Table of Contents

I. Introduction 3

II. Exercise 1: 3

1. Fundamental details: 3

2. Exercise answer: 4

2.1. Screen shorts: 4

2.2. Code: 7

III. Exercise 2: 16

1. Fundamental details: 16

2. Exercise answer: 16

2.1. Screen shorts: 16

2.2. Code: 20

IV. Exercise 3: 23

1. Fundamental details: 23

2. Exercise answer: 24

2.1. Screen shorts: 24

2.2. Code: 31

V. Conclusion 44

Figure 1: Calculator 4
Figure 2: click button1 5
Figure 3: click button + ... 5
Figure 4: click button 6 6
Figure 5: click button= 6
Figure 6: click button C 7
Figure 7: Show image 17
Figure 8: press next 18
Figure 9: Press Prev 19
Figure 10: 10 pictures 20
Figure 11: Add Date 26
Figure 12: lack of information 26
Figure 13: pick picture 27
Figure 14: Saved 29
Figure 15: list Contact 31
Figure 16: Deleted 31

Table 1: Excercise 1 3
Table 2: Excercise 2 16
Table 3: Excercise 3 23

I. Introduction:

This logbook report is an important document, recording my journey in a process of building 3 applications according to each application's requirements. The main goal of the logbook is to evaluate and record detailed results. In this report, we will look in more detail at the logbook implementation process and how it has helped me track progress, self-assess, and learn along the way.

II. Exercise 1:

1. Fundamental details:

Student name	Nguyen Duc Minh
Who did you work with?	ID: 001357686
Which Exercise is this?	Create a Calculator application. Contains 4 operators Add, Subtract, Multiply and Divide for two operands, i.e. calculate between two numbers at the same time.
How well are you able to complete your assignments?	I completed all the requested tasks.
Explain details	I have completed the criteria of exercise 1 and have not created or added any additional requirements. I think I can do a better job with the user interface design.

Table 1: Excercise 1

2. Exercise answer:

2.1. Screen shorts:

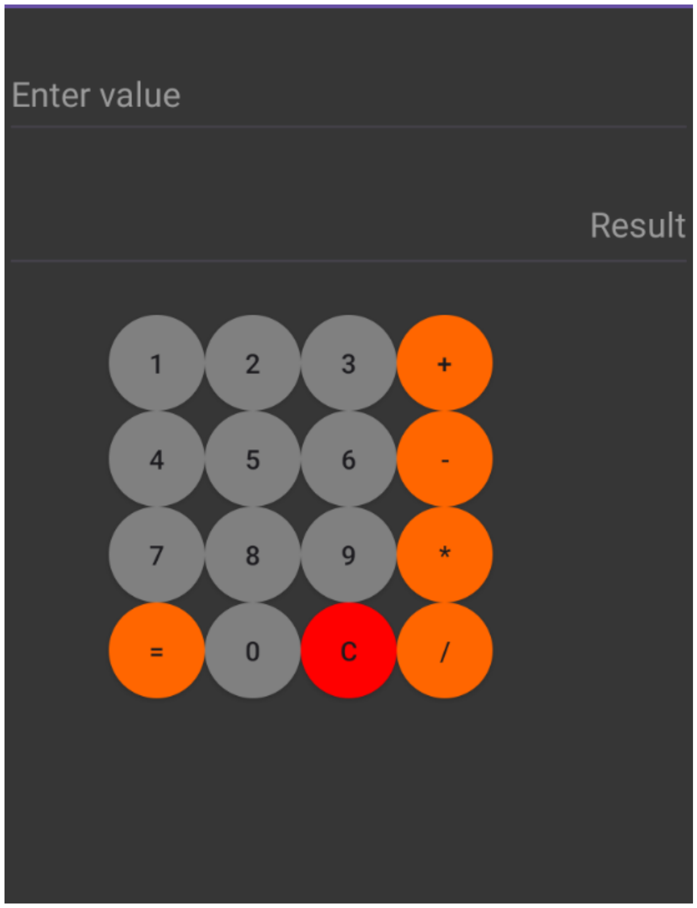


Figure 1: Calculator

This is the main screen of the Calculator application. Includes gray number buttons from 0 to 9 and orange calculation buttons Add (+), Subtract (-), Multiply (*), Divide (/) and Equal and a red Clear (C) button. Above are 2 lines of Text to display the Input (Enter value) and Output (Result).

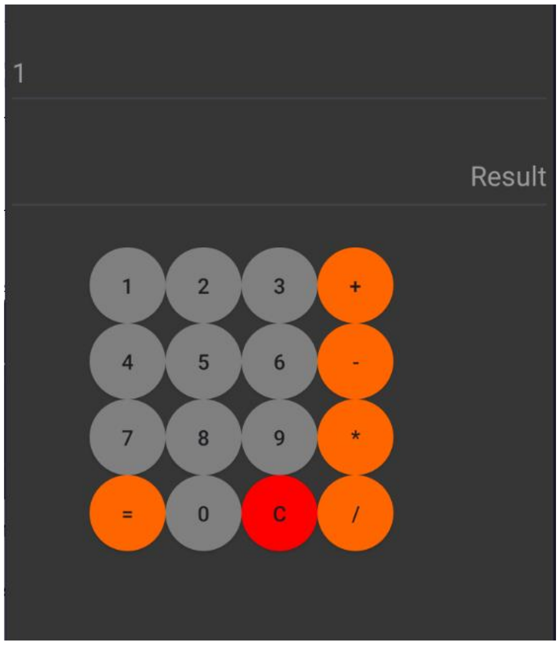


Figure 2: click button1

The first line changes when I press buttons 0-9 (I pressed 1).

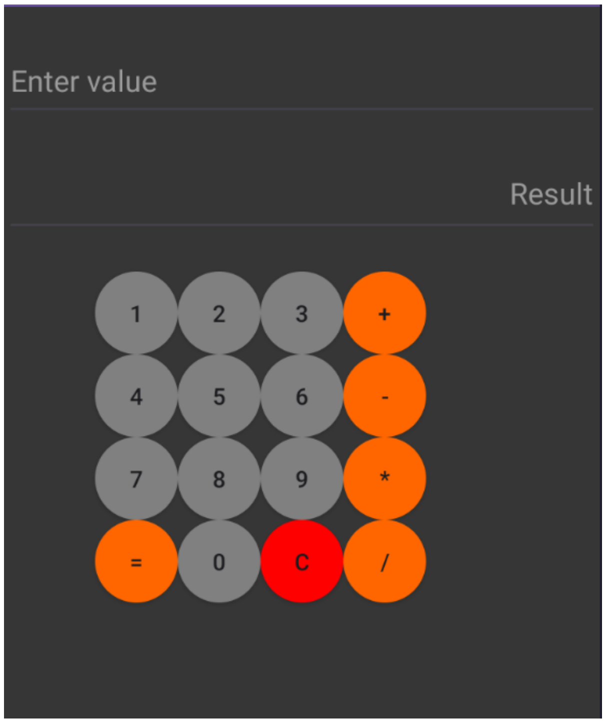


Figure 3: click button +

The above value will be hidden and will be saved immediately after I press the Add (+), Subtract (-), Multiply (*), Divide (/) buttons. (I pressed the + button).

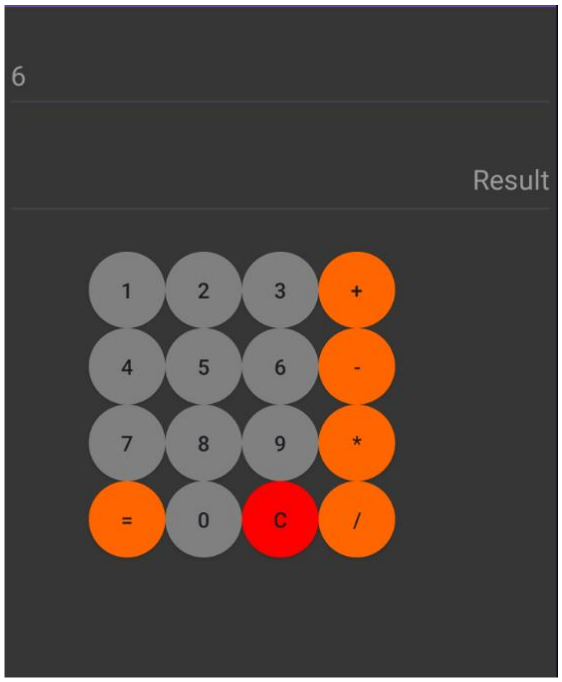


Figure 4: click button 6

Enter the second value to perform the calculation (I pressed button 6).

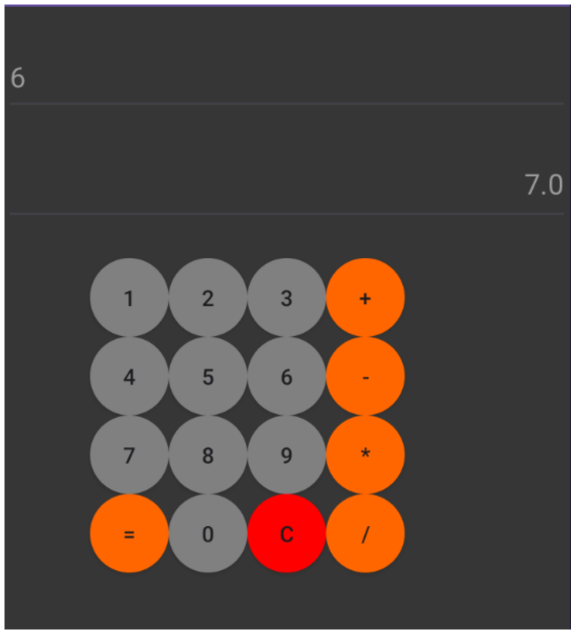


Figure 5: click button=

When you press the equal sign, the test results will be displayed in the Result line and on the right (calculation: 1+6 = 7)

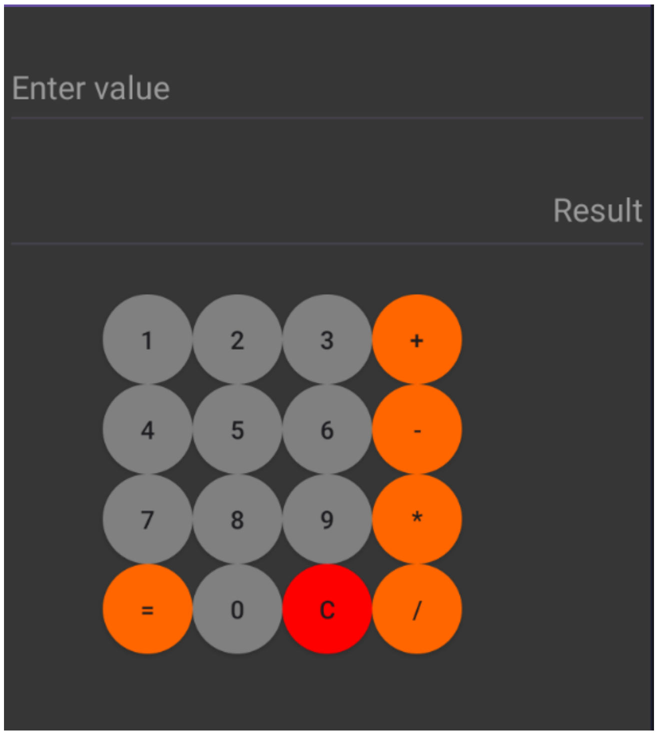


Figure 6: click button C

When pressing the Clear(C) button, the Enter value and Result lines are cleared.

2.2. Code:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/btn_normal"
android:state_pressed="false"/>
    <item android:drawable="@drawable/btn_pressed"
android:state_pressed="true"/>
-----------------------------------------------------------------------------
</selector>
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>

    <solid android:color="#808080"/>
</shape>
-----------------------------------------------------------------------------

<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>

    <solid android:color="#696969"/>
</shape>
```

The code has the effect of changing the color of the 0-9 buttons from #808080 to #696969 when clicked.

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/btn_c_normal"
android:state_pressed="false"/>
    <item android:drawable="@drawable/btn_c_pressed"
android:state_pressed="true"/>
</selector>
-----------------------------------------------------------------------------
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>
<solid android:color="#FF0000"/>
</shape>
-----------------------------------------------------------------------------

<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>
    <solid android:color="#BB0000"/>
</shape>
```

The code has the effect of changing the color of the Clear button (C) from #FF0000 to #BB0000 when clicked.

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/btn_equal_normal"
android:state_pressed="false"/>
    <item android:drawable="@drawable/btn_equal_pressed"
android:state_pressed="true"/>
</selector>
-----------------------------------------------------------------------------

<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>

    <solid android:color="#FF6600"/>
</shape>
-----------------------------------------------------------------------------

<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>

    <solid android:color="#FF6633"/>
</shape>
```

The code changes the color of the Equal(=) button from #FF6600 to #FF6633 when clicked.

```
<RelativeLayou
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#363636">
```

I use RelativeLayout because it is a group of view elements that show child view elements in relative positions. The layout_with and layout height attributes I set to match_parent with the aim of filling the screen. and I changed the background color to #363636.

```
<EditText
    android:id="@+id/edt_value"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="20dp"
    android:textColorHint="#9C9C9C"
    android:hint="Enter value"
    android:inputType="number"
    android:textColor="#9C9C9C" />
```

I use EditText so the user can enter a value. id is edt1, width is full screen, height is 50dp and distance from Top is 20dp. hint: Enter value's purpose is to show the user the input location. When entering, the user can only enter numbers(input Type="number"). When entering, the word Enter value sec will be lost. And I chose the font color as #9C9C9C.

```
<EditText
    android:id="@+id/edt_result"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_below="@+id/edt_value"
    android:layout_marginTop="20dp"
    android:gravity="right"
    android:textColorHint="#9C9C9C"
    android:hint="Result"
    android:textColor="#9C9C9C" />
```

The Result line has ID: edt2, the width is full screen, the height is 50dp and the Top is 20dp. hint: Result is intended to show the user where to display the results. And I chose the font color as #9C9C9C.

```
<android.widget.Button
    android:id="@+id/btn_1"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/edt_result"
    android:layout_alignEnd="@+id/btn_4"
    android:layout_alignRight="@+id/btn_4"
    android:layout_marginTop="20dp"
    android:text="1" />
```

For button 1, I set the id to button1 and the background image I took from the button_bg file. I set the height and width to 50dp combined with the property radius=30dp (rounded 4 corners), so my pen has a circular shape. and I placed it below the Result line (layout_below...) and 20dp away from the Result line.

two lines layout_alignEnd, layout_alignRight to position that button. Similar to button 1, the remaining buttons all have different shapes and colors, only the position, id and text are different.

```xml
<android.widget.Button
    android:id="@+id/btn_2"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/btn_1"
    android:layout_toStartOf="@+id/btn_3"
    android:layout_toLeftOf="@+id/btn_3"
    android:text="2" />

<android.widget.Button
    android:id="@+id/btn_3"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/btn_2"
    android:layout_centerHorizontal="true"
    android:text="3" />

<android.widget.Button
    android:id="@+id/btn_4"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_1"
    android:layout_toLeftOf="@+id/btn_2"
    android:text="4" />

<android.widget.Button
    android:id="@+id/btn_5"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignStart="@+id/btn_2"
    android:layout_alignLeft="@+id/btn_2"
    android:layout_alignBottom="@+id/btn_4"
    android:text="5" />

<android.widget.Button
    android:id="@+id/btn_6"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_3"
    android:layout_alignStart="@+id/btn_3"
    android:layout_alignLeft="@+id/btn_3"
    android:text="6" />

<android.widget.Button
    android:id="@+id/btn_7"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_4"
    android:layout_toLeftOf="@+id/btn_2"
    android:text="7" />

<android.widget.Button
    android:id="@+id/btn_8"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_5"
    android:layout_alignStart="@+id/btn_5"
    android:layout_alignLeft="@+id/btn_5"
    android:text="8" />

<android.widget.Button
    android:id="@+id/btn_9"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_6"
    android:layout_alignStart="@+id/btn_6"
    android:layout_alignLeft="@+id/btn_6"
    android:text="9" />

<android.widget.Button
    android:id="@+id/btn_plus"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/btn_3"
    android:layout_toRightOf="@+id/btn_3"
    android:text="+" />

<android.widget.Button
    android:id="@+id/btn_sub"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_plus"
    android:layout_alignStart="@+id/btn_plus"
    android:layout_alignLeft="@+id/btn_plus"
    android:layout_alignEnd="@+id/btn_plus"
    android:layout_alignRight="@+id/btn_plus"
    android:text="-" />

<android.widget.Button
    android:id="@+id/btn_mul"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_sub"
    android:layout_alignStart="@+id/btn_sub"
    android:layout_alignLeft="@+id/btn_sub"
    android:text="*" />

<android.widget.Button
```

```
    android:id="@+id/btn_eql"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_7"
    android:layout_toLeftOf="@+id/btn_2"
    android:text="=" />

<android.widget.Button
    android:id="@+id/btn_0"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_8"
    android:layout_alignStart="@+id/btn_8"
    android:layout_alignLeft="@+id/btn_8"
    android:text="0" />

<android.widget.Button
    android:id="@+id/btn_C"
    android:background="@drawable/btn_c"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_9"
    android:layout_alignStart="@+id/btn_9"
    android:layout_alignLeft="@+id/btn_9"
    android:text="C" />

<android.widget.Button
    android:id="@+id/btn_div"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/btn_mul"
    android:layout_alignStart="@+id/btn_mul"
    android:layout_alignLeft="@+id/btn_mul"
    android:layout_alignRight="@+id/btn_mul"
    android:text="/" />
```

```
Button btn0, btn1, btn2, btn3, btn4, btn5, btn6,
        btn7, btn8, btn9, btnAdd, btnSub, btnDiv,
        btnMul, btnC, btnEqual;
EditText resultEdt;
EditText valueEdt;
float value_1, value_2;
boolean isAddition, isSubtraction, isMultiplication, isDivision;
```

I declare the Button, EditTexts, 2 values to save when the user enters with float data type and data type boolean for isAddition, isSubtraction, isMultiplication, isDivision

```
btn0 = findViewById(R.id.btn_0);
btn1 = findViewById(R.id.btn_1);
btn2 = findViewById(R.id.btn_2);
btn3 = findViewById(R.id.btn_3);
btn4 = findViewById(R.id.btn_4);
btn5 = findViewById(R.id.btn_5);
btn6 = findViewById(R.id.btn_6);
btn7 = findViewById(R.id.btn_7);
btn8 = findViewById(R.id.btn_8);
btn9 = findViewById(R.id.btn_9);
btnAdd = findViewById(R.id.btn_plus);
btnSub = findViewById(R.id.btn_sub);
btnMul = findViewById(R.id.btn_mul);
btnDiv = findViewById(R.id.btn_div);
btnC = findViewById(R.id.btn_C);
btnEqual = findViewById(R.id.btn_eql);
valueEdt = findViewById(R.id.edt_value);
resultEdt = findViewById(R.id.edt_result);
```
Map to layout side via id.

```
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "1");
    }
});
btn2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "2");
    }
});
btn3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "3");
    }
});
btn4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "4");
    }
});
btn5.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "5");
    }
});
btn6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "6");
    }
});
```

```
btn7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "7");
    }
});
btn8.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "8");
    }
});
btn9.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "9");
    }
});
btn0.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "0");
    }
});
```

Catch events when the user clicks buttons 0-9 (setOnClickListener). The Enter value line will change the value (valueEdt.setText()) according to the button value (valueEdt getText);

```
btnAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        value_1 = Float.parseFloat(valueEdt.getText() + "");
        isAddition = true;
        valueEdt.setText(null);
    }
});
btnSub.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        value_1 = Float.parseFloat(valueEdt.getText() + "");
        isSubtraction = true;
        valueEdt.setText(null);

    }
});
btnMul.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        value_1 = Float.parseFloat(valueEdt.getText() + "");
        isMultiplication = true;
        valueEdt.setText(null);

    }
});
btnDiv.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        value_1 = Float.parseFloat(valueEdt.getText() + "");
        isDivision = true;
        valueEdt.setText(null);

    }
});
```

When you click calculations, the value just entered will be saved in the mValueOne variable. The Enter value line will prevent the user from entering the second value (valueEdt setText).

```
btnEqual.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        value_2 = Float.parseFloat(valueEdt.getText() + "");
        if (isAddition) {
            resultEdt.setText(value_1 + value_2 + "");
            isAddition = false;
        }
        if (isSubtraction) {
            resultEdt.setText(value_1 - value_2 + "");
            isSubtraction = false;
        }
        if (isMultiplication) {
            resultEdt.setText(value_1 * value_2 + "");
            isMultiplication = false;
        }
        if (isDivision) {
            resultEdt.setText(value_1 / value_2 + "");
            isDivision = false;
        }
    }
});
```

When you press the = button, the value just entered will be saved in the mValueTwo variable. The Result line will display the calculation of both saved values.

```
btnC.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText("");
        resultEdt.setText("");
    }
});
```

When pressing button C, the Enter value and Result lines will delete the currently displayed values (setText(" ")).

III. Exercise 2:

1. Fundamental details:

Student name	Nguyen Duc Minh
Who did you work with?	ID: 001357686
Which Exercise is this?	Create an App that has a user interface showing one image at a time and two buttons to display images forward and backward. Images are stored in Android Resource.
How well are you able to complete your assignments?	I completed all the requested tasks.
Explain details	I completed all the criteria of exercise 2 meticulously, followed all instructions and produced a finished product. I had no further requests and made no further changes or additions.

Table 2: Excercise 2

2. Exercise answer:

2.1. Screen shorts:

The image will be displayed on the screen and the Next and Prev buttons will be used to view the next and previous images. Picture number 35.

When I click the Next button, the next image will display up to the last image, then clicking Next will return to the first image. Picture number 36.

When I press the Prev button, the previous image will display first. Then pressing Prev will display the last image. Picture number 37.

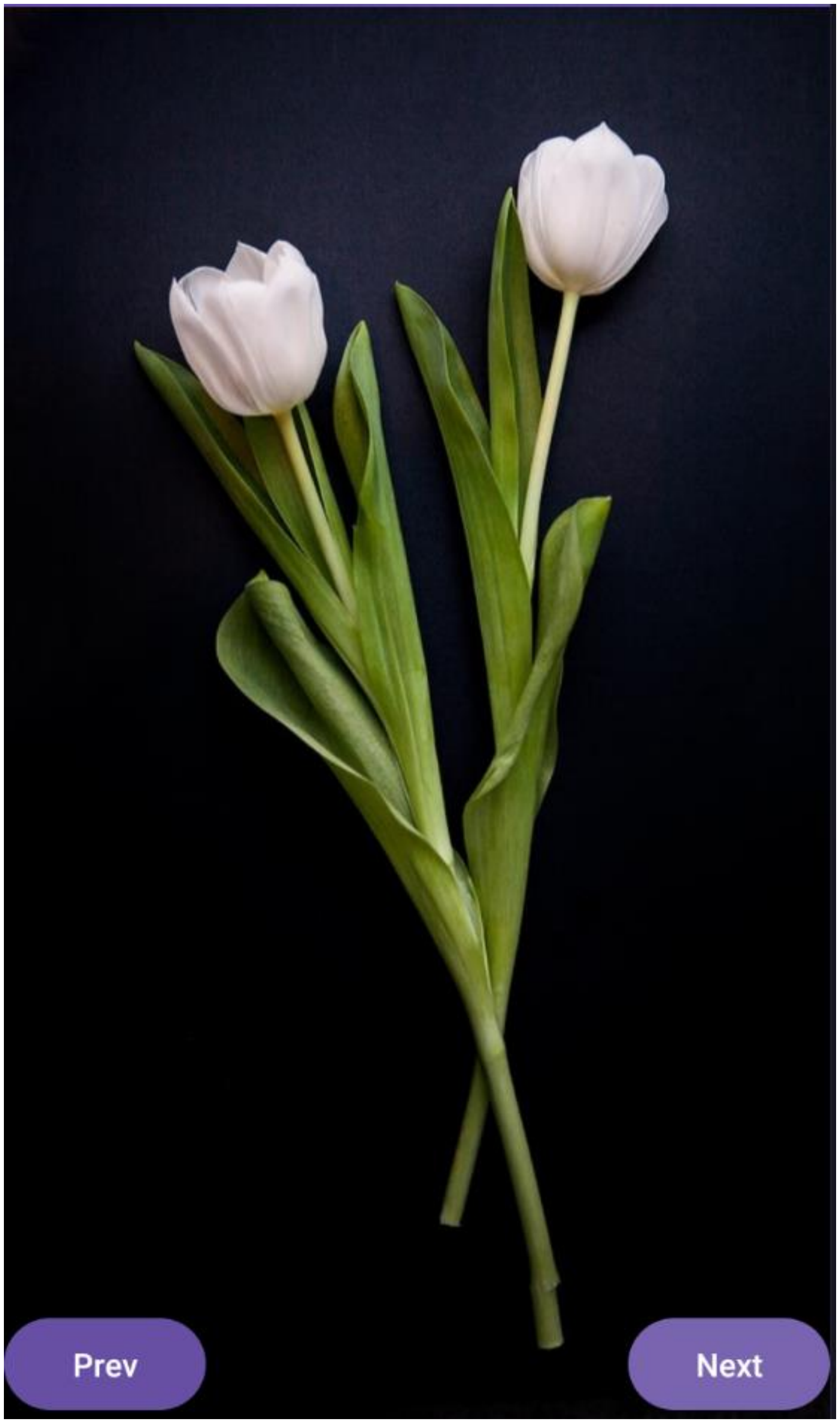


Figure 7: Show image

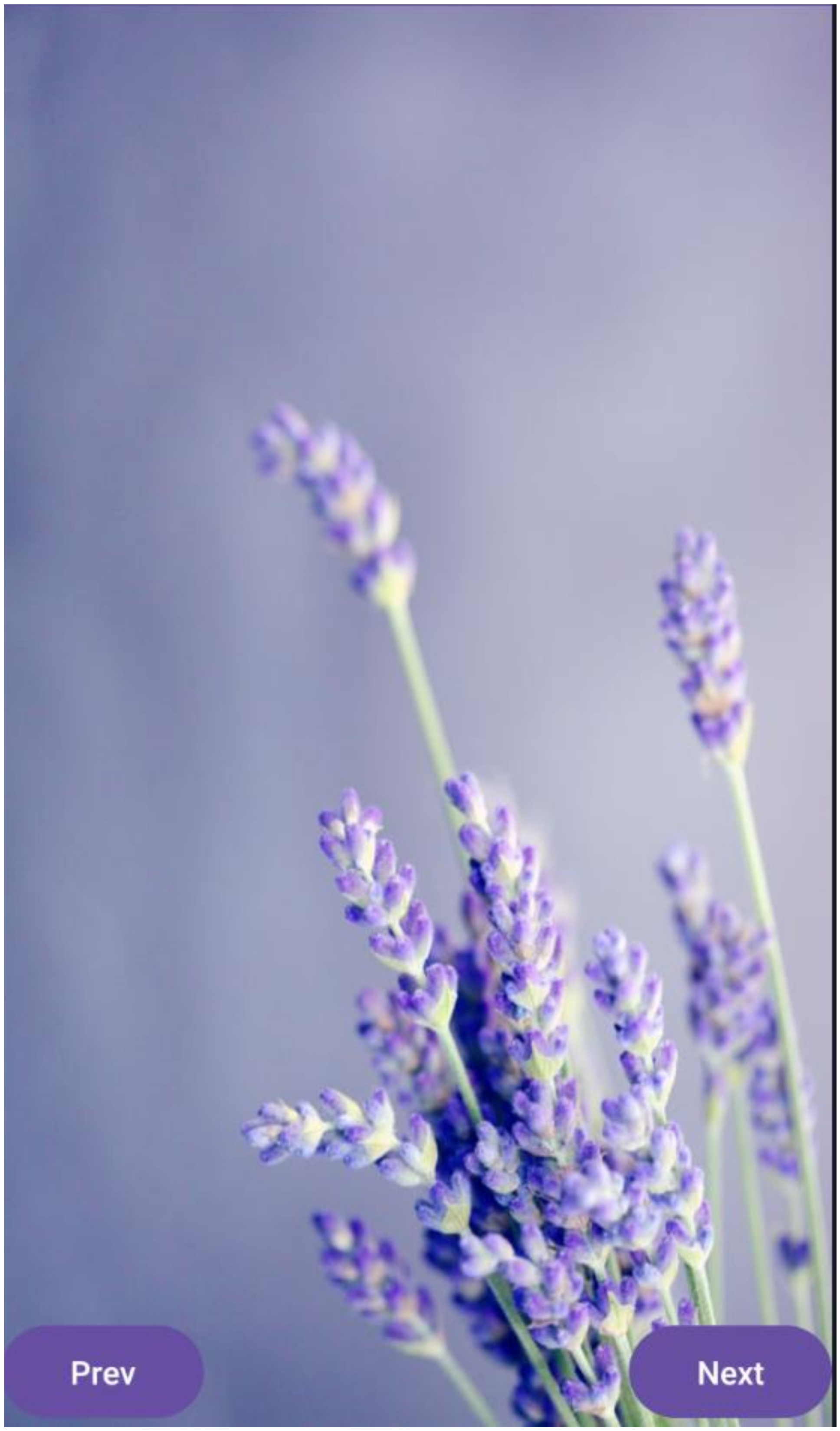


Figure 8: press next

Figure 9: Press Prev

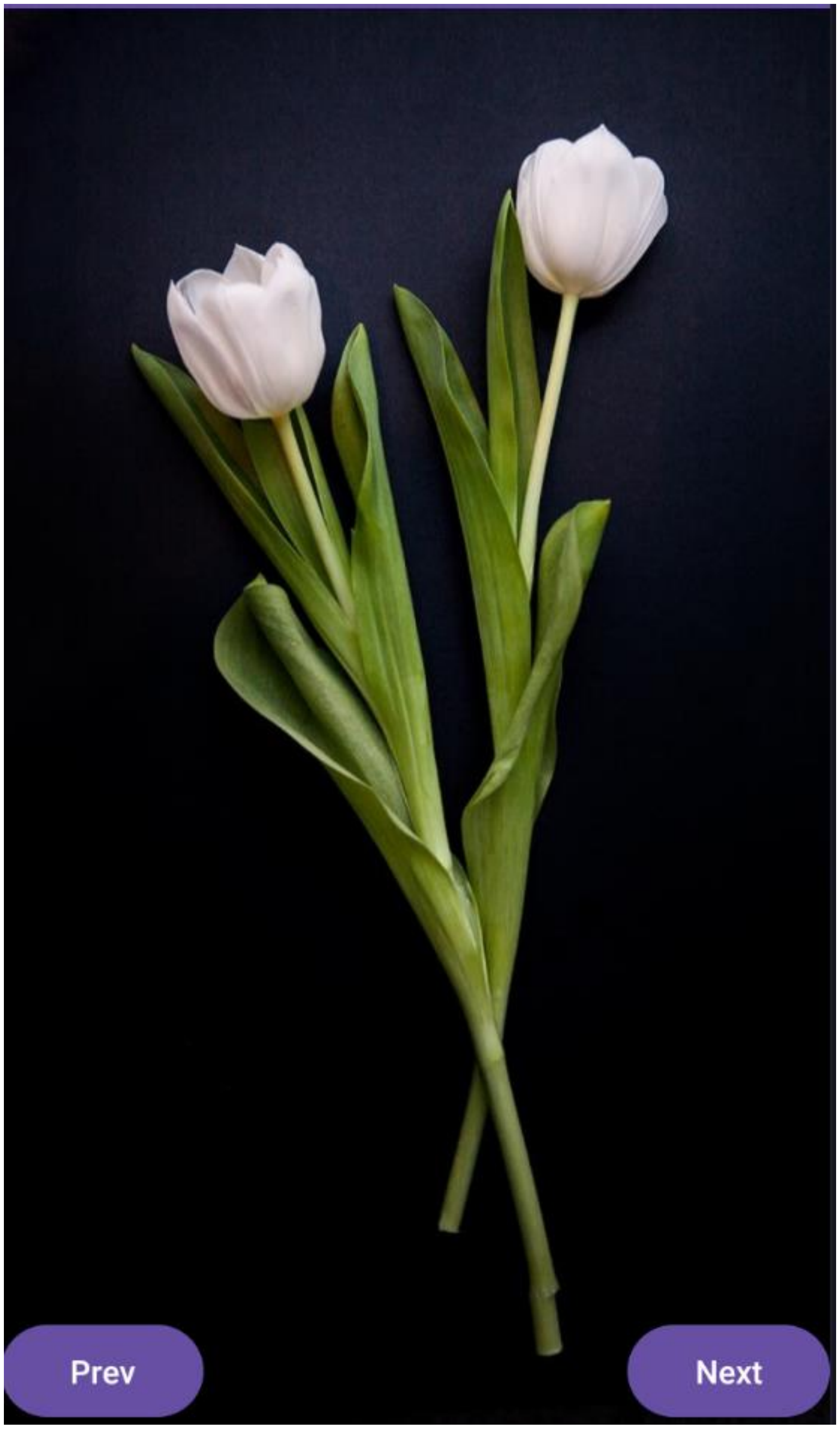
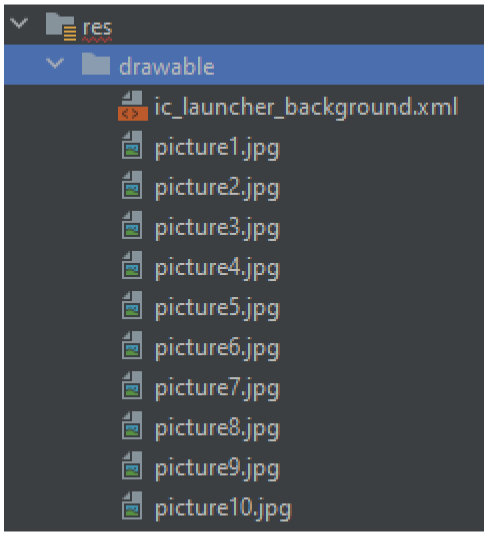


Figure 9: Press Prev

2.2. Code:



```
res
  drawable
    ic_launcher_background.xml
    picture1.jpg
    picture2.jpg
    picture3.jpg
    picture4.jpg
    picture5.jpg
    picture6.jpg
    picture7.jpg
    picture8.jpg
    picture9.jpg
    picture10.jpg
```

Figure 10: 10 pictures

I save 10 pictures into drawable file

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

I use RelativeLayout because it is a group of view elements that show child view elements in relative positions. The layout_with and layout height attributes I set to match_parent with the aim of filling the screen.

```
<ViewFlipper
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/viewFlipper">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView"
        android:src="@drawable/picture1"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView2"
        android:src="@drawable/picture2"/>
```

```xml
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView3"
        android:src="@drawable/picture3"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView4"
        android:src="@drawable/picture4"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView5"
        android:src="@drawable/picture5"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView6"
        android:src="@drawable/picture6"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView7"
        android:src="@drawable/picture7"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView8"
        android:src="@drawable/picture8"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView9"
        android:src="@drawable/picture9"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView10"
        android:src="@drawable/picture10"/>

</ViewFlipper>
```

I use ViewFlipper because it is a type of Android widget used in Android app development to create simple, interactive slideshows or image galleries. For each ImageView I link an image (scr="@drawable/picture...").

```
<Button
    android:id="@+id/next"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Next"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"/>
<Button
    android:id="@+id/previous"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Prev"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_alignTop="@+id/next"/>
```

I created 2 buttons Next and Prev. Their position is determined based on the "align" attribute.

```
ViewFlipper viewFlipper;
Button next;
Button previous;
```

I declare ViewFlipper and 2 buttons next and previous.

```
viewFlipper = findViewById(R.id.viewFlipper);
next =  findViewById(R.id.next);
previous =   findViewById(R.id.previous);
```

I map with layout through id.

```
next.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        viewFlipper.showNext();
    }
});

previous.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        viewFlipper.showPrevious();
    }
});
```

I catch events when the user clicks on 2 buttons. With the Next button I call ViewFlipper's showNext() function. Similar to the Next button, the Prev button calls the showPrevious function.

IV. Exercise 3:

1. Fundamental details:

Student name	Nguyen Duc Minh
Who did you work with?	ID: 001357686
Which Exercise is this?	Extend the ContactDatabase App developed to allow users choose an avatar/profile images for each contact. Those avatars/ profile images can be maintained in the Android resources.
How well are you able to complete your assignments?	I completed all the requested tasks.
Explain details	I have completed the criteria of exercise 3 and have not created or added any additional requirements. I think I can do a better job with the user interface design.

Table 3: Excercise 3

2. Exercise answer:

2.1. Screen shorts:

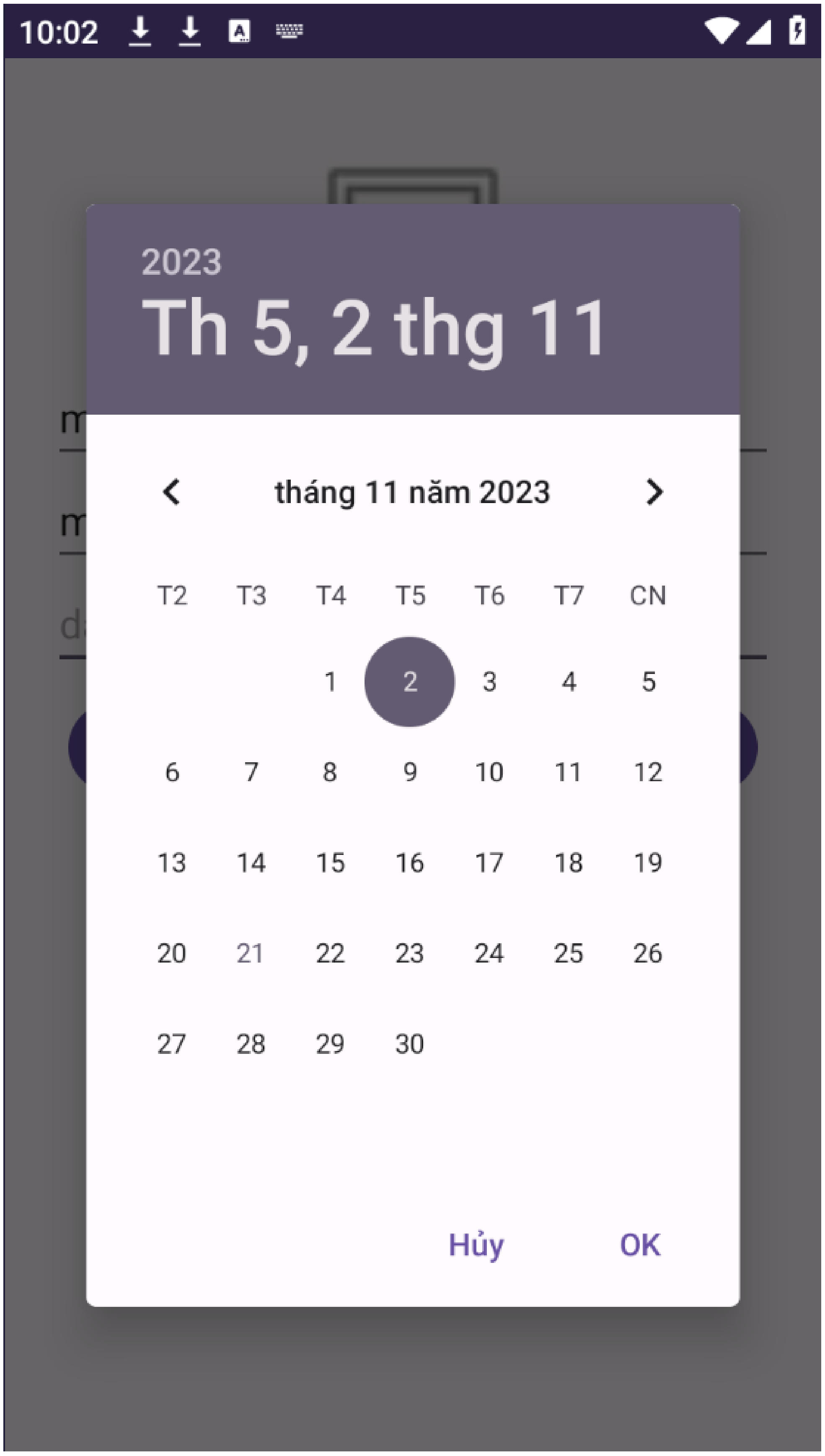
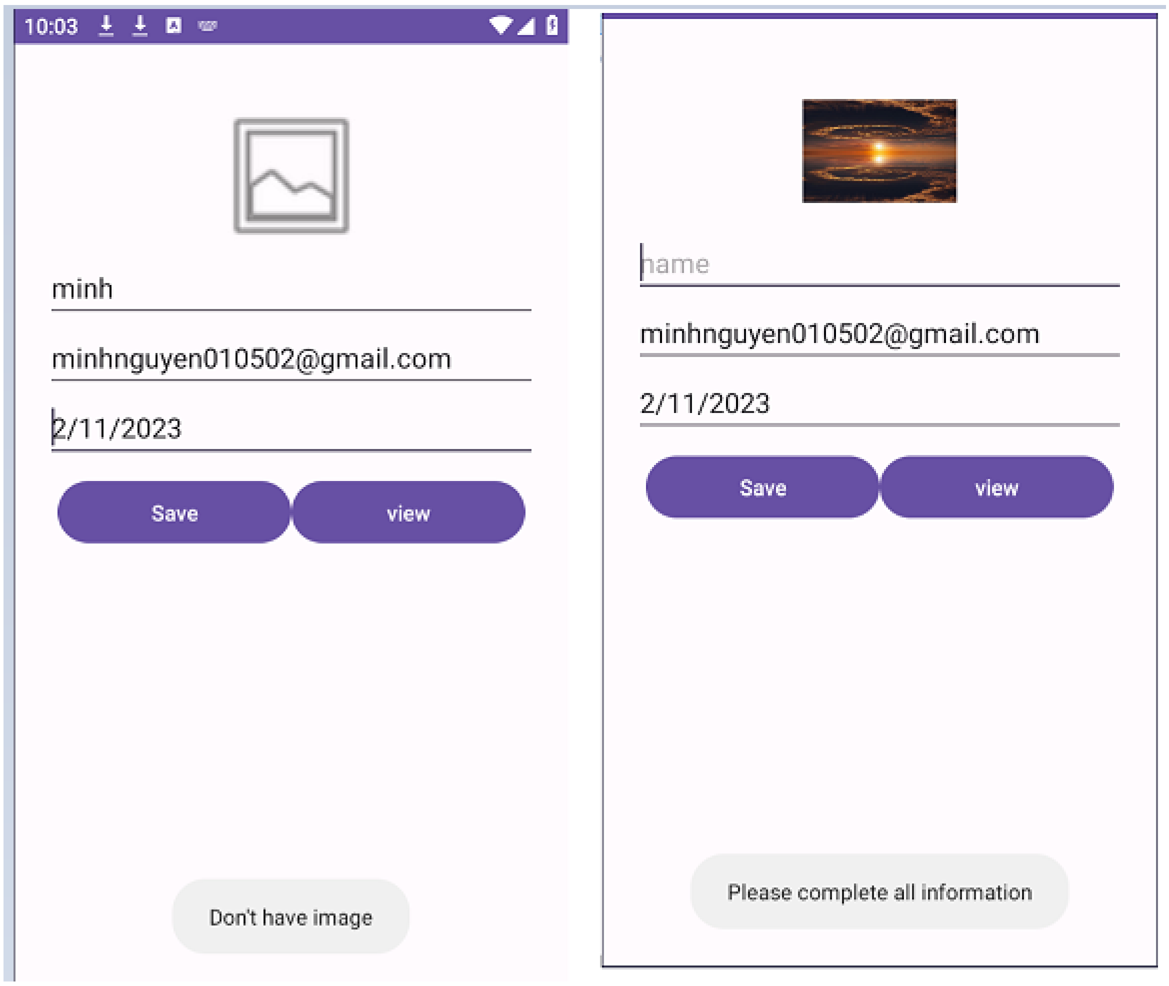

10:02
2023
Th 5, 2 thg 11
tháng 11 năm 2023
T2
T3
T4
T5
T6
T7
CN
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Hủy
OK

Figure 11: Add Date

When clicking on the Date input box, the application displays a date selection panel to avoid users entering incorrectly.

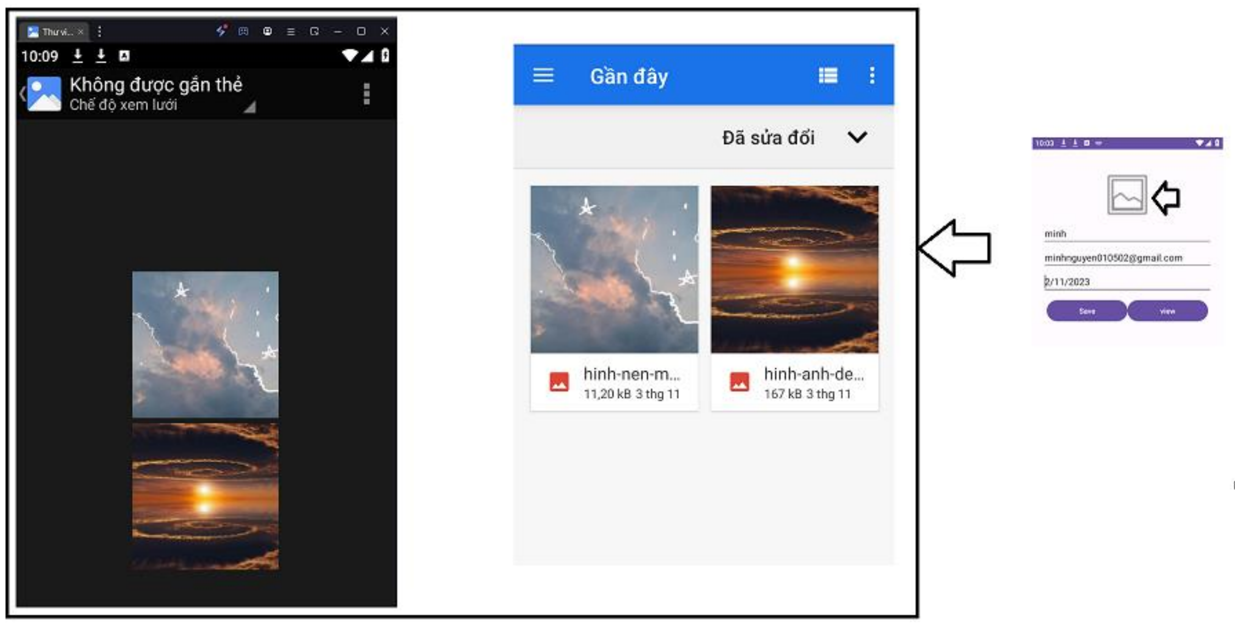


10:03
minh
minhnguyen010502@gmail.com
2/11/2023
Save
view
Don't have image
name
minhnguyen010502@gmail.com
2/11/2023
Save
view
Please complete all information

Figure 12: lack of information

The user clicks on the top image to select a photo in the device library (Figure 13). If the user does not select a photo and clicks Save, the application will notify "Don't have image"

If the user enters missing information, the application will notify "Please complete all information" and will not save. When the user fills in all the information and has selected a photo, click the Save button to save (Figure 14).

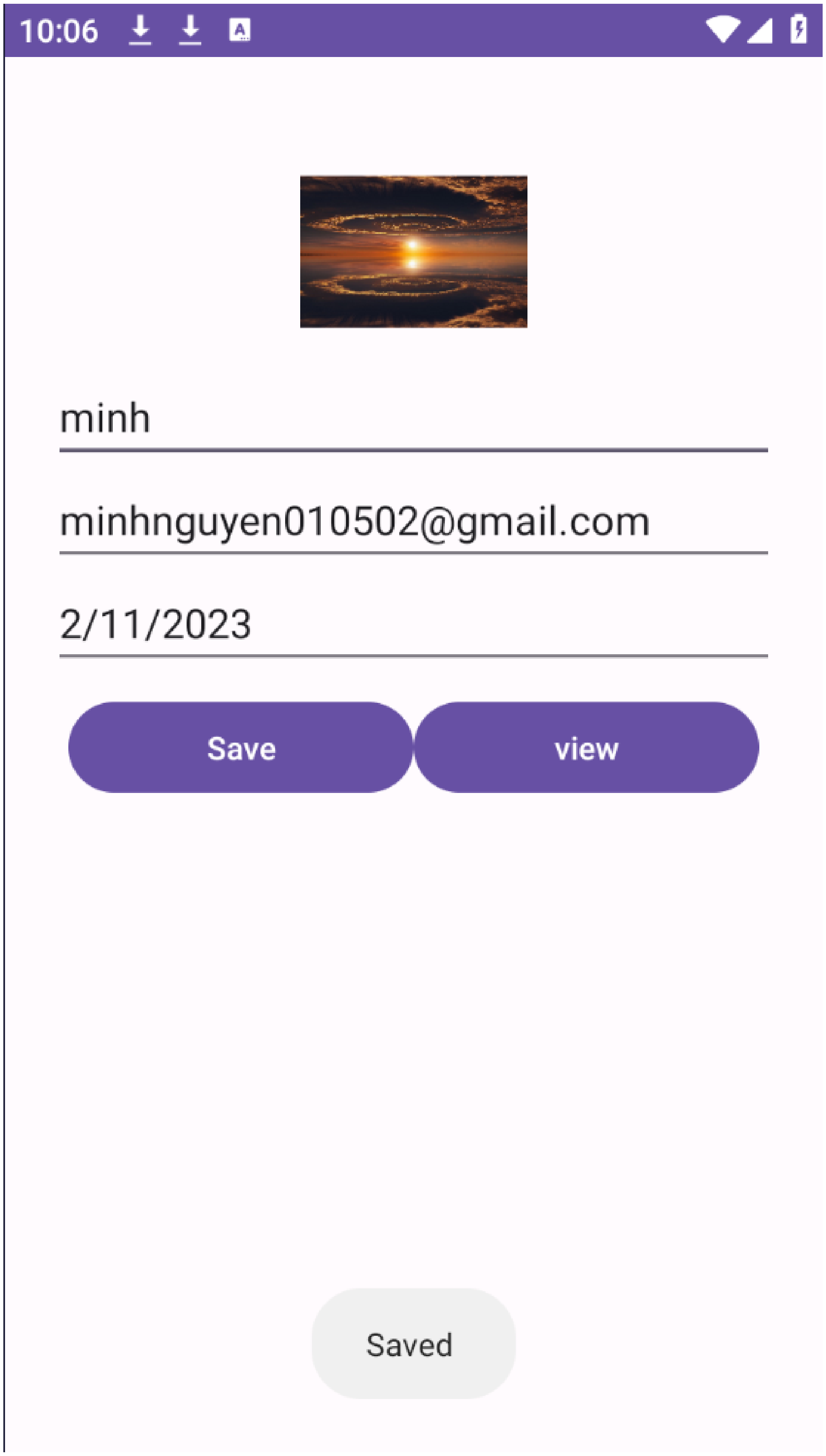


minh

minhnguyen010502@gmail.com

2/11/2023

Figure 13: pick picture



10:06

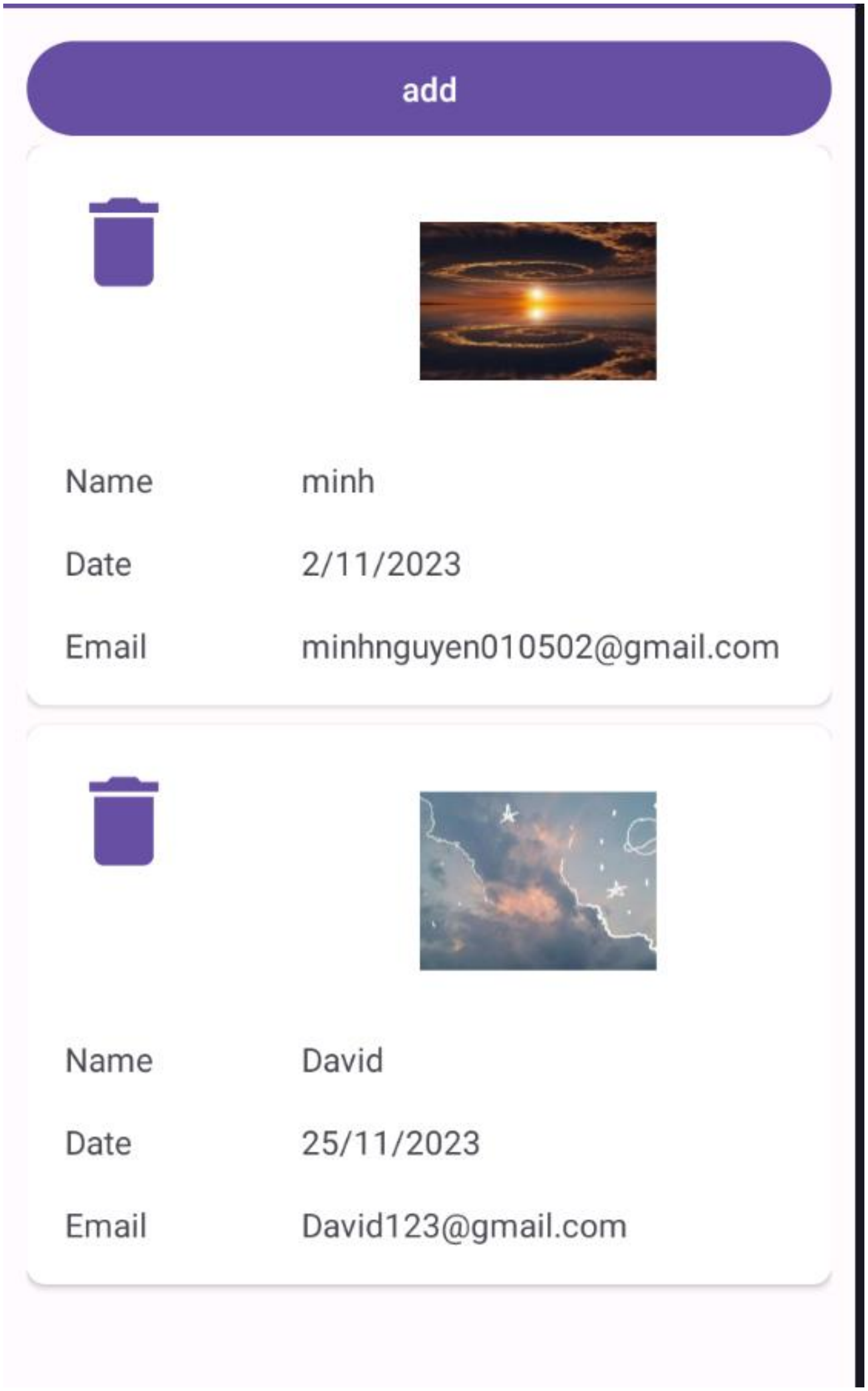
minh

minhnguyen010502@gmail.com

2/11/2023

Saved

Figure 14: Saved



Name	minh
Date	2/11/2023
Email	minhnguyen010502@gmail.com

Name	David
Date	25/11/2023
Email	David123@gmail.com

Figure 15: list Contact

The list will appear first when opening the application and when the user clicks View on the Add page. Users can view information for each Contact and delete Contact by clicking on the trash can icon in the left corner of each contact (Figure 16).

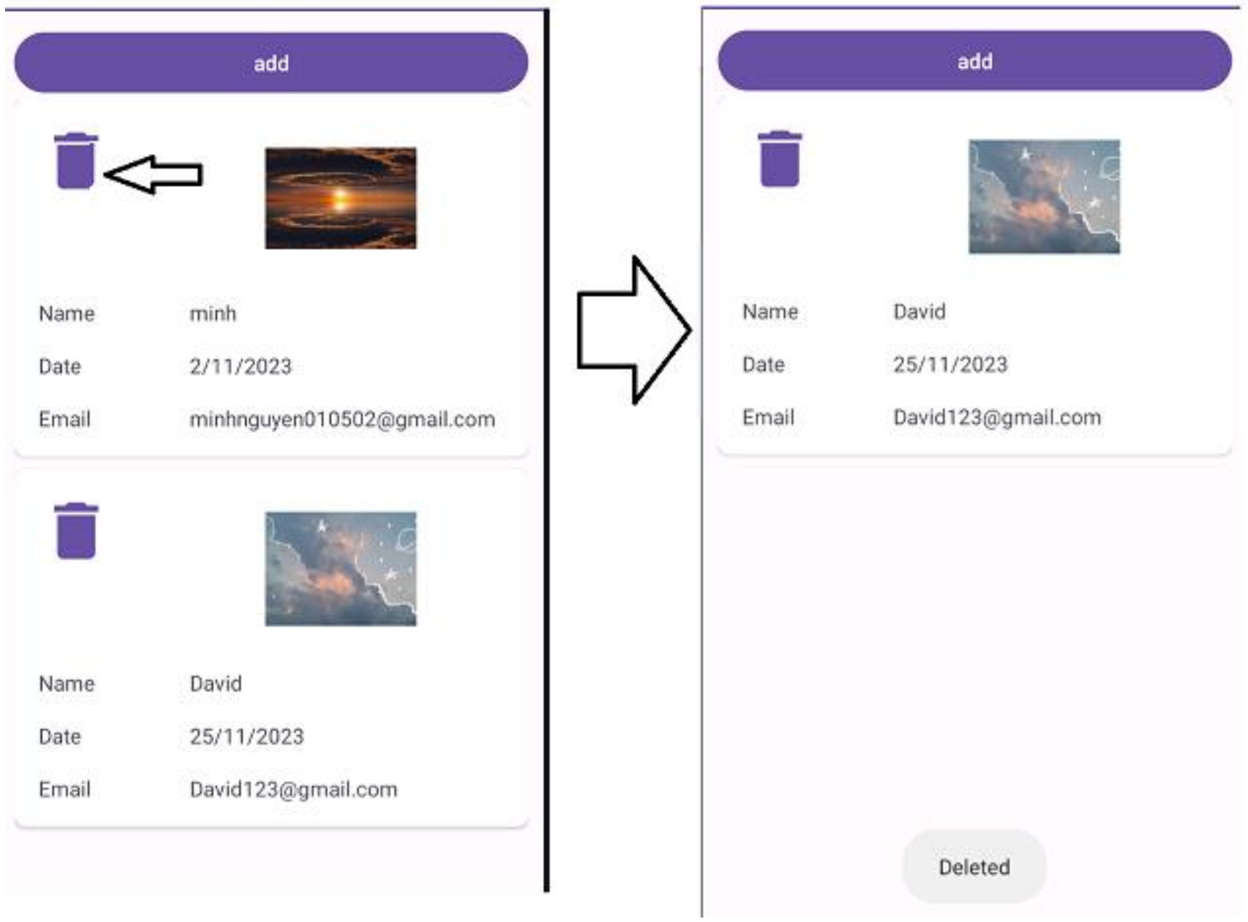


Figure 16: Deleted

2.2. Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp">

    <Button
        android:id="@+id/add"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="add" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

This layout represents a vertical arrangement of user interface elements (android:orientation:vertical), including an Add Button that triggers the action to add a new Contact with an id to identify it. and RecyclerView to display a vertical list of items. LinearLayout with vertical orientation and padding provides a simple and clean structure for these user interface elements.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp">


    <ImageView
        android:id="@+id/selectedImage"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:src="@android:drawable/ic_menu_gallery" />

    <EditText
        android:id="@+id/ename"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="name">

    </EditText>

    <EditText
        android:id="@+id/eEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="email" />

    <EditText
        android:id="@+id/edate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="date" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">

        <Button
            android:id="@+id/save"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Save" />

        <Button
            android:id="@+id/view"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="view" />

    </LinearLayout>


</LinearLayout>
```

This layout represents a vertical arrangement of user interface elements, including ImageViews for displaying images, EditTexts for text input, and Buttons for saving and viewing. The layout uses a combination of LinearLayout and layout_weight to create a visually balanced design.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="8dp"
    android:elevation="8dp"
    android:background="#CCC"
    android:layout_marginBottom="8dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="8dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="8dp">

            <Button
                android:id="@+id/btnDelete"
                android:layout_width="50dp"
                android:layout_height="50dp"
                android:background="@drawable/baseline_delete_24" />
            <ImageView
                android:id="@+id/ivImage"
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:layout_gravity="center"
                android:layout_marginStart="100dp"
                android:src="@android:drawable/ic_menu_gallery" />

        </LinearLayout>


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="8dp">
            <TextView
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Name" />
            <TextView
                android:id="@+id/tvName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Name" />
        </LinearLayout><LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="8dp">
            <TextView
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Date" />
            <TextView
                android:id="@+id/tvDate"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Date" />
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="8dp">
            <TextView
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Email" />
            <TextView
                android:id="@+id/tvEmail"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Email" />
        </LinearLayout>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Each LinearLayout in the tag groups related views horizontally, and the parent LinearLayout arranges these groups vertically. Structure designed to represent information in a card-like format with rounded corners(app:cardCornerRadius:), shadow(android:elevation:) and clear visual data separation.Contains Button (id is btnDelete) has a delete icon. Use of CardView enhances visual appeal and provides a consistent card-based user interface pattern.

```
public class MainActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    UserAdapter userAdapter;
    ArrayList<User> users;
    DBHelper dbHelper;
    Button add;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerview);
        add = findViewById(R.id.add);
        dbHelper = new DBHelper(this);

        add.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(MainActivity.this, Add.class);
                startActivityForResult(i, 0);
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        users = dbHelper.getAll();

        userAdapter =  new UserAdapter(users,this);
        recyclerView.setAdapter(userAdapter);

        LinearLayoutManager llm = new LinearLayoutManager(this);
        llm.setOrientation(RecyclerView.VERTICAL);
        recyclerView.setLayoutManager(llm);
    }
}
```

MainActivity is responsible for displaying the contact list using RecyclerView, interacting with the SQLite database through the DBHelper class. The Add button allows users to navigate to another activity (Add) to add a new contact. The onStart method is used to update the user interface with the latest data when the activity starts or continues.

```java
import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import com.example.contact.db.DBHelper;

import java.util.Calendar;

public class Add extends AppCompatActivity  {

    EditText ename;

    EditText edate;
    EditText eEmail;

    Button save, view;

    private ImageView selectedImage;
    private Uri selectedImageUri = null;
    private static final int PICK_IMAGE_REQUEST = 1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);

        view = findViewById(R.id.view);
        save = findViewById(R.id.save);
        ename = findViewById(R.id.ename);
        edate = findViewById(R.id.edate);
        eEmail = findViewById(R.id.eEmail);
        selectedImage = findViewById(R.id.selectedImage);

        edate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDatePickerDialog();
            }
        });

        selectedImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
                intent.setType("image/*"); // Chỉ chọn các tệp hình ảnh.
                startActivityForResult(intent, PICK_IMAGE_REQUEST);
            }
        });
        view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Add.this, MainActivity.class));
            }
        });

        save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name = ename.getText().toString().trim();
                String date = edate.getText().toString().trim();
                String email = eEmail.getText().toString().trim();


                if (name.isEmpty() || date.isEmpty() ||
                        email.isEmpty()) {
                    Toast.makeText(Add.this, "Please complete all
information", Toast.LENGTH_SHORT).show();
                }else {

                    DBHelper dbHelper = new DBHelper(Add.this);

                    if (selectedImageUri != null) {
                        String imagePath = selectedImageUri.toString();

                        User user = new User(name, date, email, imagePath);

                        long result = dbHelper.add(user);

                        if (result > 0) {
                            Toast.makeText(Add.this, "Saved " ,
Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(Add.this, "Failed " ,
Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(Add.this, "Don't have image",
Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
    }
    private void showDatePickerDialog() {
        DatePickerDialog datePickerDialog = new DatePickerDialog(
                this,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker view, int year, int
month, int dayOfMonth) {

                        String selectedDate =dayOfMonth + "/"+
(month+1)+"/"+year;
                        edate.setText(selectedDate);
                    }
                },
                Calendar.getInstance().get(Calendar.YEAR),
                Calendar.getInstance().get(Calendar.MONTH),
                Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
        );

        datePickerDialog.show();
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == PICK_IMAGE_REQUEST && resultCode ==
Activity.RESULT_OK && data != null) {
            selectedImageUri = data.getData();
            selectedImage.setImageURI(selectedImageUri);
        }
    }
}
```

Overall, this Add activity allows users to enter contact information, include:

- Image selection:

Clicking on the selected Image View opens the image picker, allowing users to select images from their device.

The result of the image picker is handled in the onActivityResult method.

The onActivityResult method is triggered when an image is selected. It updates the selected ImageUri and displays the selected image in the selected ImageView.

- Show date picker:

The showDatePickerDialog method initializes and displays the DatePickerDialog to select a date.

Clicking on editText will open the DatePickerDialog for the user to select a date.

The selected date is then displayed in the EditText editate.

- Save button click event:

When the user clicks the save button, the application retrieves the entered name, date, and email.

It checks if these fields are empty. If any field is empty, it displays a toast message asking the user to complete all the information.

If all information is provided, it creates a new User object with the entered data and the selected image (if any).

It then uses DBHelper to add users to the database. If successful, it will display the toast message "Saved"; otherwise, it shows "Failed" message.

```
package com.example.contact.adapter;

import android.content.Context;
import android.content.DialogInterface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.contact.R;
import com.example.contact.User;
import com.example.contact.db.DBHelper;

import java.util.ArrayList;

public class UserAdapter extends
RecyclerView.Adapter<UserAdapter.UserViewHolder>{
    ArrayList<User> users;
    Context context;

    public UserAdapter(ArrayList<User> users, Context context) {
        this.users = users;
        this.context = context;
    }

    @NonNull
    @Override
    public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View view = inflater.inflate(R.layout.detail, parent, false);
        UserViewHolder vh = new UserViewHolder(view);
        return vh;
    }

    @Override
    public void onBindViewHolder(@NonNull UserViewHolder holder, int
position) {
        final User user = users.get(position);
        holder.tvName.setText(user.getName());
        holder.tvDate.setText(user.getDate());
        holder.tvEmail.setText(user.getEmail());

        if (user.getImage() != null && !user.getImage().isEmpty()) {

            Glide.with(context).load(user.getImage()).into(holder.ivImage);
        }


        holder.btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                DBHelper dbHelper = new DBHelper(context);
                int result = dbHelper.delete(user.getId());

                if (result > 0){
                    Toast.makeText(context, "Deleted",
Toast.LENGTH_SHORT).show();
                    users.remove(user);
                    notifyDataSetChanged();
                }else {
                    Toast.makeText(context, "Failed",
Toast.LENGTH_SHORT).show();
                }
            }
        });

    }

    @Override
    public int getItemCount() {
        return users.size();
    }

    class UserViewHolder extends RecyclerView.ViewHolder{

        TextView tvName, tvDate, tvEmail;
        ImageView ivImage;
        Button  btnDelete;

        public UserViewHolder(@NonNull View v) {
            super(v);
            tvName = v.findViewById(R.id.tvName);
            tvDate = v.findViewById(R.id.tvDate);
            tvEmail = v.findViewById(R.id.tvEmail);
            ivImage = v.findViewById(R.id.ivImage);

            btnDelete = v.findViewById(R.id.btnDelete);

        }
    }


}
```

the UserAdapter serves as a bridge between the ArrayList<User> data and the RecyclerView, handling the creation of views for each item and updating them with the corresponding User data.

- onCreateViewHolder:

It uses the layout resource file R.layout.detail to create a View for each item and initializes a UserViewHolder with this view.

- onBindViewHolder:

This method is responsible for binding data to the views within the UserViewHolder.

It uses the Glide library to load and display images asynchronously.

(implementation 'com.github.bumptech.glide:glide:4.12.0')

It sets the text values for tvName, tvDate, and tvEmail based on the corresponding attributes of the User object.

It handles the click event for the delete button (btnDelete). deleting the user from the database using DBHelper, updates the dataset, and notifies the adapter that the data has changed.

- getItemCount:

This method returns the total number of items in the dataset, which corresponds to the number of User objects in the users ArrayList.

- userViewHolder:

This is an inner class that extends RecyclerView.ViewHolder. It represents a single item view within the RecyclerView.

```
package com.example.contact.db;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

```java
import com.example.contact.User;

import java.util.ArrayList;

public class DBHelper extends SQLiteOpenHelper {


    public DBHelper( Context context) {
        super(context, "user.db", null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("CREATE TABLE user (id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT , date TEXT, email TEXT, image TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS user");
        onCreate(sqLiteDatabase);
    }

    // Thêm dữ liệu
    public long add(User user) {
        SQLiteDatabase db = getWritableDatabase();

        ContentValues cv = new ContentValues();// tạo đối tượng chứa dữ liệu
        // Đưa dữ liệu vào đối tượng
        cv.put("name", user.getName());
        cv.put("date", user.getDate());
        cv.put("email", user.getEmail());
        cv.put("image", user.getImage());

        return db.insert("user",null,cv);
    }
    // Hiển thị
    public ArrayList<User> getAll(){
        ArrayList<User> users = new ArrayList<>();// tạo danh sách rỗng
        SQLiteDatabase db = getReadableDatabase();
        // Tạo con trỏ đọc dữ liệu
        Cursor cursor = db.rawQuery("SELECT * FROM user", null);

        if (cursor.moveToFirst()){  // Con trỏ ở vị trí đầu tiên
            do {
                int id = cursor.getInt(0); //  Đọc dữ liệu từ id
                String name = cursor.getString(1);
                String date = cursor.getString(2);
                String email = cursor.getString(3);
                String image = cursor.getString(4);

                User user = new User(id,name,date,email,image);
                users.add(user);
            }while (cursor.moveToNext());
        }
        return users;
    }

    public int delete(int id) {
        SQLiteDatabase db = getWritableDatabase();
        return db.delete("user", "id=?", new String[]{String.valueOf(id)});

    }
}
```

This class encapsulates the database operations for the 'user' table, providing methods to add, retrieve, and delete user records. It follows the SQLiteOpenHelper pattern commonly used in Android development for managing SQLite databases.

```
package com.example.contact;

import java.io.Serializable;

public class User implements Serializable {
    private int id;
    private String name;
    private String date;
    private String email;
    private String image;

    public User(int id, String name, String date, String email, String image)
{
        this.id = id;
        this.name = name;
        this.date = date;
        this.email = email;
        this.image = image;
    }

    public User(String name, String date, String email, String image) {
        this.name = name;
        this.date = date;
        this.email = email;
        this.image = image;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }
}
```

the User class is a model class that encapsulates the data associated with a user in an Android application. It follows best practices by encapsulating its attributes, providing constructors for flexibility in object creation, and offering getter and setter methods for controlled access to the attributes.

V. Conclusion

Conclusion, Through detailed tracking and recording, I have the opportunity to not only monitor my progress and progress but also self-assess, analyze and learn from the experience. It also helps me identify clearly state the goals and sub-goals to achieve, and provide an overview of my contributions and achievements. Although the results were not what I wanted, it helped me practice my studies. Ask questions and expand your knowledge a lot