DALHOUSIE UNIVERSITY

Faculty of Computer Science

CSCI 5708 Mobile Computing

Assignment 1: *Tip Calculator*

March 12, 2020

Submitted By

Deep Muni, B00828375, dp778309

Table of Contents

List of Figures

List of Tables

1. Introduction

Tip Calculator application will help the user to calculate the tip on the bill amount. The application is simple to use and requires no guidance or help.

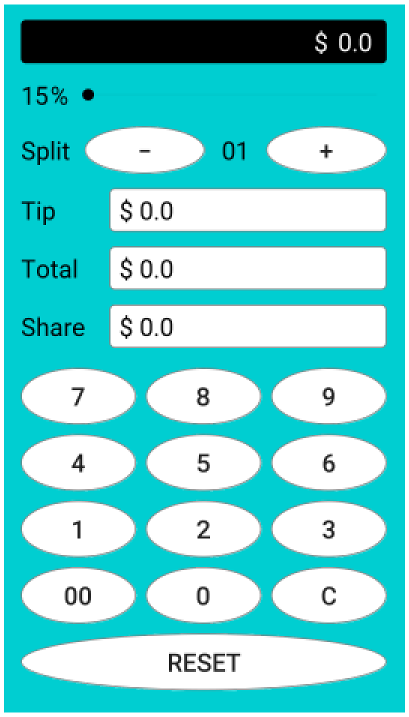


Figure 1 Tip Calculator

2. Unified Modeling Language

In this application, the user needs to input an amount, input the tip percentage and input the party size (optional). The application will calculate the tip amount, the total amount and share per head and show the output to the user on the screen.

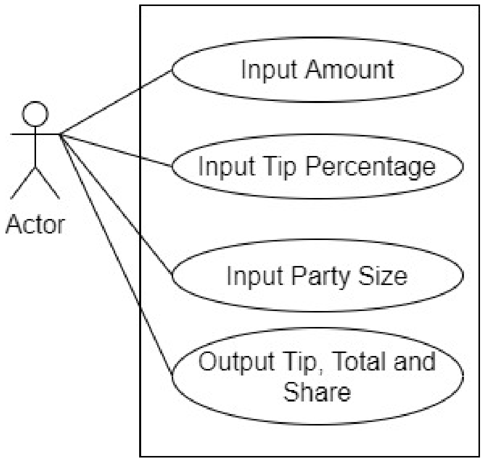


Figure 2 Unified Modeling Language Diagram

3. Tools

Below are the tools used to develop the application.

1. Android Studio [1]

   Design Prototype and Application Development were done using Android Studio

2. Draw.IO [2]

   UML Diagram and Wireframe creation was done using the tool Draw.IO

3. Android Device Vivo V9

   The application was installed on tested on the device [This is my personal device]

4. Implementation

The implementation for the application was carried out in phases. The phases are Wireframing, design prototype, finalizing the design and developing the application

4.1 Wireframe

Wireframing of the application helps to understand the initial design of the application. Using the wireframe, the design components and the application programming component can be figured out.

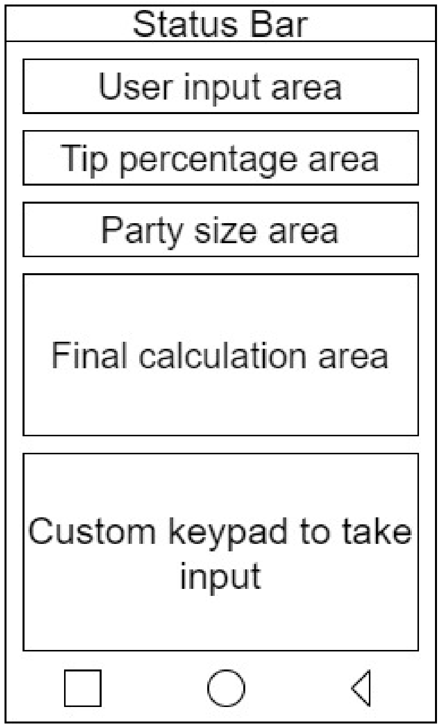


Figure 3 Wireframe

4.2 Design Prototype

The high-fidelity prototype was created in Android Studio using XML. The prototype helps to understand the alignment for each section, the color scheme to be used, the size of each section, etc.

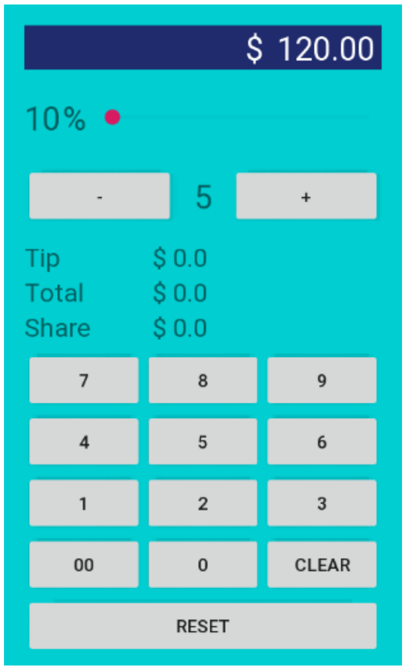


Figure 4 High Fidelity Prototype

4.3 Final Design

In the final design, the accurate color schemes were chosen, the shape of the buttons and text views were changed, the font size was increased and the text colours were changed.

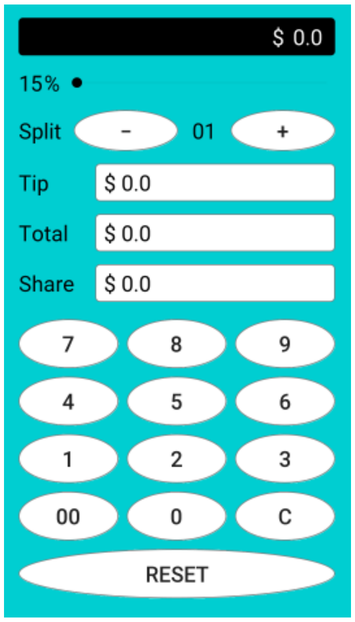


Figure 5 Final Design

4.4 Development

The application programming for the application is done using Kotlin [3].

In this application, an input area is provided where the user can enter the amount of the bill. An option for tip has been provided which has minimum value as 0 and the maximum value is 25. This is controlled using the seek bar. Another option to split the bill is provided. The user can decrement the value using the minus button and increment the value using the plus button. Default values for these three fields are set. The amount will be $ 0.0, the tip will be 15% and the party size is set to an initial value of 1.

In the next section, the user can view the result for these inputs. There are three text views provided where users can see the tip amount payable along with the bill amount. The addition of tip amount and bill amount can be seen in the next field, Total. If the user has company and has entered a value for party size, the per head amount can be seen in the next field, Split.

The next section has the keypad with 0-9 digits button, the button which appends two zeros, a clear button, and a reset button. There is no need for a calculate button because, as and when the user input the amount, change the tip value and change the party size value, the output fields will be updated automatically.

To let the user know that they have pressed a certain button, the system will provide Haptic Feedback [4] to the user. This haptic feedback is provided by a vibration sensation. (This functionality can be tested by running the application on a device).

The user can enter the amount by clicking on the button from 0-9 and the button for entering two zeros. To remove the incorrect value, the user can click the clear button "C". To reset the entire input, the user can click the reset button, which will bring all the values to the default value. To increase or decrease the number of people, the user can click the increment (+) or decrement (-) button. All this is handled using the on click listener.

To display the user regarding the percentage of tip they have entered, there is a text view that shows the current progress of the seek bar. This change of the value and reflecting it to the output is handled using the Seek Bar Change Listener [5].

Toast [6] functionality is used to provide the error message if the user entered characters greater than 9 in the input text area.

Below are the views use to design the layout

Table 1 List of views

Buttons	Increment/Decrement, Clear/Reset, Digit 0 to 9/ Double zero
Text View	Amount, Tip Amount, Total Amount, Per head Amount
Seek Bar	Tip percentage
Label	Split, Tip, Total & Share

5. Nielsen’s Usability Heuristics

While developing the application (designing and programming), I tried to cover some of Nielsen’s Usability Heuristics [7]. Below are the evidences that they are followed in the application.

5.1 Visibility of System Status

The user can see the output of the calculations as soon as they change any input. This is always available on the screen. Also, once the user changes the value on seek bar it is automatically updated in the text field near it to let the user know the percent they are tipping.

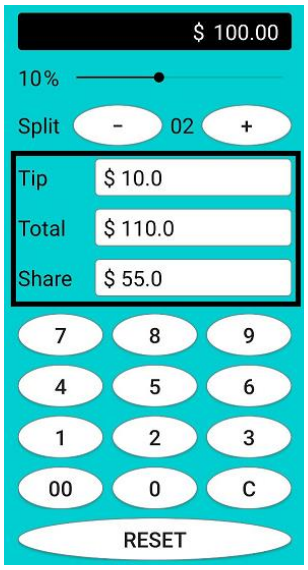


Figure 6 Output is always available on screen

5.2 User Control and Freedom

If the user enters a wrong value or wants to remove the entire input, the application provided them with a clear and reset button. The clear button will remove a single character from the input and the reset button will reset the entire input to the default values.

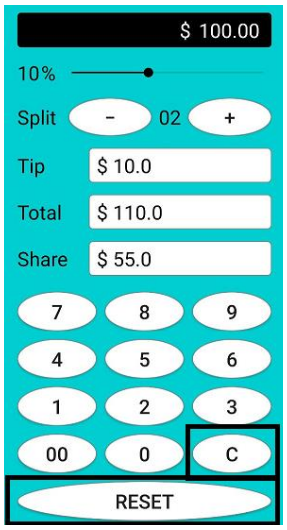
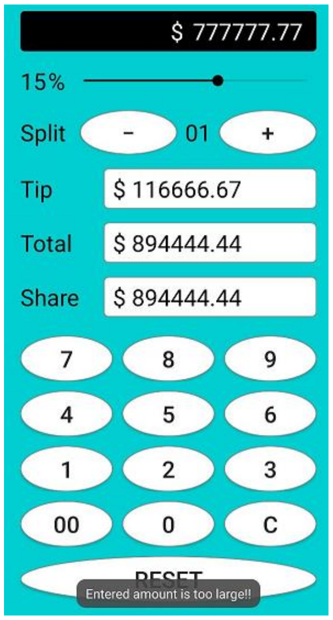


Figure 7 Clear and Reset button

5.3 Help users to recognize, Diagnose, and Recover from errors

There can be a maximum 9 characters (including the decimal) in the amount input area. After this user won't be allowed to enter any more digit and a Toast with message "Entered amount is too large!!" will be displayed. This is done to avoid the error if the number limit is reached for a particular data type.

(This idea was suggested by Prof. Meredith)



Tip	$ 116666.67
Total	$ 894444.44
Share	$ 894444.44

Entered amount is too large!!

Figure 8 Error Message

5.4 Prevent Errors

There are two pieces of evidence for this heuristic.

1. As we are working with monetary values, a mistake in entering the amount may yield incorrect results. For example, 1250.00 instead of 12.50. To avoid this scenario, I have not provided an option for the user to enter decimal points themselves. The user will enter the value starting at the cent's place and once they enter more value, it will shift to the dollar's place.

   For example: Enter $ 12.50

input value "1"	input value "2"	input value "5"	input value "0"
$ 0.01	$ 0.12	$ 1.25	$ 12.50

2. The increment and decrement value for the number of people cannot go below 1. A condition is applied in the programming to prevent this error. As any number below one will yield invalid output. If the number becomes zero, the exception will be raised that division will zero is not possible and if the number becomes negative, the per head price will be negative which is incorrect.

So, this is how I have prevented the errors from occurring in the application.

5.5 Provide Flexibility and Efficiency of Use

To allow user to enter a big number with many zeros, the button “00” will append two zeros together. This is done to provide ease to the user.

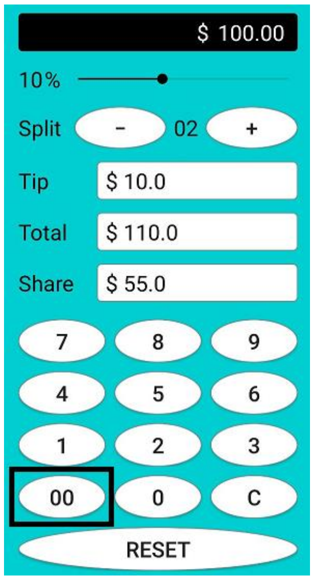


Figure 9 Button to append two zeros

5.6 Focus on Aesthetic and Minimalist Design

To provide the user with a good user interface, only the required sections are inserted. They are properly aligned so that the user does not get confused. The input bar has a black color, which can be noticed as soon as the application is opened. This helps the user understand that the value is to be inserted in that field. For tip percent and the number of people, a different type of input style is being used so that the user does not get confused while they are inserting values using the custom keypad. They can see the update in the output as soon as they change the input, i.e., they don’t need to click on the button such as “calculate” to see the output.

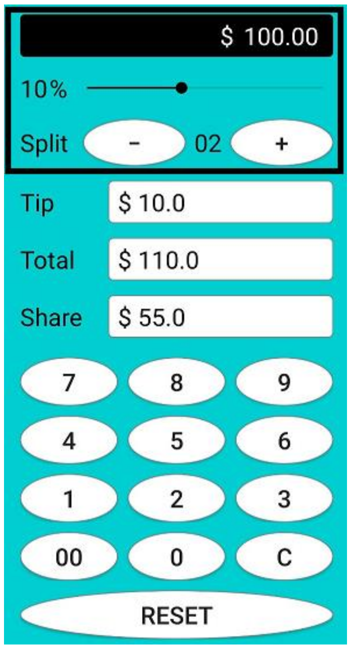


Figure 10 Different style of input

6. Test cases

Below are some of the test scenarios which I tried and resolved using my Android device.

6.1 Test Scenario 1

User is able to remove an incorrect character using the clear button “C”.

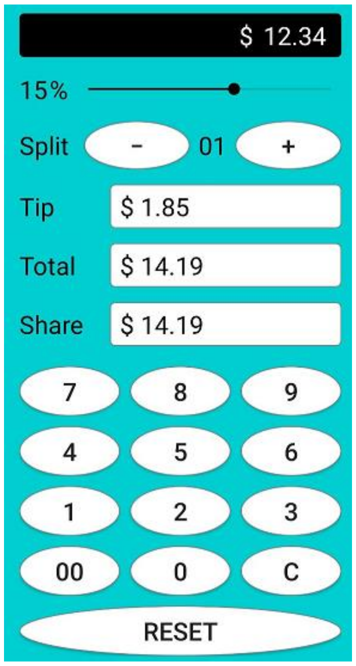

$ 12.34
15%
Split 01
Tip $ 1.85
Total $ 14.19
Share $ 14.19

Figure 11 Incorrect value entered

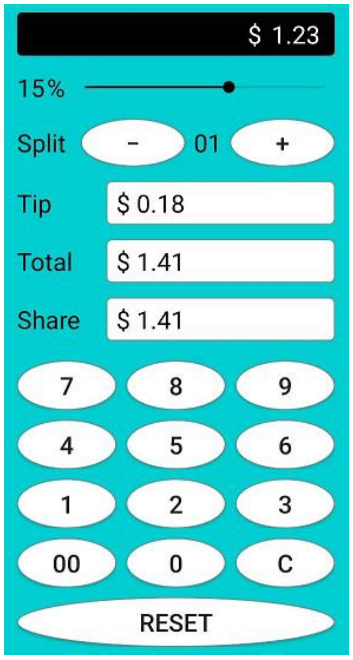

$ 1.23
15%
Split 01
Tip $ 0.18
Total $ 1.41
Share $ 1.41

Figure 12 Incorrect value corrected

6.2 Test Scenario 2

User is able to reset the entire input.

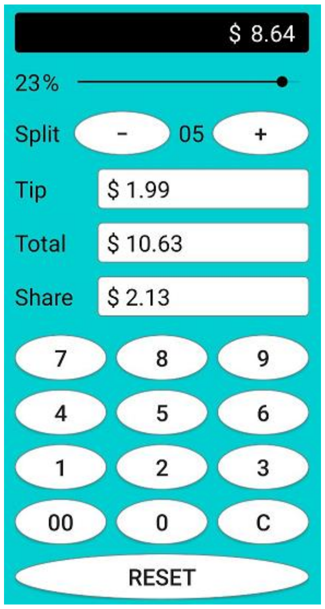

$ 8.64
23%
Split 05
Tip $ 1.99
Total $ 10.63
Share $ 2.13

Figure 13 Entire input needs to be removed

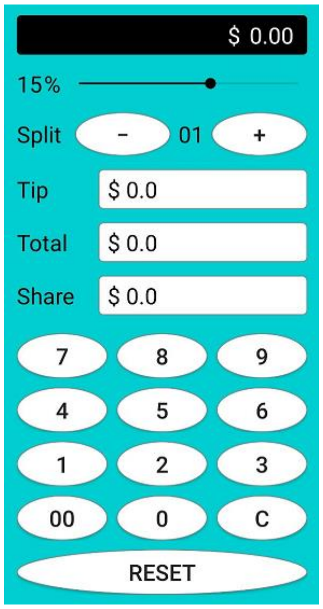
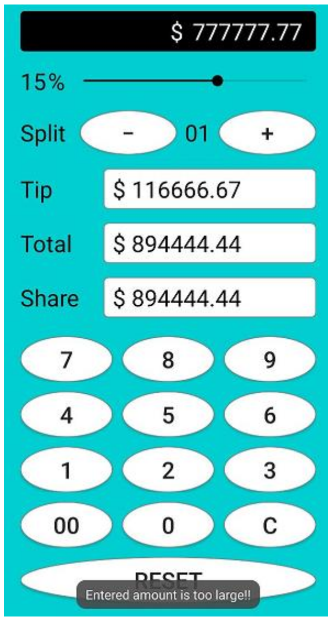

$ 0.00
15%
Split 01
Tip $ 0.0
Total $ 0.0
Share $ 0.0

Figure 14 Entire output reset to default value

6.3 Test Scenario 3

User tries to enter a value which is greater than 9 characters (including the decimal point).



Entered amount is too large!!

Figure 15 Error Message

6.4 Test Scenario 4

Application is able to provide correct output for the given input.

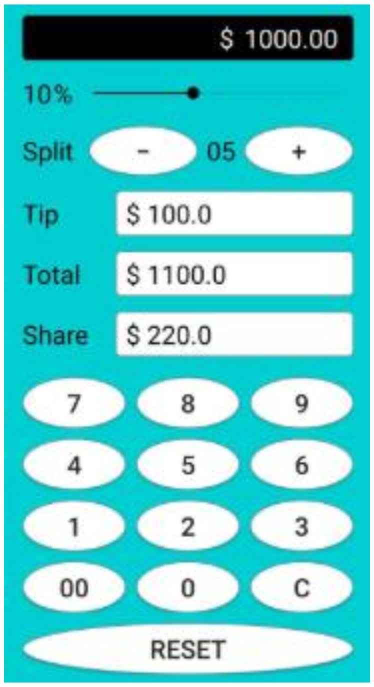


Figure 16 Accurate Result

7. Reference

[1] "Download Android Studio and SDK tools | Android Developers", Android Developers, 2020. [Online]. Available: https://developer.android.com/studio. [Accessed: 08- Mar- 2020]

[2] "Flowchart Maker & Online Diagram Software", Draw.io, 2020. [Online]. Available: https://www.draw.io/. [Accessed: 08- Mar- 2020]

[3] Kotlin, 2020. [Online]. Available: https://kotlinlang.org/. [Accessed: 08- Mar- 2020]

[4] "HapticFeedbackConstants | Android Developers", Android Developers, 2020. [Online]. Available: https://developer.android.com/reference/android/view/HapticFeedbackConstants. [Accessed: 09- Mar- 2020]

[5] "SeekBar | Android Developers", Android Developers, 2020. [Online]. Available: https://developer.android.com/reference/android/widget/SeekBar. [Accessed: 09- Mar- 2020]

[6] "Toasts overview | Android Developers", Android Developers, 2020. [Online]. Available: https://developer.android.com/guide/topics/ui/notifiers/toasts. [Accessed: 10- Mar- 2020]

[7] Neilsen, J. (1994). Enhancing the explanatory power of usability heuristic. In: ACM CHI'94 Conf. pp.152-158.