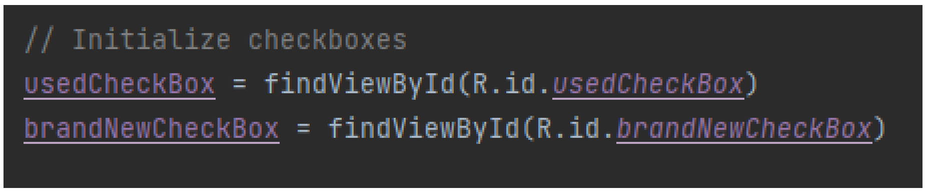
```
Status  4 passed  4 tests, 13 s 24 ms
Filter tests:
Tests                        Duration   Pixel_3a_API_34_extension_level_7_x86_64
Test Results                      5 s   4/4
  MainActivityUITest              5 s   4/4
    testBorrowButton              2 s   ✓
    testCheckBoxes                1 s   ✓
    testItemsDetails           928 ms   ✓
    testNextButton             874 ms   ✓
```

Gap 4: Using multiple UI elements

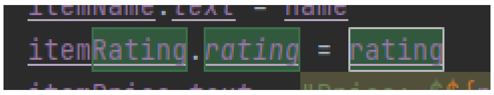
I have implemented RatingBar and CheckBox as the advanced UI elements for this app. This can be found in the MainActivity.kt



```
private lateinit var itemImage: ImageView
private lateinit var itemName: TextView
private lateinit var itemRating: RatingBar
private lateinit var itemPrice: TextView
private lateinit var nextButton: Button
private lateinit var borrowButton: Button
private lateinit var usedCheckBox: CheckBox
private lateinit var brandNewCheckBox: CheckBox
```



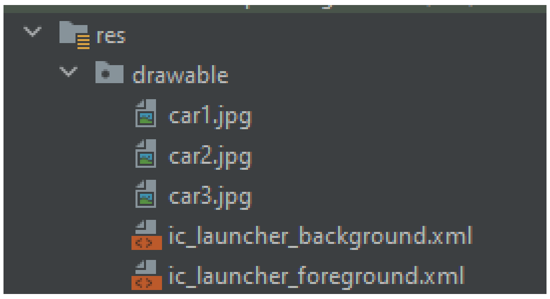
```
// Initialize checkboxes
usedCheckBox = findViewById(R.id.usedCheckBox)
brandNewCheckBox = findViewById(R.id.brandNewCheckBox)
```



```
itemRating.rating = rating
```

Gap 5: Working with resources (images)

I have included 3 car images in the /app/res/drawable folder by dragging the images straight to the folder, which is faster than the conventional way.

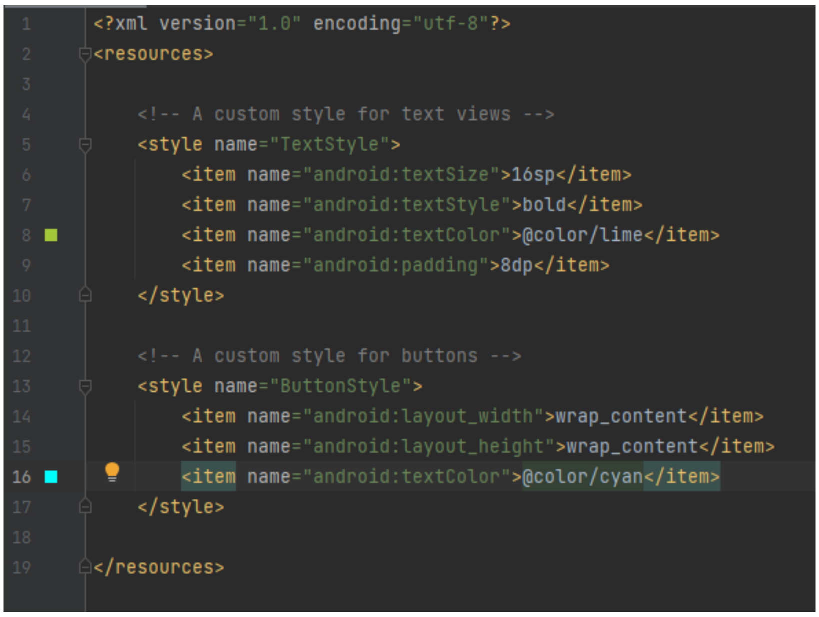


```
res
  drawable
    car1.jpg
    car2.jpg
    car3.jpg
    ic_launcher_background.xml
    ic_launcher_foreground.xml
```

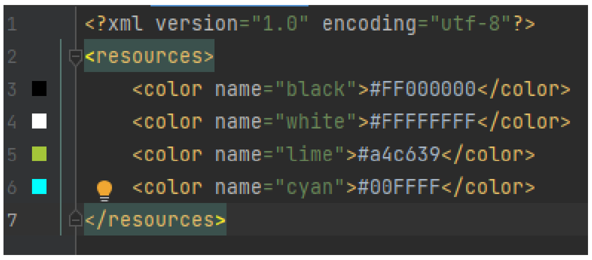
Car images resource: I got all from WallpapersCraft: https://wallpaperscraft.com

Gap 6: Using styles in the app

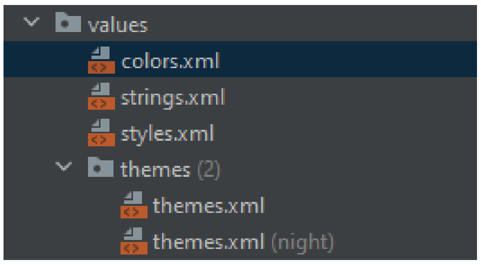
I have used 2 different styles, one for text views and one for buttons. They are placed in the value folder and mostly related to the color of the text, as shown in the pictures below:



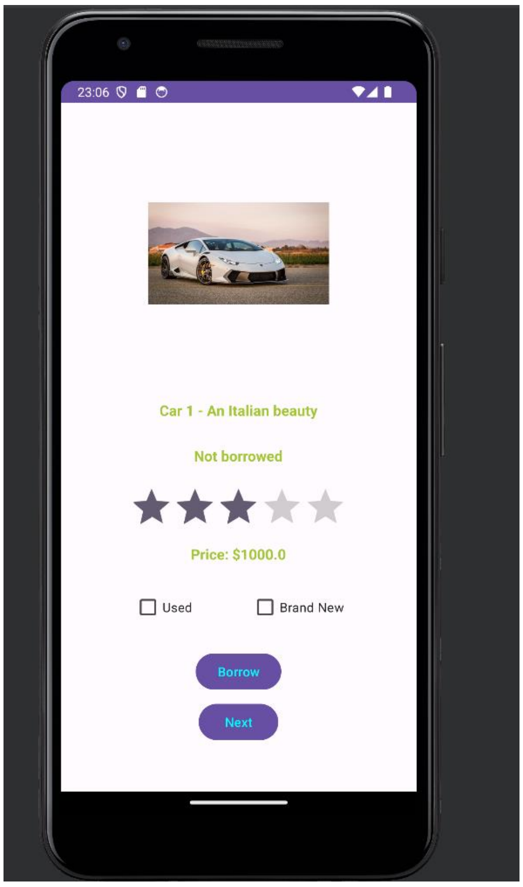
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <!-- A custom style for text views -->
    <style name="TextStyle">
        <item name="android:textSize">16sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textColor">@color/lime</item>
        <item name="android:padding">8dp</item>
    </style>

    <!-- A custom style for buttons -->
    <style name="ButtonStyle">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">@color/cyan</item>
    </style>

</resources>
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="lime">#a4c639</color>
    <color name="cyan">#00FFFF</color>
</resources>
```

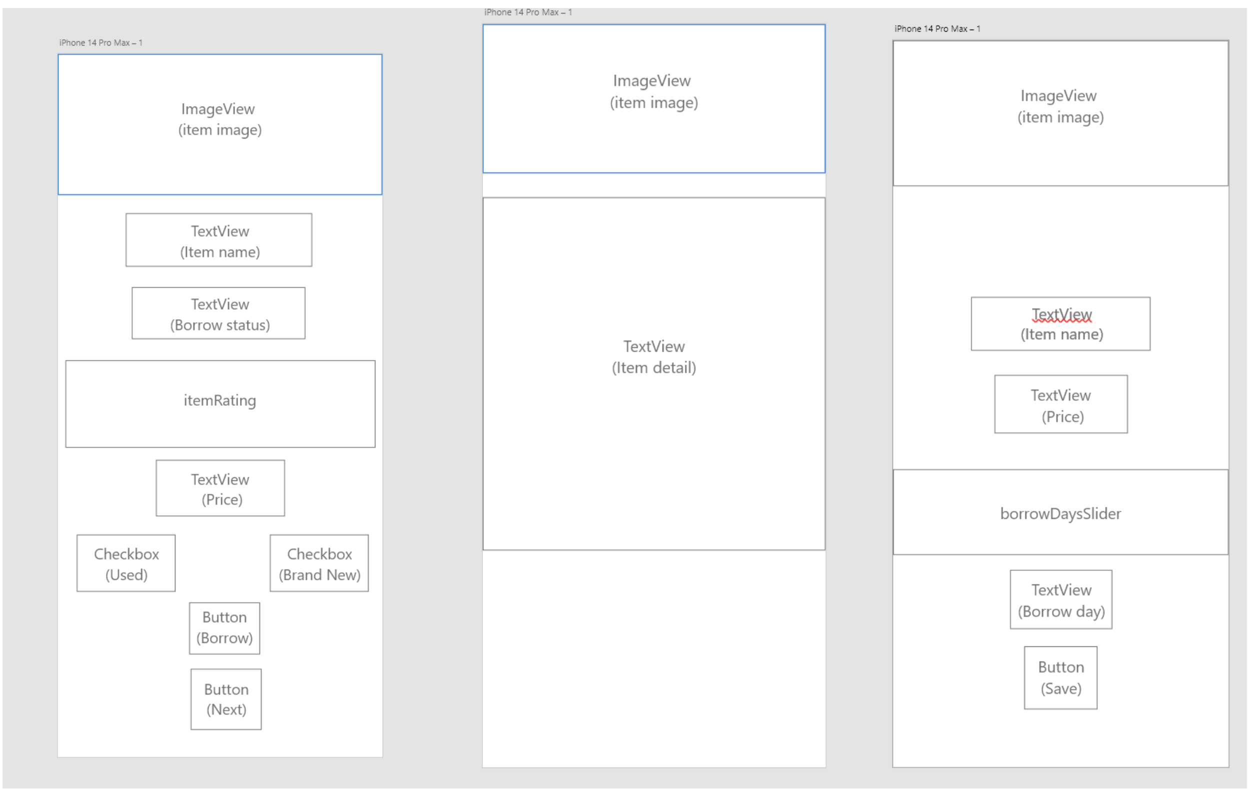


The result would look like this:



Gap 7: Using sketches for simple screens

Here are my initial sketches for the app using Adobe XD. These are quite rough as I just need the idea for my app layout:



From left to right: Activity main, activity item details, activity details.

Gap 8: Using intents and explanations

In general, intents are often used for starting new activities, launching external apps, and passing information between various parts of the application. In this app, intent is used in these activities:

1. MainActivity: The MainActivity is the entry point of the application, and it displays a list of rental items. It uses Intents in the following ways:

Borrow Button Click: When the "Borrow" button is clicked, an Intent is used to launch the DetailsActivity for the selected rental item. Data about the selected item is passed to DetailsActivity using the putExtra method of the Intent.

```
borrowButton.setOnClickListener {
    val intent = Intent(this, DetailsActivity::class.java)
    intent.putExtra("selectedItem", rentalItems[currentIndex])
    startForResult.launch(intent)
}
```

Item Image Click: When the item image is clicked, an Intent is used to launch the ItemDetailActivity for the selected rental item. Data about the selected item is passed to ItemDetailActivity using the putExtra method of the Intent.

```
itemImage.setOnClickListener {
    val intent = Intent(this, ItemDetailActivity::class.java)
    intent.putExtra("selectedItem", rentalItems[currentIndex])
    startActivity(intent)
}
```

2. DetailsActivity: The DetailsActivity is responsible for displaying detailed information about a selected rental item and allowing the user to interact with it. Intents are used in the following ways:

Save Button Click: When the "Save" button is clicked, an Intent is created to pass updated information about the rental item back to the MainActivity. This is done by creating an Intent, adding data to it using putExtra, and then setting the result of the activity to Activity.RESULT_OK before finishing.

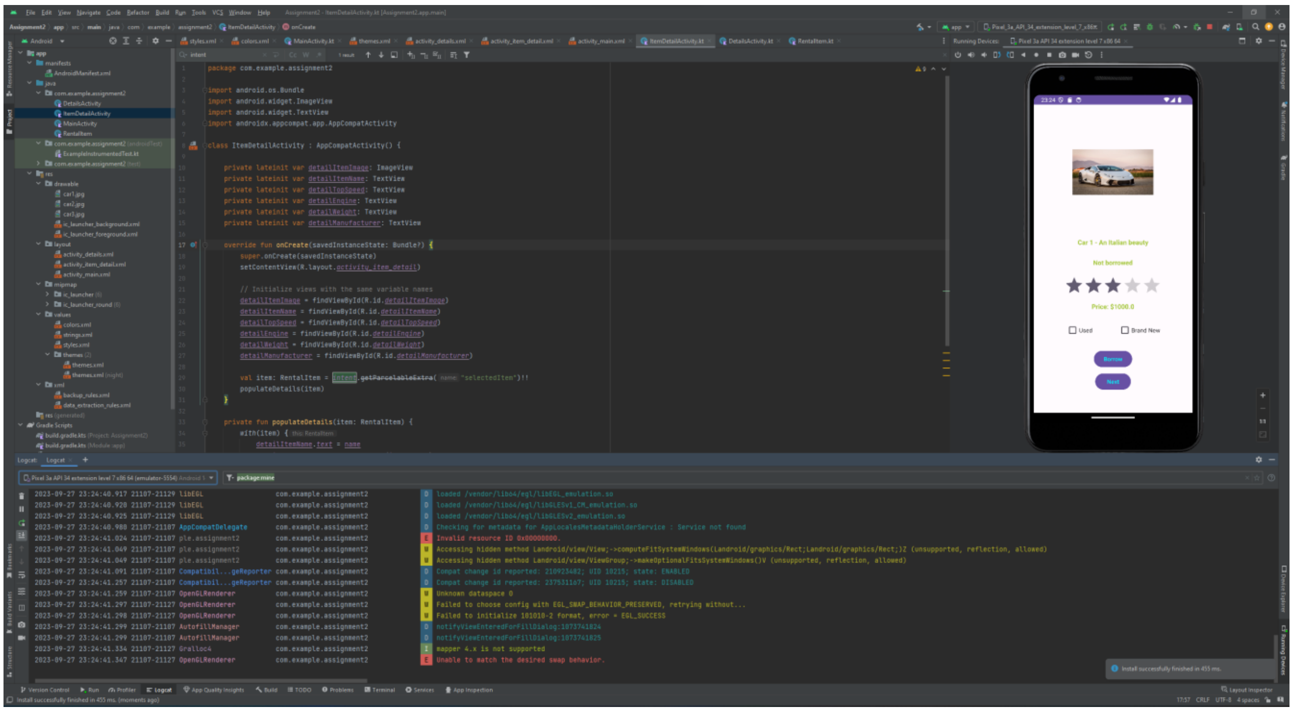
```
val resultIntent = Intent().apply {
    putExtra("updatedItem", item)
    putExtra("borrowDays", borrowedDays)
}

Toast.makeText(this, "Successfully borrowed for $borrowedDays days",
Toast.LENGTH_SHORT).show()
setResult(Activity.RESULT_OK, resultIntent)
finish()
```

3. ItemDetailActivity: The ItemDetailActivity displays detailed information about a selected rental item. It receives data about the item through an Intent, which is retrieved in the onCreate method.

```
val item: RentalItem = intent.getParcelableExtra("selectedItem")!!
```

Gap 9: Command of IDE

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



```
package com.example.assignment2

import android.os.Bundle
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class ItemDetailActivity : AppCompatActivity() {

    private lateinit var detailItemImage: ImageView
    private lateinit var detailItemName: TextView
    private lateinit var detailTopSpeed: TextView
    private lateinit var detailEngine: TextView
    private lateinit var detailWeight: TextView
    private lateinit var detailManufacturer: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_item_detail)

        // Initialize views with the same variable names
        detailItemImage = findViewById(R.id.detailItemImage)
        detailItemName = findViewById(R.id.detailItemName)
        detailTopSpeed = findViewById(R.id.detailTopSpeed)
        detailEngine = findViewById(R.id.detailEngine)
        detailWeight = findViewById(R.id.detailWeight)
        detailManufacturer = findViewById(R.id.detailManufacturer)

        val item: RentalItem = intent.getParcelableExtra( name: "selectedItem")!!
        populateDetails(item)
    }

    private fun populateDetails(item: RentalItem) {
        with(item) { this: RentalItem
            detailItemName.text = name
```

```
2023-09-27 23:24:40.917 21107-21129 libEGL                  com.example.assignment2   D  loaded /vendor/lib64/egl/libEGL_emulation.so
2023-09-27 23:24:40.920 21107-21129 libEGL                  com.example.assignment2   D  loaded /vendor/lib64/egl/libGLESv1_CM_emulation.so
2023-09-27 23:24:40.925 21107-21129 libEGL                  com.example.assignment2   D  loaded /vendor/lib64/egl/libGLESv2_emulation.so
2023-09-27 23:24:40.988 21107-21107 AppCompatDelegate       com.example.assignment2   D  Checking for metadata for AppLocalesMetadataHolderService : Service not found
2023-09-27 23:24:41.024 21107-21107 ple.assignment2         com.example.assignment2   E  Invalid resource ID 0x00000000.
2023-09-27 23:24:41.049 21107-21107 ple.assignment2         com.example.assignment2   W  Accessing hidden method Landroid/view/View;->computeFitSystemWindows(Landroid/graphics/Rect;Landroid/graphics/Rect;)Z (unsupported, reflection, allowed)
2023-09-27 23:24:41.049 21107-21107 ple.assignment2         com.example.assignment2   W  Accessing hidden method Landroid/view/ViewGroup;->makeOptionalFitsSystemWindows()V (unsupported, reflection, allowed)
2023-09-27 23:24:41.091 21107-21107 Compatibil...geReporter com.example.assignment2   D  Compat change id reported: 210923482; UID 10215; state: ENABLED
2023-09-27 23:24:41.257 21107-21107 Compatibil...geReporter com.example.assignment2   D  Compat change id reported: 237531167; UID 10215; state: DISABLED
2023-09-27 23:24:41.259 21107-21107 OpenGLRenderer          com.example.assignment2   W  Unknown dataspace 0
2023-09-27 23:24:41.297 21107-21127 OpenGLRenderer          com.example.assignment2   W  Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
2023-09-27 23:24:41.298 21107-21127 OpenGLRenderer          com.example.assignment2   W  Failed to initialize 101010-2 format, error = EGL_SUCCESS
2023-09-27 23:24:41.299 21107-21107 AutofillManager         com.example.assignment2   D  notifyViewEnteredForFillDialog:1073741824
2023-09-27 23:24:41.299 21107-21107 AutofillManager         com.example.assignment2   D  notifyViewEnteredForFillDialog:1073741825
2023-09-27 23:24:41.334 21107-21127 Gralloc4                com.example.assignment2   I  mapper 4.x is not supported
2023-09-27 23:24:41.347 21107-21127 OpenGLRenderer          com.example.assignment2   E  Unable to match the desired swap behavior.
```

Install successfully finished in 455 ms. (moments ago)

Gap 10: Using Toast to create pop-up messages

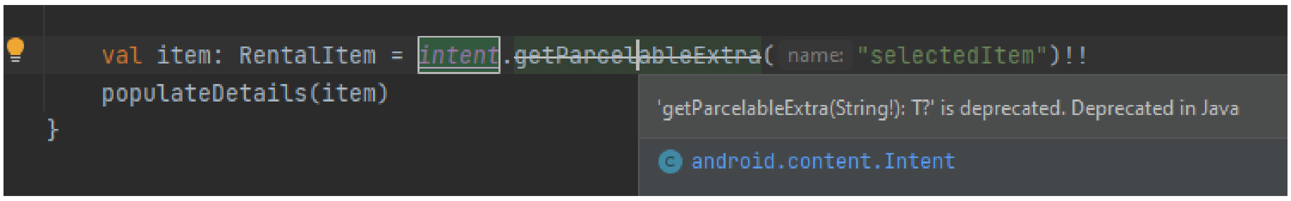
In addition to the use of Snackbar in Gap 2, I have also implemented Toast for when the item is successfully borrowed, and for when the user return from the borrow day slider page:

```
Toast.makeText(this, "Successfully borrowed for $borrowedDays days",
Toast.LENGTH_SHORT).show()
setResult(Activity.RESULT_OK, resultIntent)
finish()
```

```
override fun onBackPressed() {
    Toast.makeText(this, "Continue finding the car you want!",
Toast.LENGTH_SHORT).show()
    super.onBackPressed()
}
```

Open Issues and Recommendations

The issue I'm having is that getParcelableExtra() is crossed out with the error of it being deprecated. I have tried to look for the solution online and it is suggested that lowering the API level can help to resolve it. I haven't succeeded in doing so and I haven't found any alternative solutions. I hope that I could receive some feedback on any alternative methods or confirmation that this issue is acceptable.



```
    val item: RentalItem = intent.~~getParcelableExtra~~( name: "selectedItem")!!
    populateDetails(item)
}
```

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

android.content.Intent