

# Assignment 1

*by* Chan Seow Fen

---

**Submission date:** 23-Oct-2023 10:20PM (UTC+0800)

**Submission ID:** 2204702428

**File name:** New\_Microsoft\_Word\_Document.docx (14.6M)

**Word count:** 3053

**Character count:** 15286

## 1.0 Brief Report on Application Building

### 1.1 Components

The components used in building the application, Simple Discount Calculator App included 1 Main Activity (MainActivity.kt), which is the user interface controller for the user to interact with the application and do the calculation for discount. Furthermore, the project included 1 data class (DiscountData.kt) to represent the data layer, which is responsible to store the user input (Sales Price, Discount, Tax) and output, which is the discount information (Save Amount, Tax Amount, Total Price). In addition, the project also included 1 view model (DiscountModel.kt) for data persistence as it corresponds to the project requirement, which enable the user to retrieve back the discount information even though there happens a configuration change such as rotating phone. Moreover, the project also having an object class (Calculation.kt) to handle business logic, which is calculating the discount information in this case. Furthermore, the project included a class (AboutDialog.kt) for showing application information such as the application version and the programmer's name. In addition, the project also has a class for (RangeInputFilters) limiting the range of input. Moreover, it corresponds to the requirements of the project, two layouts, which is portrait and landscape layout is included for the user to rotate the application according to their preference. Lastly, the project components included 4 resources files (menu.xml, colours.xml, strings.xml, themes.xml) which used to apply for design themes as well as holding for each piece of text used in the application.

### 1.2 Design Themes

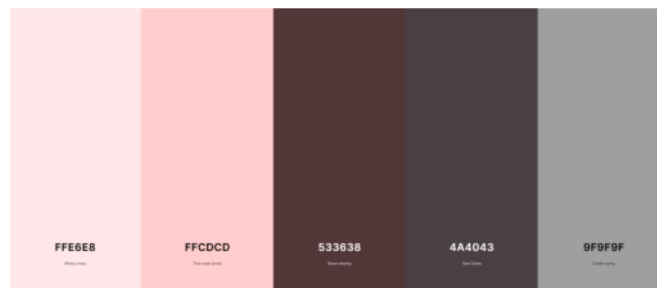


Figure 1.1 Colour Palette of User Interface Design

As shown in *Figure 1.1*, the design theme of the application is basically using the main colour, pink and brown which is a very aesthetically combination as a palette. There are also other colours that are close to the main theme colour as a minor colour to make the application seem livelier and more entertaining. The design is more towards the minimalist idea, which keeping the user interface to be as simple as possible, clean and without images that will distract the user from the main task, which is calculating the discount information, as well as reducing the system performance.

Moreover, the design of the application also focuses on user friendliness, which reduce the user's input as well as speed up the discount calculation. To illustrate that, the application allows users to directly switch upon Percent Off mode or Fixed Amount mode without the need to jump into other view which make the user convenient to perform task. Furthermore, users are allowed to slide the slider (SeekBar) for the tax input to the desired amount instead of typing by themselves. The application also remain the availability of text input for user to type the value themselves and make the slider (SeekBar) as an option for the user to choose according to their

preferences. It is also same for the Discount Percentage in Percent Off mode. In addition, the user also allowed to enable or disable the calculation for including Tax Rate, by only single touch on the With Tax Switch which serves convenience to the user. On top of that, if user choose to disable Tax Rate, the Tax Rate input will automatically fill in with 0 as 0% without the need for users to type in themselves. Moreover, in Fixed Amount mode, the slider (SeekBar) is removed from the user interface and replaced with 6 number button (-10, -5, -1, +1, +5, +10), which allows the user to press on it and modify the Discount Amount without the need to type in themselves. As identical to Discount Percentage and Tax Rate, the input field is also remained available for user to type in themselves to fulfil different preferences of all users. Last but not least, the application also provided a feature to clear all including the input by user as well as the previous discount information by one click, which is on the toolbar option. It makes the user convenient as they do not have to clear the input field one by one themselves.

### **1.3 GUI Widgets**

The GUI Widgets that have been applied in the application included TextView, EditText, Button, Switch, RadioGroup and RadioButton, SeekBar, toolbar and CardView. Some of the widgets such as toolbar, Radio Button, SeekBar, Switch and Button have applied themes as in themes resource to make it aesthetic, minimalistic and follow the ideal user interface design. To further illustrate, TextView have been used for the header (Select Discount Type), the symbol at the end of input field (\$, %) as well as for displaying the discount information (Your Save:, Tax:, Total Price:...). Next, EditText is used for user input (Sales Price, Discount, Tax). Following that, Button is used for the user to perform calculation (Calculate). Then, RadioGroup is used as a container for 2 RadioButton (Percent Off, Fixed Amount) which serve for users to select discount option. Next, SeekBar is applied for Percent Off Discount and Tax Rate for user to drag the value. Following that, toolbar is the one on the top of the user interface, which showing the application title, application information after pressing the about icon and clear all option after pressing the triple dot icon. Lastly, CardView is used as a container to place the discount information in a card view which make the application cleaner and more aesthetic.

## 2.0 Screenshots of Application

### 2.1 Portrait Mode

MY PHONE 100% 01:42

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

Enter Price \$

Enter Discount %

With Tax ☒ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 2.1 Portrait Mode on Percent Off and With Tax On (Calculate Button Disabled as no input)

MY PHONE 100% 01:42

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☐ Percent Off ☒ Fixed Amount

Enter Price \$

Enter Discount \$

-10 -5 -1 +1 +5 +10

With Tax ☒ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 2.2 Portrait Mode on Fixed Amount and With Tax On (Calculate Button Disabled as no input)

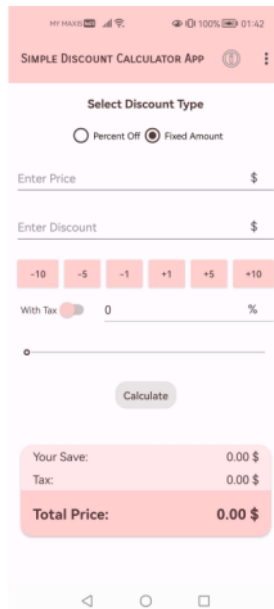


Figure 2.3 Portrait Mode on Fixed Amount and With Tax Off (Calculate Button Disabled as no input)



Figure 2.4 Portrait Mode on Fixed Amount and With Tax Off (Calculate Button is enabled as input is valid)

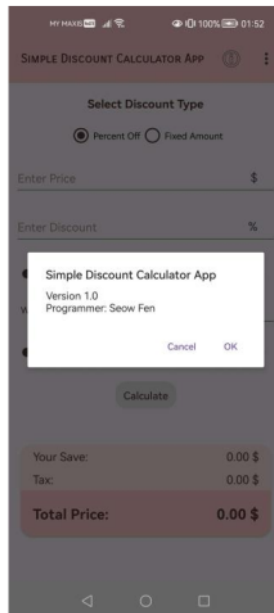


Figure 2.5 Portrait Mode About Message

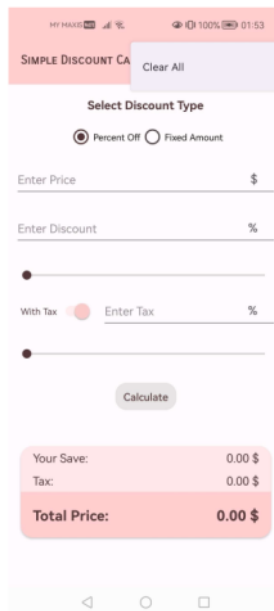


Figure 2.6 Portrait Mode Tool Bar

## 2.2 Landscape Mode

MY MAXIS 100% 01:41

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

Enter Price \$ Enter Discount %

Calculate

With Tax ☐ Enter Tax %

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 2.7 Landscape Mode on Percent off

MY MAXIS 100% 01:41

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☐ Percent Off ☒ Fixed Amount

Enter Price \$ Enter Discount \$

Calculate

-10 -5 -1 +1 +5 +10

With Tax ☐ Enter Tax %

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 2.8 Landscape Mode on Fixed Amount

### 3.0 Test Cases

#### 3.1 Percent Off Option

##### 3.1.1 Test Case 1

	Input		Expected Output
Sales Price (\$)	45	Your Save:	4.50 \$
Percent Off (%)	10	Tax:	0.00 \$
Tax (%)	0	Total Price:	40.50 \$

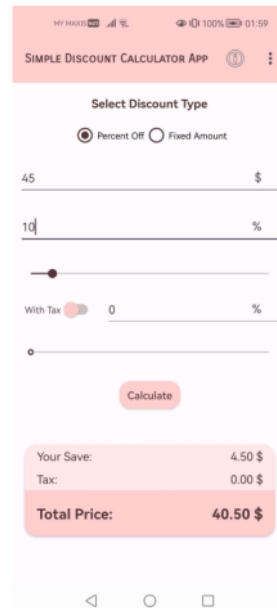


Figure 3.1 Percent Off Test Case 1 Output



### 3.1.2 Test Case 2

	Input		Expected Output
Sales Price (\$)	60	Your Save:	15.00 \$
Percent Off (%)	25	Tax:	6.00 \$
Tax (%)	10	Total Price:	51.00 \$

The screenshot shows a mobile application interface for a 'SIMPLE DISCOUNT CALCULATOR APP'. At the top, there's a status bar with 'MY PHONE', signal strength, 100% battery, and the time 02:09. Below the app title, a section titled 'Select Discount Type' has two radio buttons: 'Percent Off' (selected) and 'Fixed Amount'. The input fields are: 'Sales Price (\$)' with a value of 60, 'Percent Off (%)' with a value of 25, and 'With Tax' with a value of 10. Each input has a corresponding slider below it. A 'Calculate' button is positioned below the sliders. The output section at the bottom displays: 'Your Save: 15.00 \$', 'Tax: 6.00 \$', and 'Total Price: 51.00 \$'. The bottom of the screen shows standard Android navigation icons.

Figure 3.2 Percent Off Test Case 2 Output

### 3.2 Fixed Amount Option

#### 3.2.1 Test Case 1

	Input		Expected Output
Sales Price (\$)	50	Your Save:	10.00\$
Fixed Amount (\$)	10	Tax:	0.00 \$
Tax (%)	0	Total Price:	40.00\$

MY PHONE 100% 01:59

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☐ Percent Off ☒ Fixed Amount

50 \$

10 \$

-10 -5 -1 +1 +5 +10

With Tax ☐ 0 %

Calculate

Your Save: 10.00 \$

Tax: 0.00 \$

Total Price: 40.00 \$

Figure 3.3 Fixed Amount Test Case 1 Output

### 3.2.2 Test Case 2

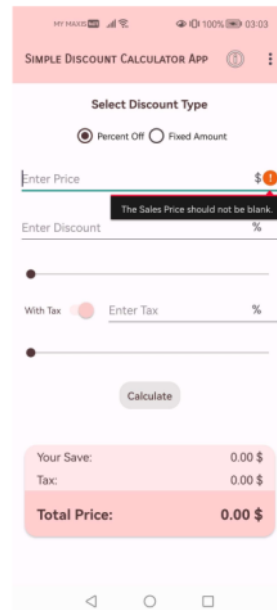
	Input		Expected Output
Sales Price (\$)	1016	Your Save:	210.00\$
Fixed Amount (\$)	210	Tax:	60.96 \$
Tax (%)	6	Total Price:	866.96

The screenshot shows a mobile application interface for a 'SIMPLE DISCOUNT CALCULATOR APP'. The status bar at the top indicates 'MY PHONE', signal strength, Wi-Fi, 100% battery, and the time 02:16. The app title is 'SIMPLE DISCOUNT CALCULATOR APP' with an information icon and a menu icon. The main heading is 'Select Discount Type' with two radio buttons: 'Percent Off' (unselected) and 'Fixed Amount' (selected). Below this, there are two input fields: the first contains '1016' with a '\$' symbol, and the second contains '210' with a '\$' symbol. A row of six buttons follows: '-10', '-5', '-1', '+1', '+5', and '+10'. Below these is a 'With Tax' section with a toggle switch set to 'On', a text input field containing '6', and a '%' symbol. A horizontal slider is positioned below the input field. A 'Calculate' button is centered below the slider. The output section at the bottom displays three items: 'Your Save: 210.00 \$', 'Tax: 60.96 \$', and 'Total Price: 866.96 \$'. The bottom of the screen shows standard Android navigation icons.

Figure 3.4 Fixed Amount Test Case 2 Output

### 3.3 Input Validation

#### 3.3.1 Sales Price Validation



MY PHONE 100% 03:03

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

Enter Price \$

The Sales Price should not be blank.

Enter Discount %

With Tax ☐ Enter Tax %

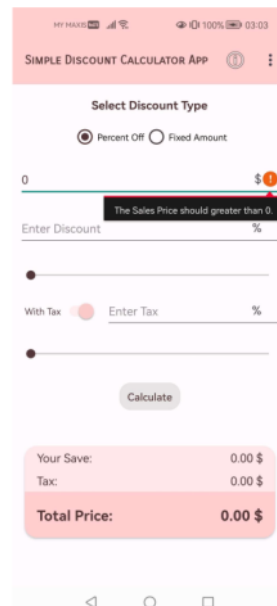
Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.1 Invalid Sales Price – input is blank



MY PHONE 100% 03:03

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

0 \$

The Sales Price should greater than 0.

Enter Discount %

With Tax ☐ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.2 Invalid Sales Price – input is not greater than 0

### 3.3.2 Percent Off Discount Validation

MY PHONE 100% 03:03

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

10 \$

Enter Discount %

The Discount Percentage should not be blank.

With Tax ☐ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.3 Invalid Discount (Percent Off) – input is blank

MY PHONE 100% 03:03

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☒ Percent Off ☐ Fixed Amount

10 \$

0 %

The Discount Percentage should be greater than 0.

With Tax ☐ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.4 Invalid Discount (Percent Off) – input is not greater than 0

### 3.3.3 Fixed Amount Discount Validation

MY PHONE 100% 16:08

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☐ Percent Off ☒ Fixed Amount

10 \$

Enter Discount \$

The Discount Amount should not be blank.

-10 -5 -1 +1 +5 +10

With Tax ☐ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.5 Invalid Discount (Fixed Amount) – input is blank

MY PHONE 100% 16:08

SIMPLE DISCOUNT CALCULATOR APP

Select Discount Type

☐ Percent Off ☒ Fixed Amount

10 \$

0 \$

The Discount Amount should be greater than 0.

-10 -5 -1 +1 +5 +10

With Tax ☐ Enter Tax %

Calculate

Your Save: 0.00 \$

Tax: 0.00 \$

Total Price: 0.00 \$

Figure 4.6 Invalid Discount (Fixed Amount) – input is not greater than 0

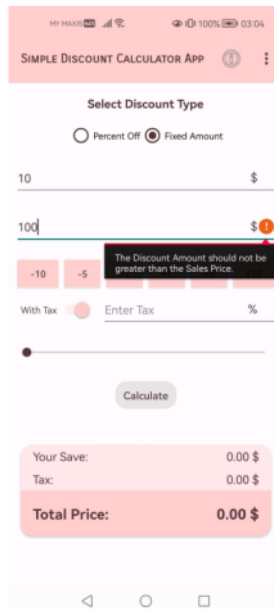


Figure 4.7 Invalid Discount (Fixed Amount) – input is greater than Sales Price

### 3.3.4 Tax Validation

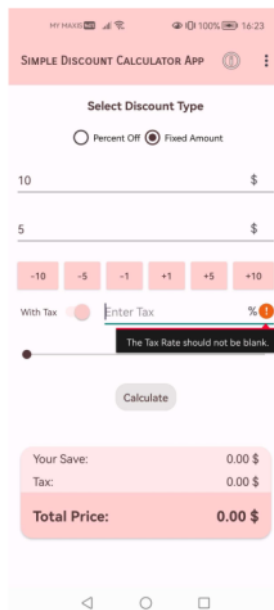


Figure 4.8 Invalid Tax – input is blank

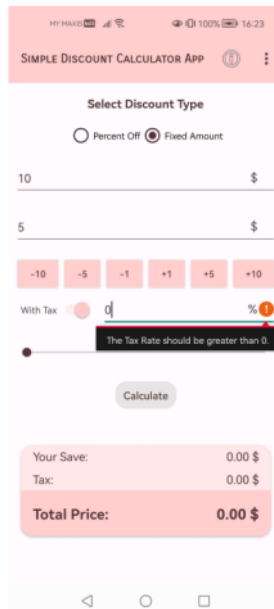


Figure 4.9 Invalid Tax – input is not greater than 0



### 3.3.5 Validation Handling Details

As range filter function is implemented for Sales Price [0.00-10000000.00], Discount (Percent Off) [0-100], Discount Fixed Amount [0-10000000] and Tax [0-100], hence it will prevent user from entering input other than that range, thus reducing the error handling of input validation. Moreover, the slider (SeekBar) for Percent Off discount and Tax, as well as the button for Fixed Amount have set to ensure that the input field aligns with their respective range.



Figure 4.10 In Fixed Amount Mode

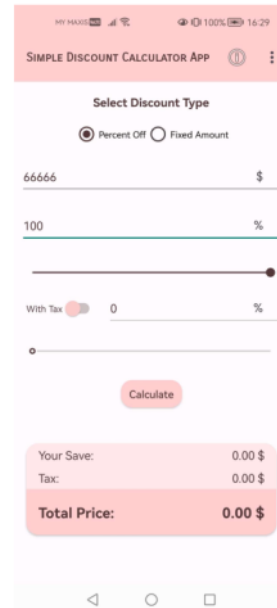


Figure 4.11 Switch to Percent Off Mode

Moreover, the validation handling also included when user entered number that is greater than 100 in Fixed Amount mode and switch to Percent Off, the program will automatically change the Discount Edit Text into 100 as 100 is the maximum for Percent Off mode as shown in Figure 4.10, Figure 4.11.

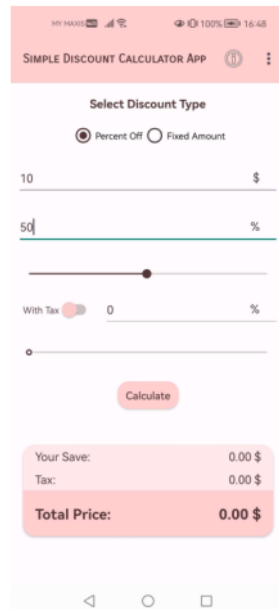


Figure 4.12 In Percent Off Mode

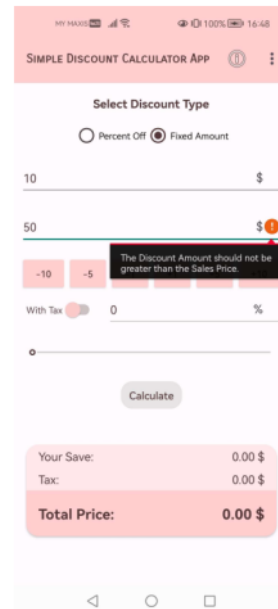


Figure 4.13 Switch to Fixed Amount Mode

Furthermore, there is also a validation handling that handles the checking of Discount (Fixed Amount) is valid or not after switching the mode from Percent Off to Fixed Amount even though user did not make changes to the input field as shown in Figure 4.12, Figure 4.13.

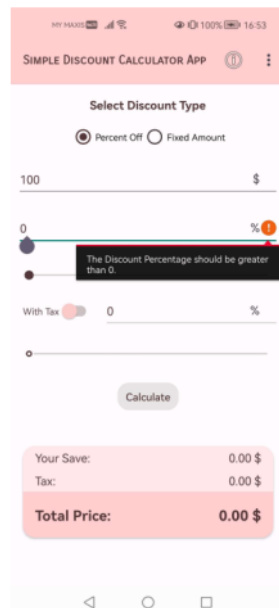


Figure 4.14 In Percent Off Mode

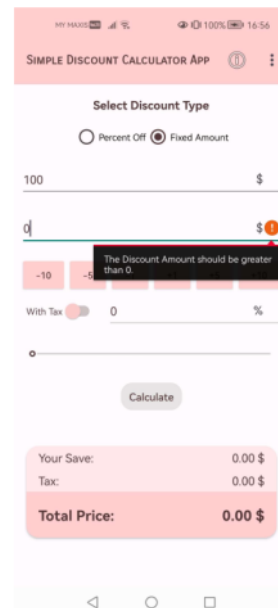
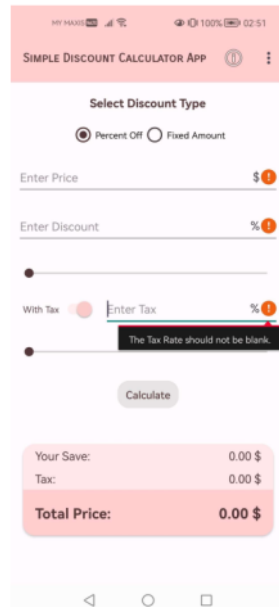


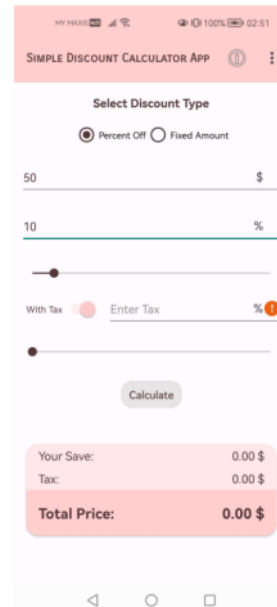
Figure 4.15 Switch to Fixed Amount Mode

In addition, the program also able to switch the error message *according* to the mode that user switch to as shown in *Figure 4.14*, *Figure 4.15* as one of the examples.

### **3.4 Enable of Calculate Button**



*Figure 5.1* 3 Invalid Input-- Calculate Button Disabled



*Figure 5.2* 1 Invalid Input - Calculate Button Disabled

Figure 5.1 and Figure 5.2 shows that, the Calculate button is not enable as long as there is an invalid input.



Figure 5.3 No Invalid Input – Calculate Button Enabled

Figure 5.3 shows that, after all the input is valid, the Calculate button is then only enabled.

### 3.5 Data Persistence

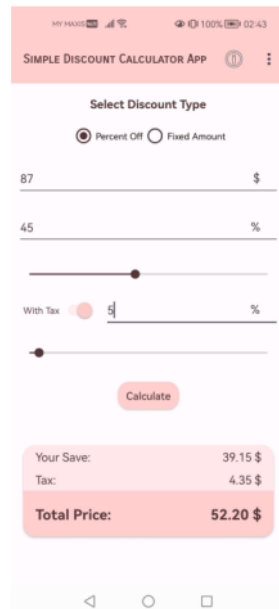


Figure 6.1 Before Configuration Changes

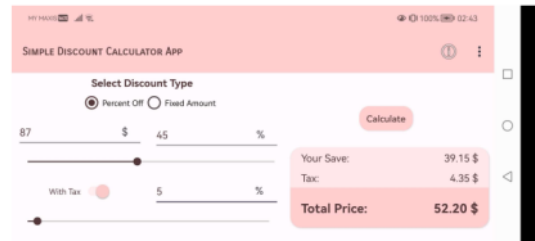


Figure 6.2 After Configuration Changes

As shown in *Figure 6.1* and *Figure 6.2*, the application is able to save and restore the discount information (save amount, tax amount, total price) after configuration changes, which is rotating the phone.

## 4.0 Weaknesses and Strengths

### 4.1 Weaknesses

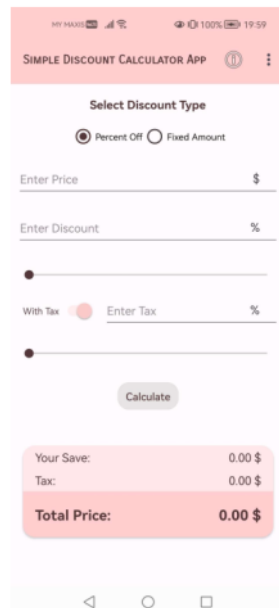


Figure 7.1 Before Rotating

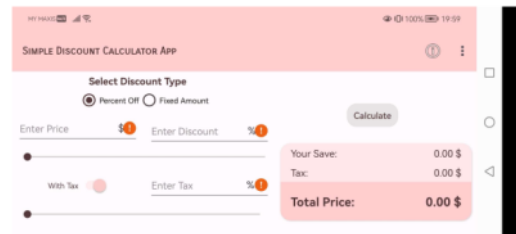


Figure 7.2 After Rotating

One of the weaknesses of the application is that, if the user rotates the phone which cause the configuration changes while have not entered any input yet, it will immediately show the error message for the input validation as shown in *Figure 7.1*, *Figure 7.2* which is undesired behaviour as error message should only show after user have modify the input field with invalid input. Due to the time constraints, have not get to figure out the solution to eliminate this weakness of the application.

Moreover, another weakness of the application is unable to accept the Discount Amount for Fixed Amount Mode as Double type which allow decimal points. The reason is that both Discount Percentage and Discount Amount for Percent Off and Fixed Amount mode is using the same EditText widget, implying that if the input type of Discount changed into numberDecimal, the SeekBar progress for Discount Percentage will also be number with decimal point, which is inconvenient for user to drag as the sensitivity is too high, need to drag very accurately for whole number (Eg: 80.00). Hence, for better user experience, Discount Amount of Fixed Amount mode could only accept integer. It might be a solution to eliminate this weakness but however have not get to figure out due to time constraints.

### 4.2 Strengths

First of all, one of the strengths of the application is that it has a clean, minimalistic and user-friendly interface. Moreover, the application has applied custom styles to make the application visually appealing and align with Material Design Guidelines or Human Computer Interaction (HCI) standards.

In addition, the application come along with real-time validation and error handling for user inputs as soon as user have modified the input field. To further illustrate the application has included plenty of handling that cover most of the possible situation that may cause error. For example, as mentioned in [3.3.5 Validation Handling Details](#), even though a range has been set for Discount Percentage of Percent Off mode, it might happens the situation that user key in value larger than the range during Fixed Amount mode and switch to Percent Off mode, causing the error that the input is not flagged as invalid as the TextWatcher only work after user make changes on the EditText, hence set the value programmatically whenever user switch from Fixed Amount to Percent Off mode with Discount that greater than 100 to 100 will handle this situation.

Furthermore, flexibility and convenient to use is one of the highlights of this application. To illustrate that, as stated in [1.2 Design Themes](#), the application allows user to choose between directly enter the input themselves or using SeekBar or Button to enter the input. Moreover, the "Clear All" tools option allows users to clear all the input instantly without the need to delete the input manually. In addition, users can also switch between Percent Off and Fixed Amount mode easily and quickly and perform task according to their needs. Moreover, users can choose to perform discount calculation with tax or without including tax rate easily by just turning on and off on the "With Tax" Switch, thus serving convenience and flexibility to users.

Last but not least, the pros of the application that lies on the code layer is that having a proper code structure with commenting that highly improve the readability and maintainability of the code. To illustrate that, most of the code have commented with their respective purpose and similar code have been placed together instead of distributed around everywhere. In addition, the code is well-structured and follows best practices for Android Applications Development. For instance, separating User Interface logic from business logic.

## **5.0 Self Reflection**

### **5.1 Course Impact and Accomplishments**

After enrolling the course, Mobile Application Development (CET3013), I have developed a better understanding on programming which I never learnt such thoroughly and understood the theory. Unlike simply memorizing theory like what I used to do in the past, I am now able to understand the underlying principles and concepts deeply. This is crucial aspect of learning as it empowers me to apply my knowledge effectively. Moreover, I have developed the skill for developing a functional Android Application, which will be very valuable and inspirational, no matter on unlocking a new option for my final year project, or on potential career purpose in the future, or even as a hobby to develop the ideal mobile application that fits my personal requirements and preferences.

By working on this project, I have achieved on implementing functional Switch, SeekBar and TextWatcher that have not been covered in the lecture by researching the online materials. Moreover, I also accomplished the bidirectional synchronization on the EditText and SeekBar, which means the changes made on EditText will reflect on the SeekBar as progress and vice versa, which could provide an interactive user experience. In addition, I also accomplished success on the aspect of handling error of most of the possible situations according to my user interface design. Moreover, by building this application, I also get to understand more about mobile application development as building by my own can always learn more than simply following guidelines or instructions.

### **5.2 Successes and Problem-Solving in Design and Implementation**

Things that went well during the design and implementation of the application includes the user interface, which I am very satisfied with the end product which is ideal and aligned with my aesthetic conception. Most of the design part went well when I am working on the project, there are only a few minor issues and can be addressed by adding theme. However, there is a issue that cost me a lot of time to resolve it, which is, the Fixed Amount Discount Button (-10, -5, -1, +1, +5, +10) appear to be different from what I see on the Android Studio layout design, which those button perfectly fit in the layout design but when I run the application on my smartphone, it turns out to be insufficient place for the button and ended up squeezing up the button as well as the text on it. I have tried tons of approach to fix it, including set smaller margin, set smaller button, set smaller text on the button, applying themes on the Internet, however, none of this approach address the issue. After consuming up plenty of time, finally figured out setting the inset smaller so that the text has larger space to fit in the button.

On the other hand, which is the implementation of functional parts, most of them went pretty well especially on the last part where implementing the listener on calculation button, perform the calculation, display the discount information and view model, the process went very smoothly. The major issue that I faced was the bug caused by the bidirectional synchronization of EditText and SeekBar. To illustrate that, since one of the changes on them will reflect to another, take Tax as example, if the user clear the input of Tax, then the SeekBar progress will also sync to its minimum value, however, I have set the minimum value of SeekBar as "1", resulting the SeekBar reflect "1" back to the Tax input field and hence causing the user need to delete twice in order to make the tax input field blank again. I have tried plenty of ways to get rid of this problem, but none of them works, when I was actually going to give up as I have spent too much time on this, I have finally figured out a logic implementation way, which is initializing a isTaxClear Boolean variable as false and only reflect the changes on SeekBar to Tax input field when the isTaxClear



is not true and after that immediately set it back to false, and there is also a assignment on isTaxClear equal to Tax.isBlank() in TextWatcher. Hence, it makes it become when user just started the app without entering the input, thus the Tax input field is blank, however, as the initialization of isTaxClear is false, hence the SeekBar can reflect the progress value to the Tax input field. After that, isTaxClear is set to false. If the user clears the Tax input field, the isTaxClear assignment in TextWatcher will set it to true, thus the SeekBar would not reflect the progress "1" to the Tax input field as isTaxClear is not false and right after that isTaxClear is set back to false and hence resulting if the user drags the SeekBar it will reflect the progress to the Tax input field. Thus, resolving the issue and same goes to Discount SeekBar.

# Assignment 1

## ORIGINALITY REPORT

1 %

SIMILARITY INDEX

1 %

INTERNET SOURCES

0 %

PUBLICATIONS

0 %

STUDENT PAPERS

## PRIMARY SOURCES

1

Submitted to Asia Pacific University College of  
Technology and Innovation (UCTI)

Student Paper

<1 %

2

[tdr.lib.ntu.edu.tw](http://tdr.lib.ntu.edu.tw)

Internet Source

<1 %

Exclude quotes Off

Exclude bibliography Off

Exclude matches Off