Table of Contents

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. Basic information:

Student name	Nguyen Duc Minh
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 did you complete the exercise?	I tried but couldn't complete it☐ I did it but I feel I should have done better☐ I did everything that was asked ✓ I did more than was asked for☐
Explain	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:

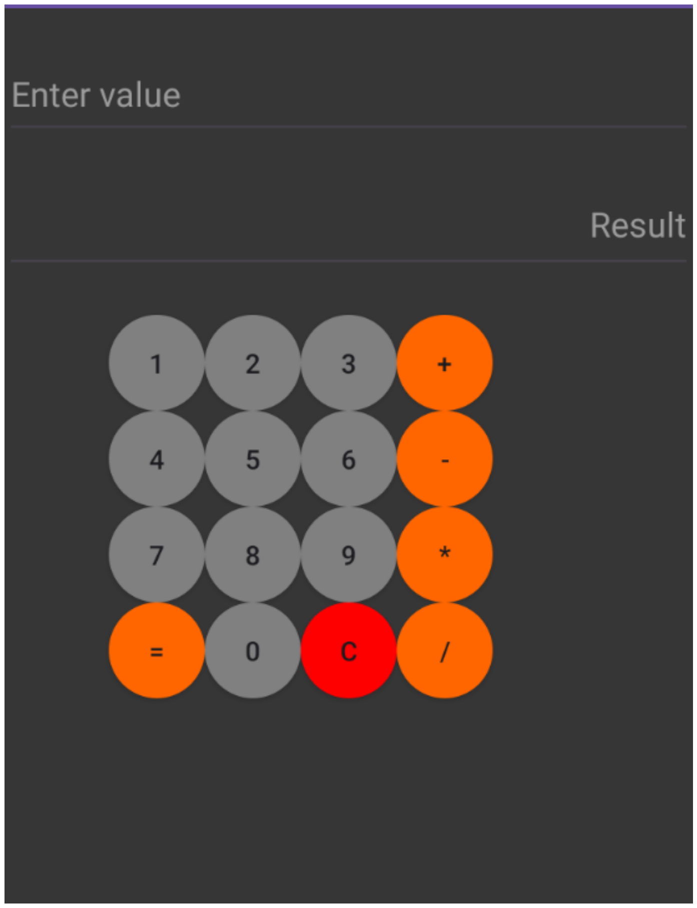


Figure 1: Calculator

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).

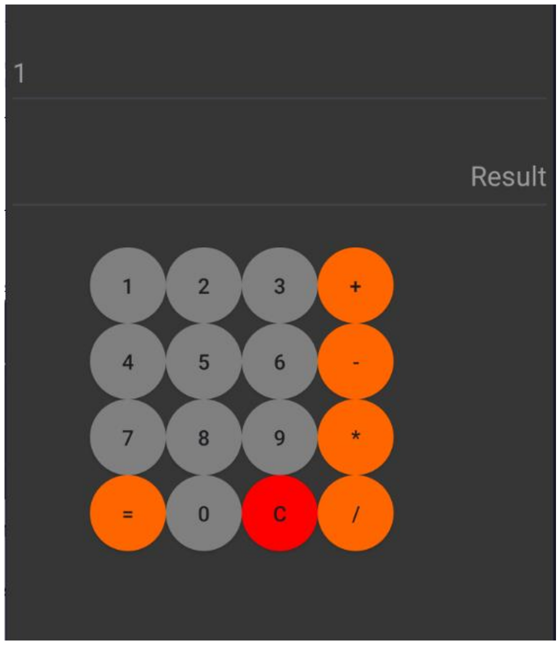


Figure 2: click button1

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

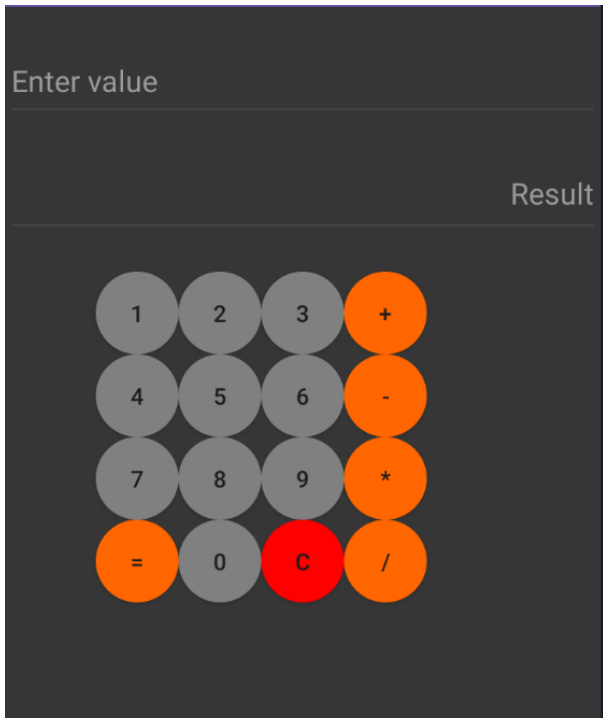


Figure 3: click button +

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

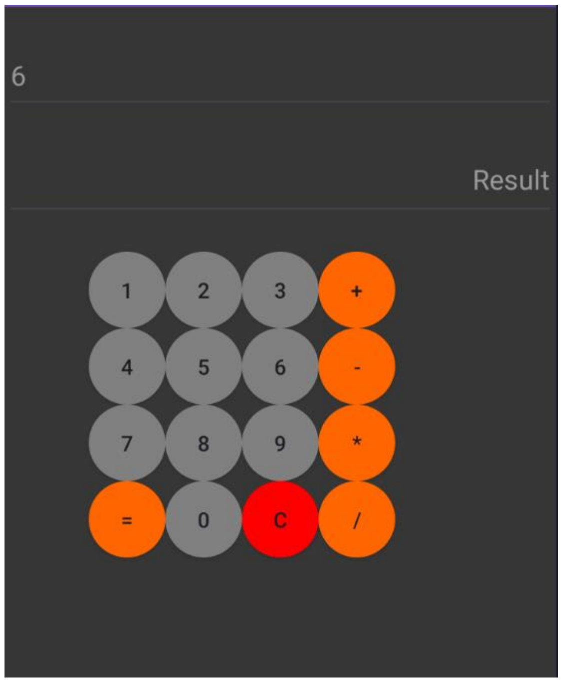


Figure 4: click button 6

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

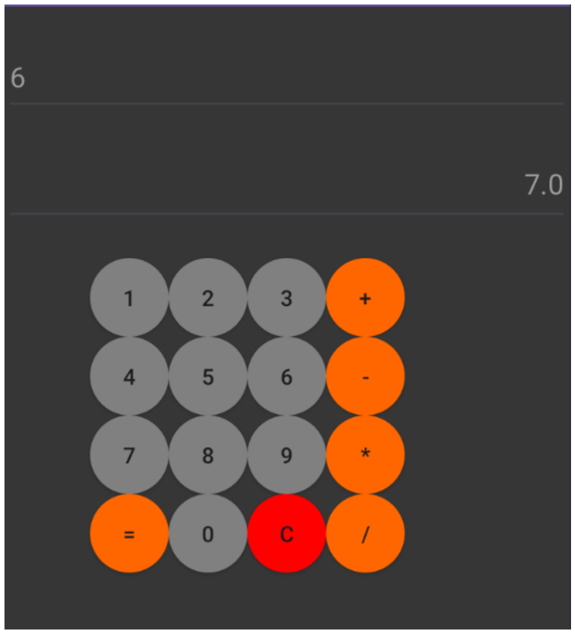


Figure 5: click button=

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

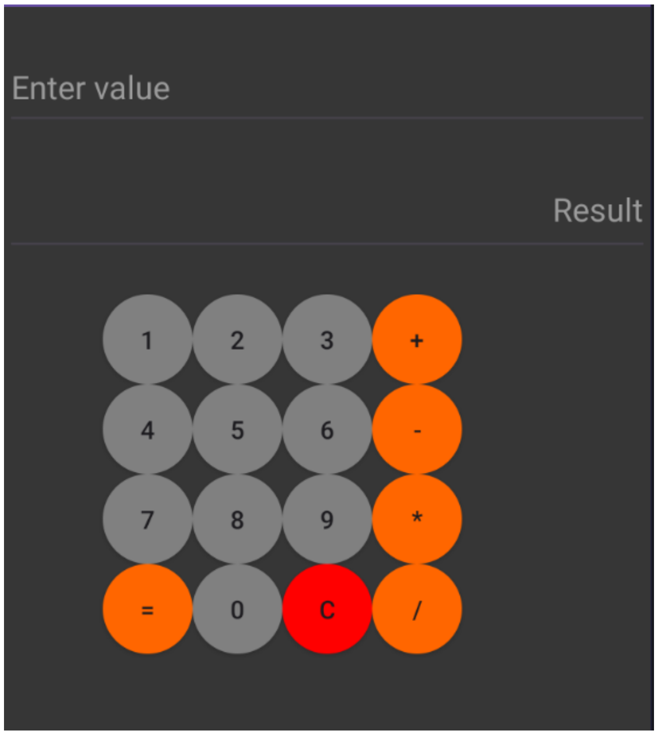


Figure 6: click button C

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

2.2. Code:

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

```
<?xml version="1.0" encoding="utf-8"?>
<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>
-----------------------------------------------------------------------------
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>
<solid android:color="#FF0000"/>
</shape>
-----------------------------------------------------------------------------
<?xml version="1.0" encoding="utf-8"?>
<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.

```
<?xml version="1.0" encoding="utf-8"?>
<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>
-----------------------------------------------------------------------------
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="30dp"/>

    <solid android:color="#FF6600"/>
</shape>
-----------------------------------------------------------------------------
<?xml version="1.0" encoding="utf-8"?>
<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.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    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/edt1"
    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/edt2"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_below="@+id/edt1"
    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/button1"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/edt2"
    android:layout_alignEnd="@+id/button4"
    android:layout_alignRight="@+id/button4"
    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.

```
<android.widget.Button
    android:id="@+id/button2"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/button1"
    android:layout_toStartOf="@+id/button3"
    android:layout_toLeftOf="@+id/button3"
    android:text="2" />

<android.widget.Button
    android:id="@+id/button3"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/button2"
    android:layout_centerHorizontal="true"
    android:text="3" />

<android.widget.Button
    android:id="@+id/button4"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button1"
    android:layout_toLeftOf="@+id/button2"
    android:text="4" />

<android.widget.Button
    android:id="@+id/button5"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignStart="@+id/button2"
    android:layout_alignLeft="@+id/button2"
    android:layout_alignBottom="@+id/button4"
    android:text="5" />

<android.widget.Button
    android:id="@+id/button6"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button3"
    android:layout_alignStart="@+id/button3"
    android:layout_alignLeft="@+id/button3"
    android:text="6" />

<android.widget.Button
    android:id="@+id/button7"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button4"
    android:layout_toLeftOf="@+id/button2"
    android:text="7" />

<android.widget.Button
    android:id="@+id/button8"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button5"
    android:layout_alignStart="@+id/button5"
    android:layout_alignLeft="@+id/button5"
    android:text="8" />

<android.widget.Button
    android:id="@+id/button9"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button6"
    android:layout_alignStart="@+id/button6"
    android:layout_alignLeft="@+id/button6"
    android:text="9" />

<android.widget.Button
    android:id="@+id/buttonadd"
```

```xml
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignTop="@+id/button3"
    android:layout_toRightOf="@+id/button3"
    android:text="+" />

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

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

<android.widget.Button
    android:id="@+id/buttoneql"
    android:background="@drawable/btn_equal"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button7"
    android:layout_toLeftOf="@+id/button2"
    android:text="=" />

<android.widget.Button
    android:id="@+id/button0"
    android:background="@drawable/button_bg"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button8"
    android:layout_alignStart="@+id/button8"
    android:layout_alignLeft="@+id/button8"
    android:text="0" />

<android.widget.Button
    android:id="@+id/buttonC"
    android:background="@drawable/btn_c"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/button9"
    android:layout_alignStart="@+id/button9"
    android:layout_alignLeft="@+id/button9"
    android:text="C" />
```

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

```
Button button0, button1, button2, button3, button4, button5, button6,
        button7, button8, button9, buttonAdd, buttonSub, buttonDivision,
        buttonMul, buttonC, buttonEqual;
EditText resultEdt;
EditText valueEdt;
float mValueOne, mValueTwo;
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

```
button0 = findViewById(R.id.button0);
button1 = findViewById(R.id.button1);
button2 = findViewById(R.id.button2);
button3 = findViewById(R.id.button3);
button4 = findViewById(R.id.button4);
button5 = findViewById(R.id.button5);
button6 = findViewById(R.id.button6);
button7 = findViewById(R.id.button7);
button8 = findViewById(R.id.button8);
button9 = findViewById(R.id.button9);
buttonAdd = findViewById(R.id.buttonadd);
buttonSub = findViewById(R.id.buttonsub);
buttonMul = findViewById(R.id.buttonmul);
buttonDivision = findViewById(R.id.buttondiv);
buttonC = findViewById(R.id.buttonC);
buttonEqual = findViewById(R.id.buttoneql);
valueEdt = findViewById(R.id.edt1);
resultEdt = findViewById(R.id.edt2);
```

Map to layout side via id.

```
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "1");
    }
});
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "2");
    }
});
button3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "3");
    }
});
button4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "4");
    }
});
button5.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "5");
    }
});
button6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "6");
    }
});
button7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "7");
    }
});
button8.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "8");
    }
});
button9.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valueEdt.setText(valueEdt.getText() + "9");
    }
});
button0.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()).

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

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

    }
});
buttonDivision.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mValueOne = 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(null)).

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

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

Figure 7: setOnClickListener C

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

III. Exercise 2:

1. Basic information:

Student name	Nguyen Duc Minh
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 did you complete the exercise?	I tried but couldn't complete it☐ I did it but I feel I should have done better☐ I did everything that was asked ✓ I did more than was asked for☐
Explain	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.

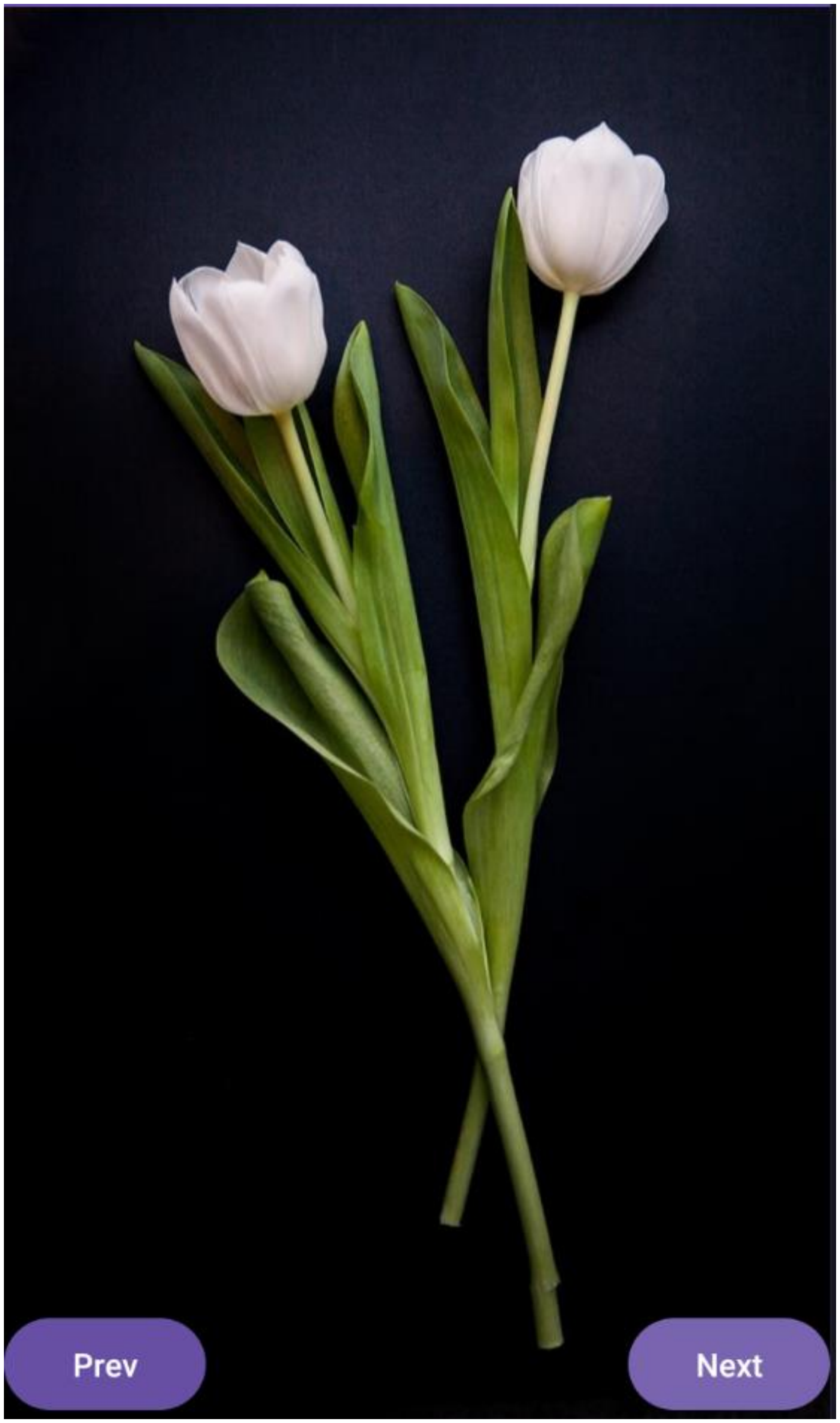


Figure 8: Show image

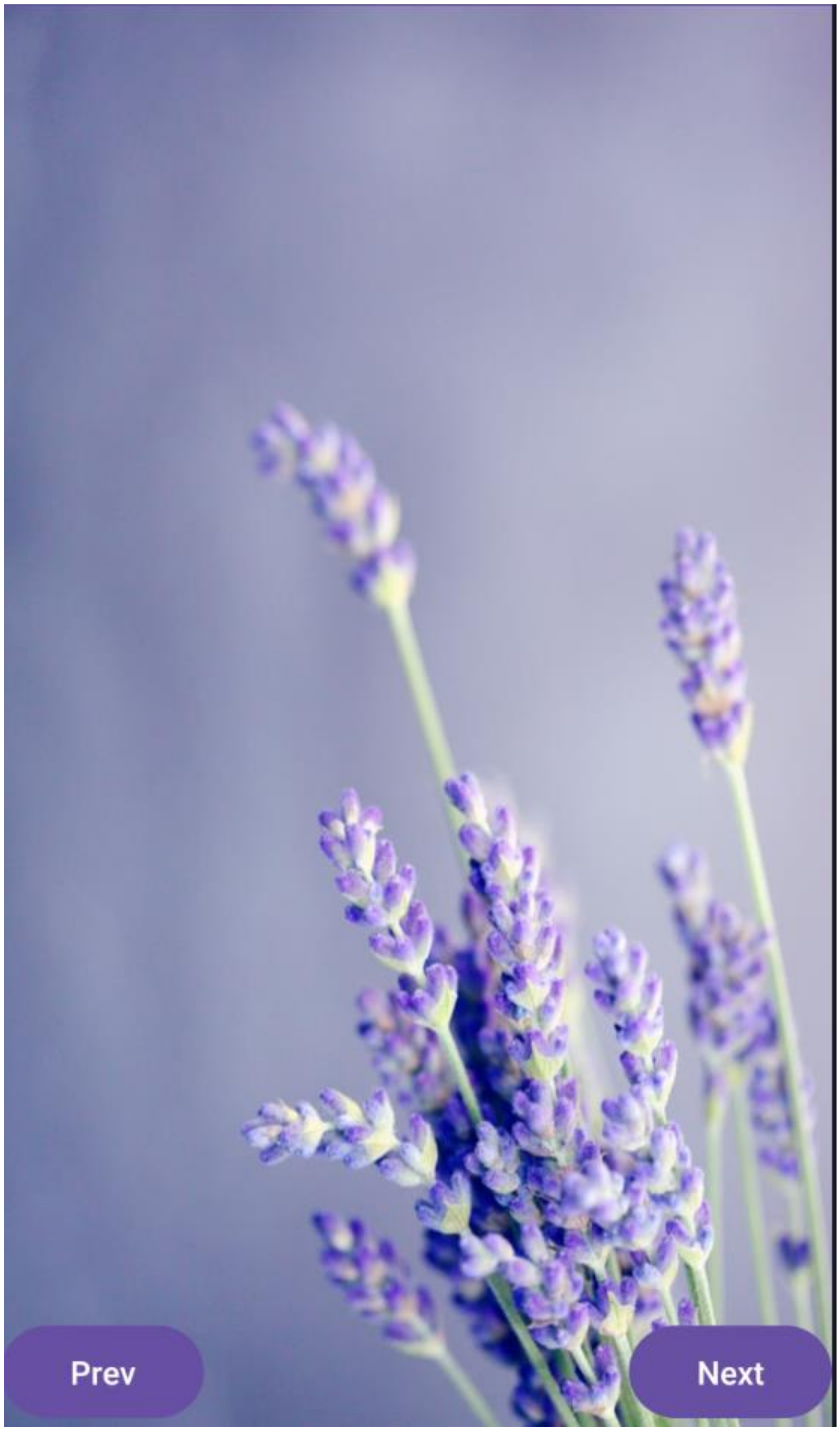


Figure 9: press next

Figure 10: Press Prev

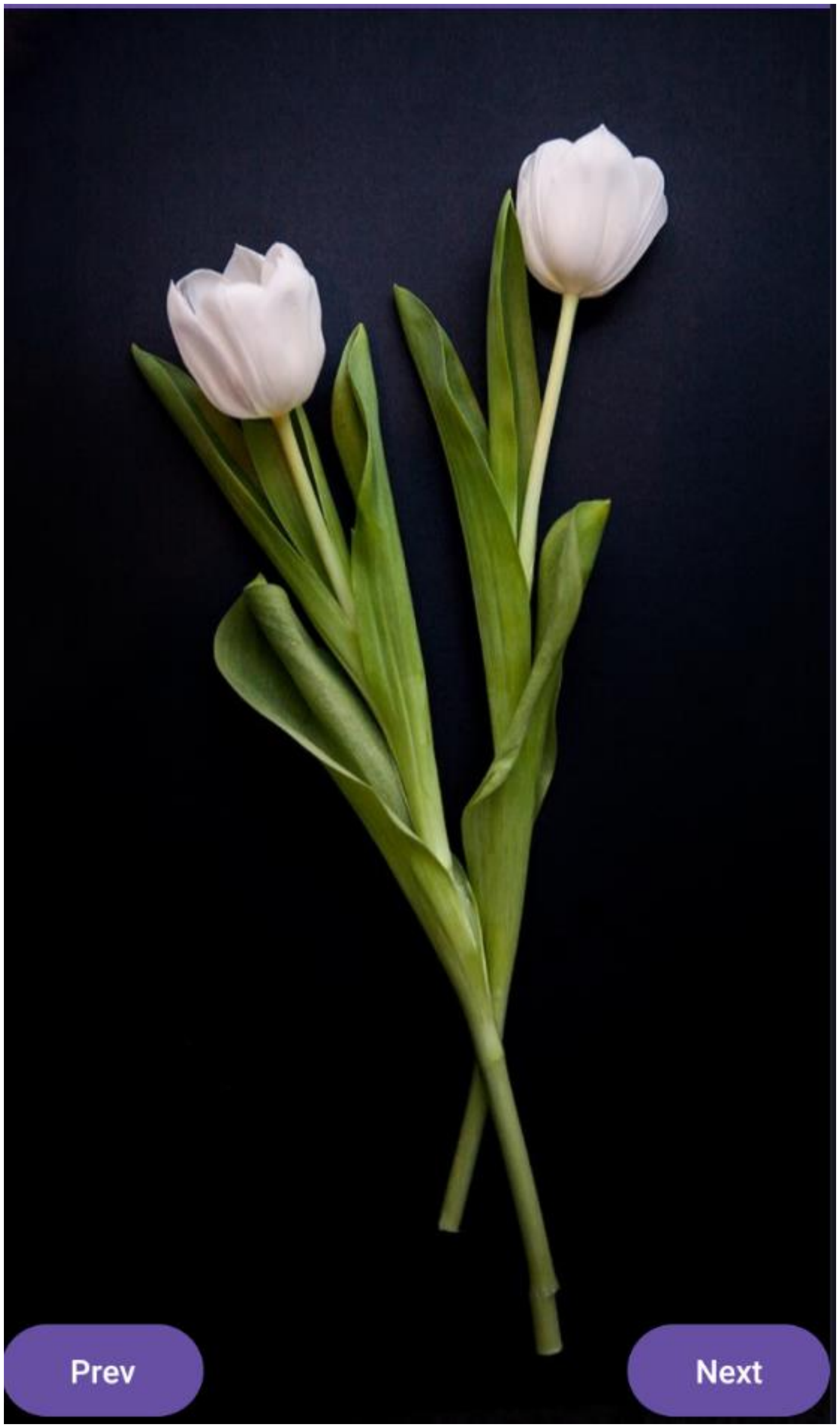


Figure 10: Press Prev

2.2. Code:

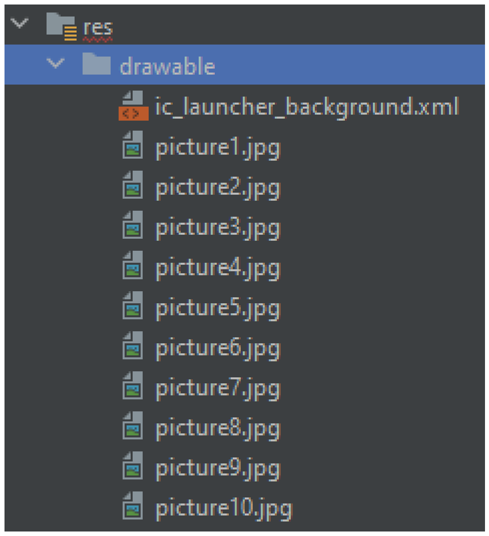

res
drawable
ic_launcher_background.xml
picture1.jpg
picture2.jpg
picture3.jpg
picture4.jpg
picture5.jpg
picture6.jpg
picture7.jpg
picture8.jpg
picture9.jpg
picture10.jpg

Figure 11: 10 pictures

I save 10 pictures into drawable file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    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.

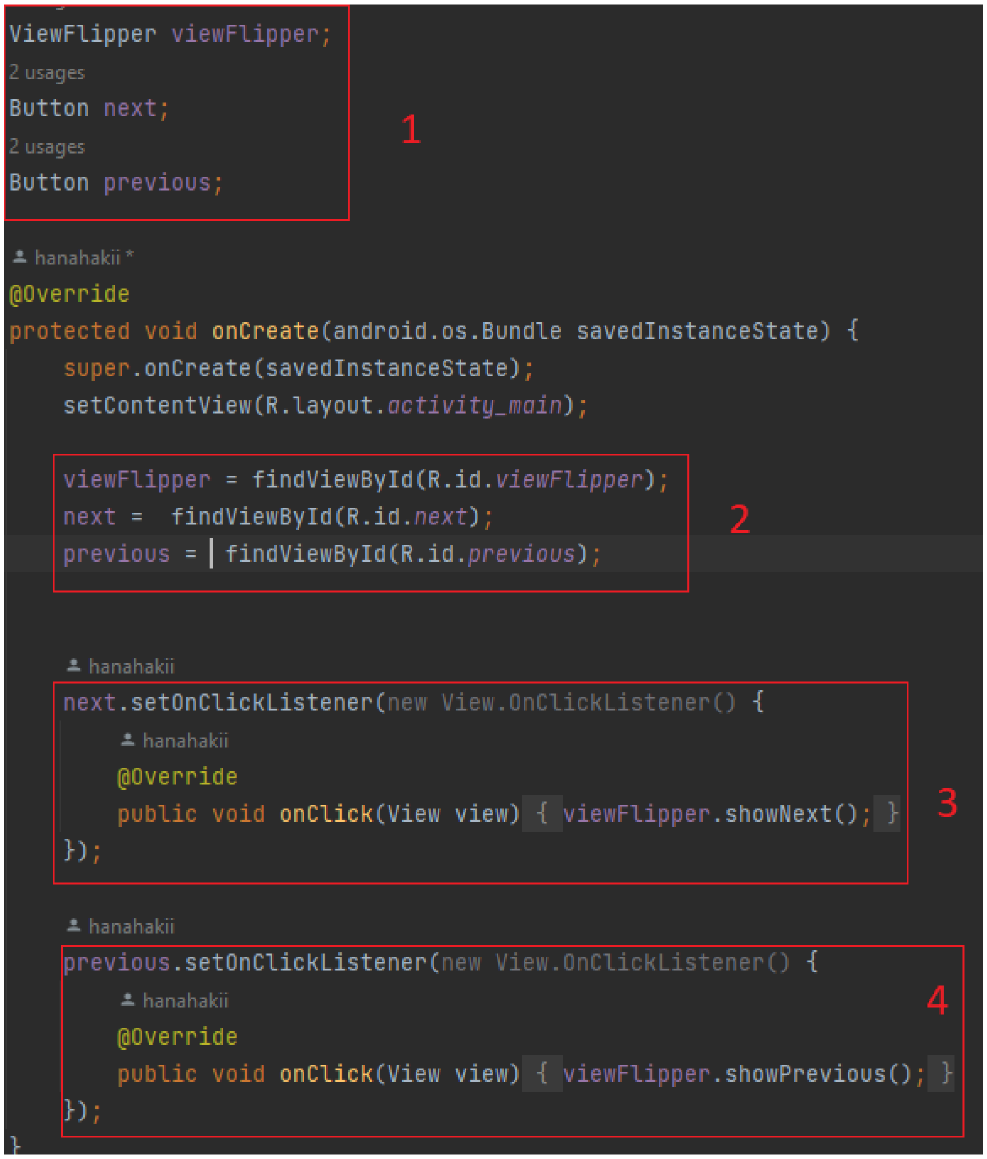
```
<ViewFlipper
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/viewFlipper">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:id="@+id/imageView"
        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"/>
    <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;
2 usages
Button next;
2 usages
Button previous;
```
1

```
hanahakii *
@Override
protected void onCreate(android.os.Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    viewFlipper = findViewById(R.id.viewFlipper);
    next =  findViewById(R.id.next);
    previous = | findViewById(R.id.previous);
```
2

```
    hanahakii
    next.setOnClickListener(new View.OnClickListener() {
        hanahakii
        @Override
        public void onClick(View view) { viewFlipper.showNext(); }
    });
```
3

```
    hanahakii
    previous.setOnClickListener(new View.OnClickListener() {
        hanahakii
        @Override
        public void onClick(View view) { viewFlipper.showPrevious(); }
    });
}
```
4

Figure 12: main activity

```
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();
    }
});
```

In boxes 3 and 4, 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. Basic information:

Student name	Nguyen Duc Minh
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 did you complete the exercise?	I tried but couldn't complete it☐ I did it but I feel I should have done better ✓ I did everything that was asked ☐ I did more than was asked for☐
Explain	I tried a lot, spent my time and energy on the task, but unfortunately, I still could not complete the main requirements of the assignment. Despite my best efforts, the complexity and challenges of the task were still formidable obstacles, preventing me from completing it completely as originally planned.

Table 3: Excercise 3

2. Exercise answer:

2.1. Screen shorts:

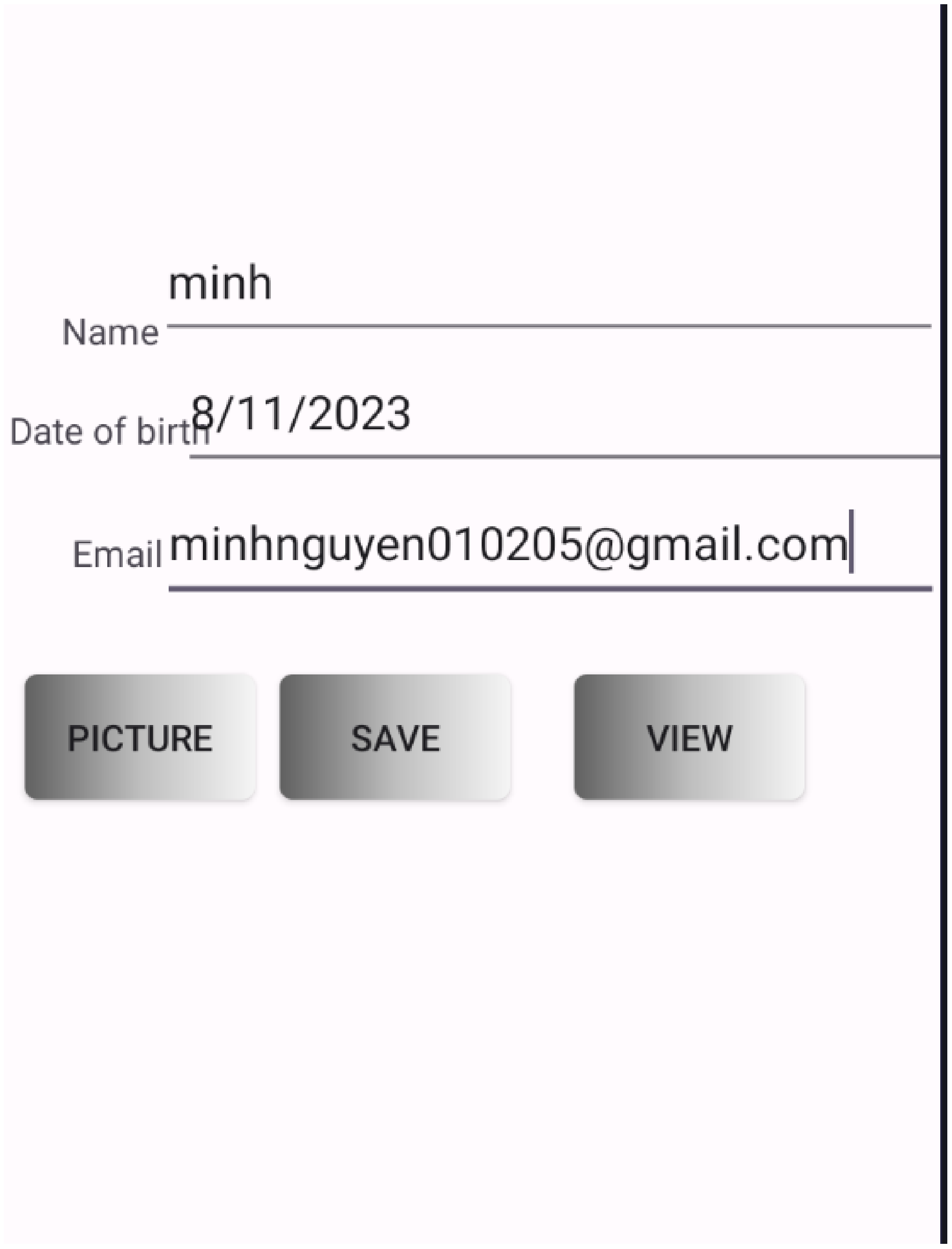

Name: minh
Date of birth: 8/11/2023
Email: minhnguyen010205@gmail.com

Figure 13: Insert data

This is the main screen of the application. Includes 3 header lines and 3 lines to enter information. Below are 3 buttons: Save to save information, View to view information and Picture to select photos, but this function is not yet completed.

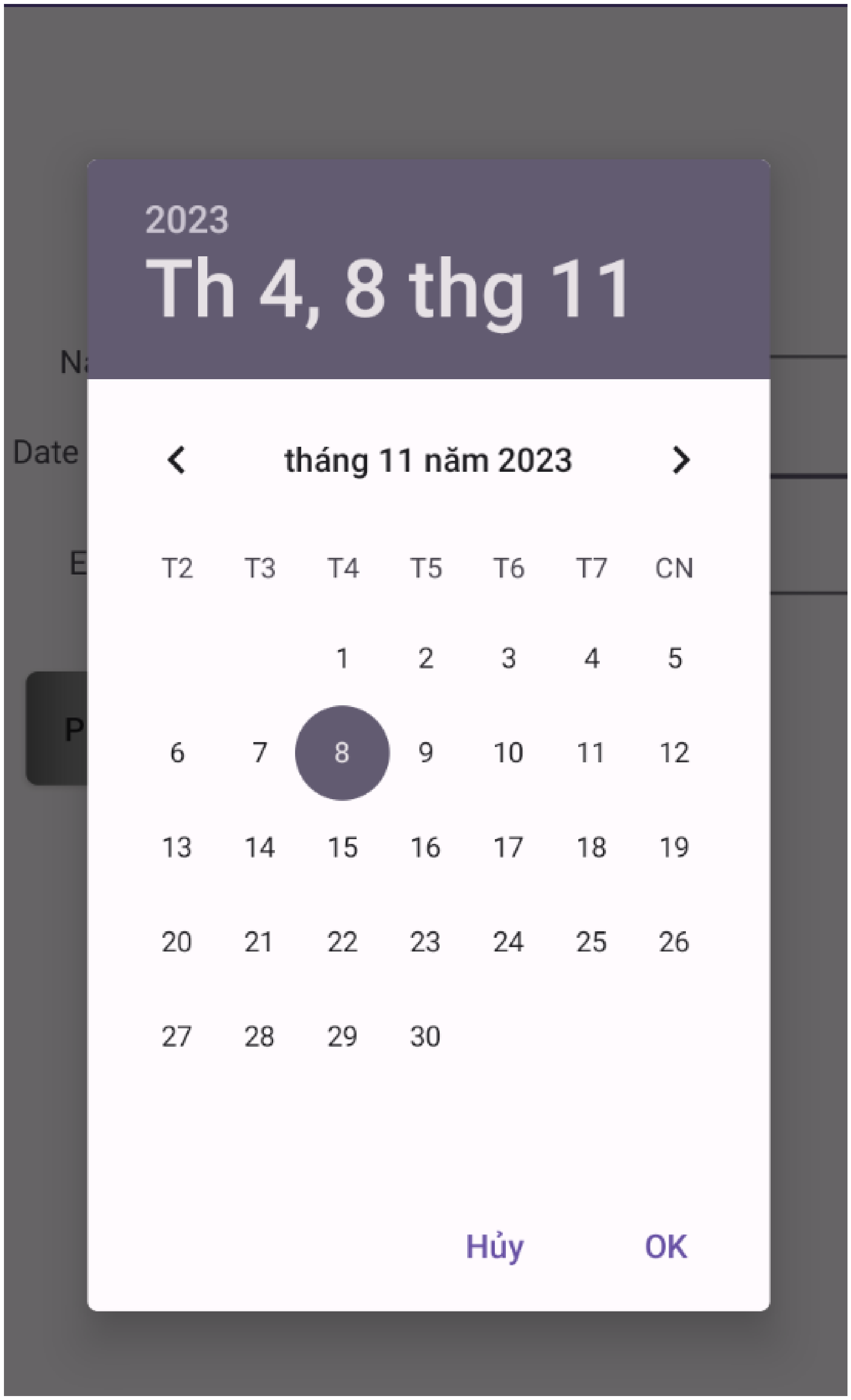
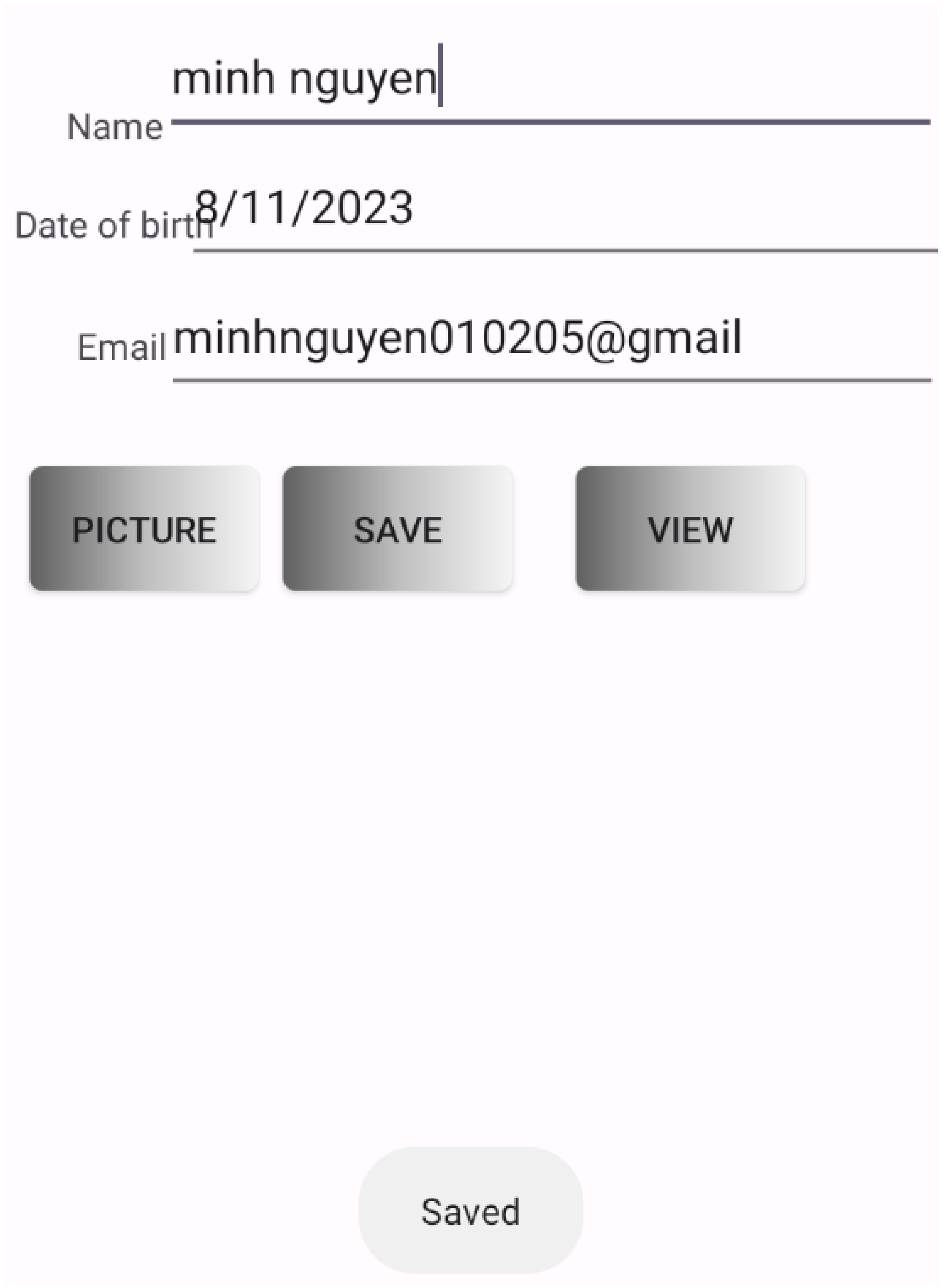


Figure 14: pick date

When clicking on the date input box, the application displays a Dialog box so the user can select the date.

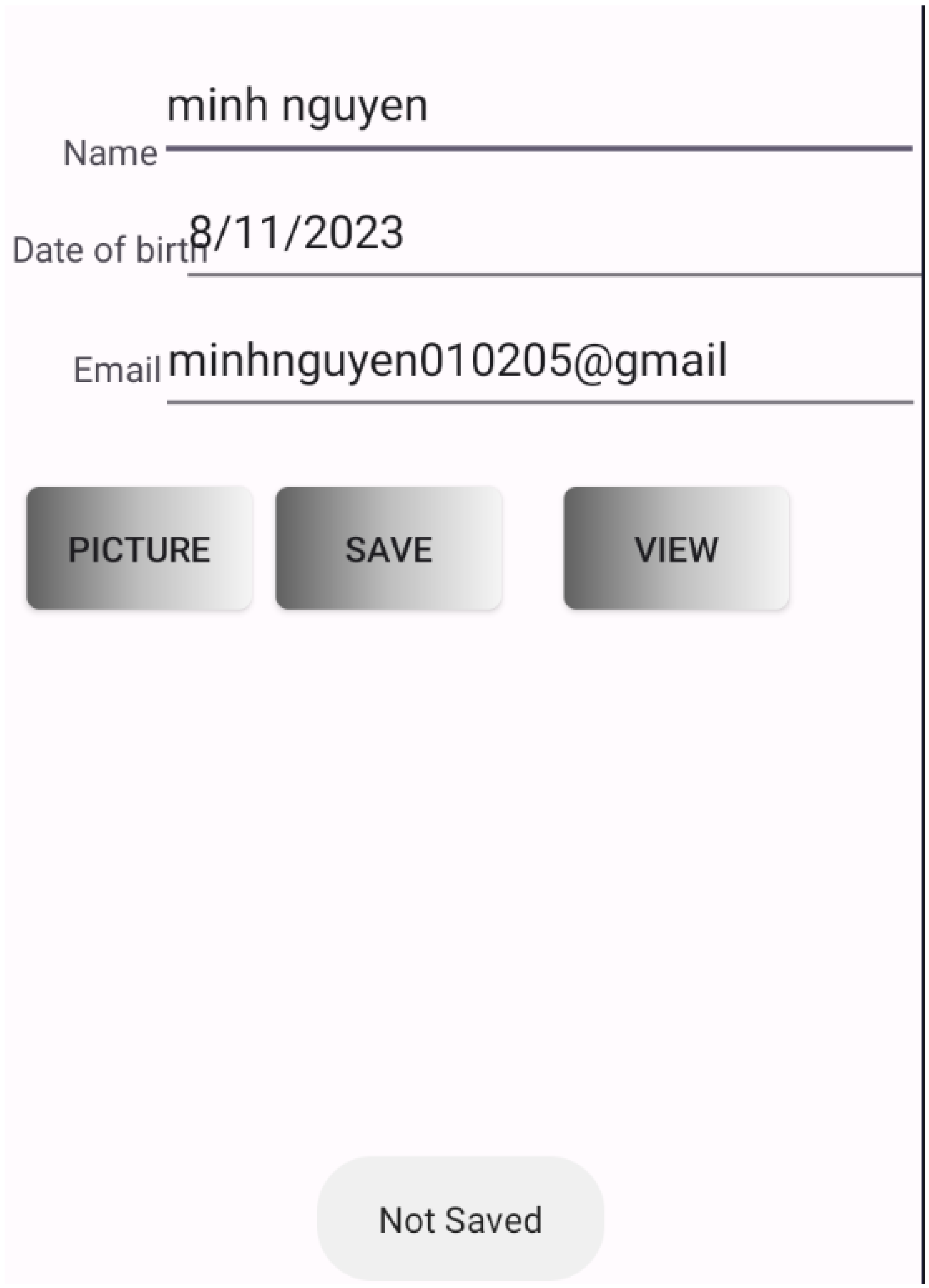

Name minh nguyen

Date of birth 8/11/2023

Email minhnguyen010205@gmail

Figure 15: Saved

The information will be checked and if the name is the same, it cannot be saved (Figure 45). If information already exists with the same name, it will not be possible to save (Figure 46).



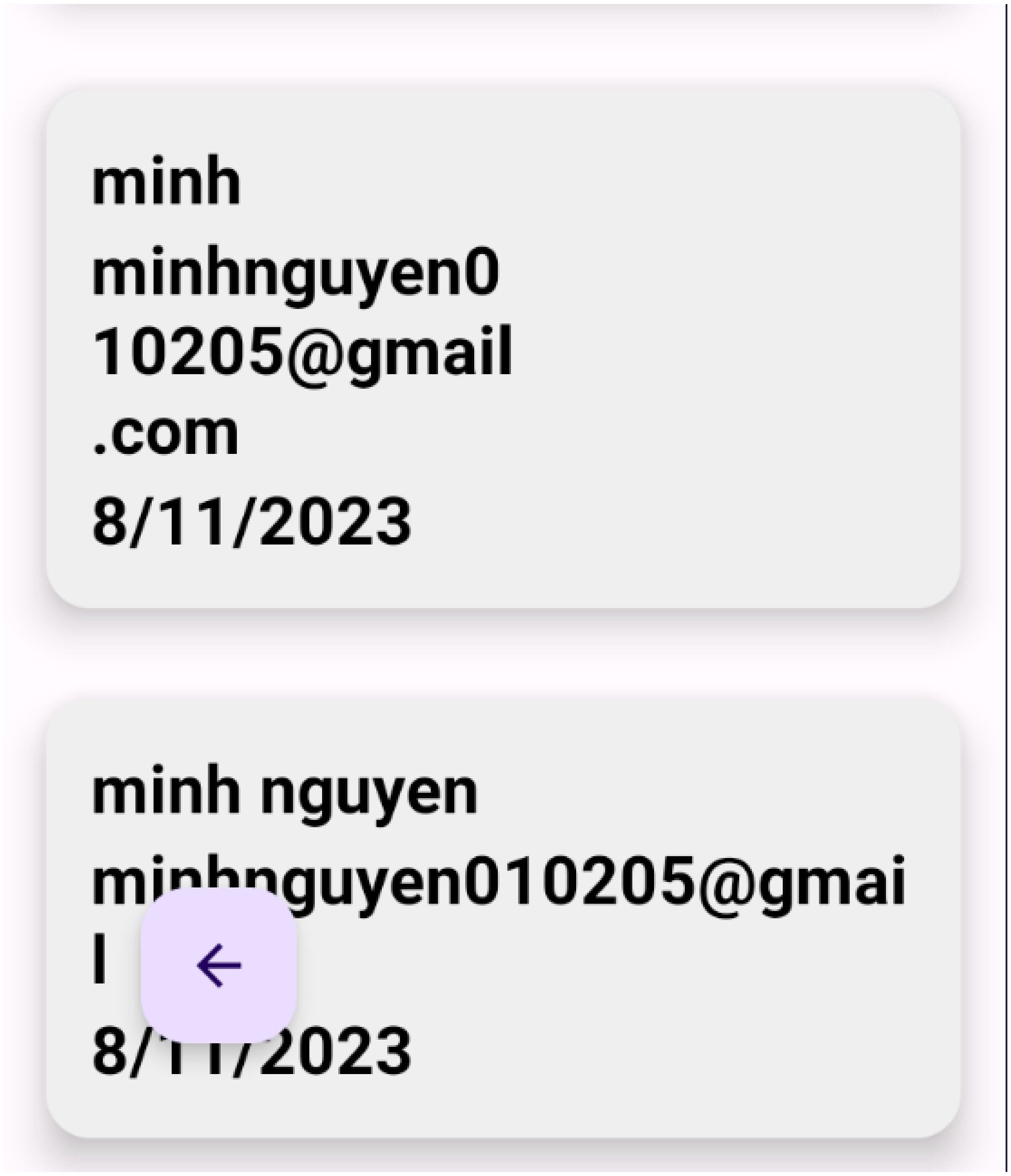
Name minh nguyen

Date of birth 8/11/2023

Email minhnguyen010205@gmail

Not Saved

Figure 16: not save


minh
minhnguyen010205@gmail.com
8/11/2023

minh nguyen
minhnguyen010205@gmail
8/11/2023

Figure 17: list user

A list of information appears and below there is a button to return to the screen to enter information

2.2. Code:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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">
```

I choose ConstraintLayout because it is a powerful and flexible layout manager for building complex user interfaces in Android applications. And it allows me to create responsive and adaptive layouts while reducing the need for nested view groups, which can lead to improved performance.

```xml
<TextView
    android:id="@+id/txtName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:text="Name"
    app:layout_constraintBottom_toTopOf="@+id/txtDateOfBirth"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.879" />

<TextView
    android:id="@+id/txtEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="28dp"
    android:layout_marginBottom="372dp"
    android:text="Email"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

<TextView
    android:id="@+id/txtDateOfBirth"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:layout_marginBottom="28dp"
    android:text="Date of birth"
    app:layout_constraintBottom_toTopOf="@+id/txtEmail"
    app:layout_constraintStart_toStartOf="parent" />
```

The 3 TextViews are set using properties like app:layout_constraintStart_toStartOf, app:layout_constraintEnd_toEndOf, app:layout_constraintTop_toTopOf, and app:layout_constraintBottom_toBottomOf.

```
<EditText
    android:id="@+id/edtName"
    android:layout_width="300dp"
    android:layout_height="50dp"
    android:ems="10"
    android:hint="Name"
    android:inputType="text"
    app:layout_constraintBottom_toTopOf="@+id/edtDateofBirth"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.758"
    app:layout_constraintStart_toEndOf="@+id/txtName"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<EditText
    android:id="@+id/edtDateofBirth"
    android:layout_width="300dp"
    android:layout_height="50dp"
    android:ems="10"
    android:hint="Date of birth"
    android:inputType="text"
    app:layout_constraintBottom_toTopOf="@+id/edtEmail"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.586"
    app:layout_constraintStart_toEndOf="@+id/txtDateOfBirth" />

<EditText
    android:id="@+id/edtEmail"
    android:layout_width="300dp"
    android:layout_height="50dp"
    android:layout_marginBottom="360dp"
    android:ems="10"
    android:hint="Email"
    android:inputType="text"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.716"
    app:layout_constraintStart_toEndOf="@+id/txtEmail" />
```

Similar to 3 TextViews 3 EditText is also set using properties like app:layout_constraintStart_toStartOf, app:layout_constraintEnd_toEndOf, app:layout_constraintTop_toTopOf, and app:layout_constraintBottom_toBottomOf.

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/btnSave"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/gradient_normal"
    android:text="Save"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/btnView"
    app:layout_constraintHorizontal_bias="0.814"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/edtEmail"
    app:layout_constraintVertical_bias="0.08" />

<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/btnPicture"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/gradient_normal"
    android:text="picture"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/btnSave"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/edtEmail"
    app:layout_constraintVertical_bias="0.08" />

<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/btnView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="52dp"
    android:background="@drawable/gradient_normal"
    android:text="View"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/edtEmail"
    app:layout_constraintVertical_bias="0.08" />
```

The 3 buttons are also bound by the same properties as above. Each button gets an additional background taken from drawable/gradient_nomamal.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
            <gradient
                android:startColor="#606060"
                android:centerColor="#bebebe"
                android:endColor="#f5f5f5" />
            <corners android:radius="5dp" />
        </shape>
    </item>
</selector>
```

The background of the buttons will be rectangular and the color will change from left to right according to the color tone #606060 -> #bebebe -> #f5f5f5. and rounded corners of 5dp (radius = 5dp)

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context=".UserList">

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

    </androidx.recyclerview.widget.RecyclerView>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginBottom="50dp"
        android:clickable="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="?attr/actionModeCloseDrawable" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

I chose RecycleView because it is part of the Android support library and offers better performance and more customization options to create. I give below a FloatingActionButton to use as a back button.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="16dp"
    android:layout_margin="16dp"
    android:backgroundTint="#efefef">
```

For each information, I choose CardView. I choose the color for each card to be @efefef and the 4 corners are 16dp. and equidistant from all four sides is 16dp (layout_margin).

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:orientation="vertical">
```

---l

choose one more Layout, LinearLayout, and Orientation, Vertical.

```xml
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_gravity="center_vertical">

    <TextView
        android:id="@+id/NameTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name: "
        android:textSize="24dp"
        android:textColor="@color/black"
        android:textStyle="bold">
    </TextView>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_gravity="center_vertical">

    <TextView
        android:id="@+id/EmailTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email: "
        android:textSize="24dp"
        android:textColor="@color/black"
        android:textStyle="bold">
    </TextView>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_gravity="center_vertical">

    <TextView
        android:id="@+id/DateOfBirthTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Date of birth: "
        android:textSize="24dp"
        android:textColor="@color/black"
        android:textStyle="bold">
    </TextView>

</LinearLayout>
```

Each data includes 3 pieces of information: email name and date of birth. Each information is a LinearLayout with orientation being Horizontal. and will center (layout_gravity). The text color is black (TextColor) and bold (TextStyle).

```java
package com.example.contactdatabase;

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

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

public class DBHelper extends SQLiteOpenHelper {

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

    @Override
    public void onCreate(SQLiteDatabase DB) {
        DB.execSQL("create Table Userdetails(name TEXT primary key, email
TEXT, date TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase DB, int i, int ii) {
        DB.execSQL("drop Table if exists Userdetails");
    }

    public Boolean insertuserdata(String name, String email, String date){
        SQLiteDatabase DB = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("name", name);
        contentValues.put("email", email);
        contentValues.put("date", date);

        long result = DB.insert("Userdetails", null, contentValues);
        if (result == -1){
            return false;
        }else {
            return true;
        }
    }

    public Cursor getdata(){
        SQLiteDatabase DB = this.getWritableDatabase();
        Cursor cursor = DB.rawQuery("Select * from Userdetails", null);
        return cursor;
    }
}
```

DBHelper will inherit the SQLiteOpenHelper class. This is a class that Android allows to handle SQLite database operations, so I can create another class that inherits it and customize the database control to my liking.

After inheriting from the SQLiteOpenHelper class, I need to override the onCreate() and onUpgrade methods.

onCreate(): This is where I write table creation statements. It is called when the database has been created.

onUpgrade(): It is called when the database is upgraded, for example editing table structures, adding changes to the database, etc.

I will create a method insertuserdata() that takes 3 parameters. ContentValues is used to store values corresponding to fields in the table. SQLiteDatabase contains methods to create, delete, and execute SQL commands, which will be used to insert values into fields in the table.

I will use Cursor to store the return value. The getdata() method will return all users in the table.

```
package com.example.contactdatabase;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
public class userAdapter extends
RecyclerView.Adapter<userAdapter.MyViewHolder> {
    private Context context;
    private ArrayList name_id, emai_id,date_id;
    public userAdapter(Context context, ArrayList name_id, ArrayList emai_id,
ArrayList date_id) {
        this.context = context;
        this.name_id = name_id;
        this.emai_id = emai_id;
        this.date_id = date_id;
    }
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(context).inflate(R.layout.userentry,parent,false);
        return new MyViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position)
{
        holder.name_id.setText(String.valueOf(name_id.get(position)));
        holder.emai_id.setText(String.valueOf(emai_id.get(position)));
        holder.date_id.setText(String.valueOf(date_id.get(position)));
    }
    @Override
    public int getItemCount() {
        return name_id.size();
    }
    public class MyViewHolder extends RecyclerView.ViewHolder {

        TextView name_id, emai_id,date_id;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            name_id = itemView.findViewById(R.id.NameTxt);
            emai_id = itemView.findViewById(R.id.EmailTxt);
            date_id = itemView.findViewById(R.id.DateOfBirthTxt);
        }
    }
}
```

Adapter is an important component to connect data to RecyclerView. What is certain is that data will be displayed and ViewHolders managed correctly.

The adapter must develop important methods such as onCreateViewHolder, onBindViewHolder, and getItemCount.

onCreateViewHolder: Creates a new ViewHolder for each item in the list.

onBindViewHolder: Bind data to ViewHolder for each item.

getItemCount: Returns the number of items in the list (name_id.size()).

The adapter can also monitor events when users interact with the list and handle actions accordingly.

The `RecyclerView.ViewHolder` class contains a View object to represent an item in the list. Specifically, it contains UI (Views) components like TextView(name, eamil,birthdate. I will map these Views from XML resource and bind them to NameTxt variable members , EmailTxt, DateOfBirthTxt

----

```
package com.example.contactdatabase;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
```

```java
import androidx.recyclerview.widget.RecyclerView;

import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;

public class UserList extends AppCompatActivity {
    RecyclerView recyclerView;
    ArrayList<String> name, email, date;
    FloatingActionButton back;
    DBHelper DB;
    userAdapter adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_list);
        DB = new DBHelper(this);
        name = new ArrayList<>();
        email = new ArrayList<>();
        date = new ArrayList<>();
        recyclerView = findViewById(R.id.recyclerView);
        adapter = new userAdapter(this, name, email, date);
        recyclerView.setAdapter(adapter);
        back = findViewById(R.id.back);
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        displaydata();
    }

    private void displaydata() {
        Cursor cursor = DB.getdata();
        if (cursor.getCount()==0){
            Toast.makeText(UserList.this, "No Entry Exists",
Toast.LENGTH_SHORT).show();
            return;
        }else {
            while (cursor.moveToNext()){
                name.add(cursor.getString(0));
                email.add(cursor.getString(1));
                date.add(cursor.getString(2));
            }
        }
    }
}
```

I declare the elements. Each attribute I consider is an ArrayList. The back button performs the finish() function as well as returning to the main screen (input screen). I created a displaydata() function that uses Cusor to retrieve data. If there is no data, the message "No Entry Exists" will be displayed. If cso will use a while loop to print out and use the cursor.moveToNext() function.

```
date.setOnClickListener(view -> {
    MyDatePicker dlg = new MyDatePicker();
    dlg.setDateField(date);
    dlg.show(getSupportFragmentManager(), " Date!");
});
public static class MyDatePicker extends DialogFragment implements
DatePickerDialog.OnDateSetListener {
    private EditText dateField;
    public void setDateField(EditText dateField) {
        this.dateField = dateField;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the current date as the default date in the picker
        if (dateField.getText().length() != 0) {
            String date = dateField.getText().toString();
            String[] separated = date.split("/");
            int year = Integer.parseInt(separated[2]);
            int month = Integer.parseInt(separated[1]);
            int day = Integer.parseInt(separated[0]);
            return new DatePickerDialog(getActivity(), this, year, month - 1,
day);
        } else {
            final Calendar c = Calendar.getInstance();
            int year = c.get(Calendar.YEAR);
            int month = c.get(Calendar.MONTH);
            int day = c.get(Calendar.DAY_OF_MONTH);
            return new DatePickerDialog(getActivity(), this, year, month,
day);
        }
        // Create a new instance of DatePickerDialog and return it
    }
    @Override
    public void onDateSet(DatePicker datePicker, int selectedYear,
                          int selectedMonth, int selectedDay) {
        String dateReturn = selectedDay + "/" + (selectedMonth + 1) + "/"
                + selectedYear;
        dateField.setText(dateReturn);
    }
}
```

When clicking on the date input bar, a new View is created and calls to Class MyDatePicker. This class will have the setDateField() function.

Create a dialog for users to select date, month, year.

```java
package com.example.contactdatabase;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.Intent;
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 java.util.Calendar;

public class MainActivity extends AppCompatActivity {
    EditText name,email,date;
    Button save, view;
    DBHelper DB;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    name = findViewById(R.id.edtName);
    email = findViewById(R.id.edtEmail);
    date = findViewById(R.id.edtDateofBirth);
    save = findViewById(R.id.btnSave);
    view = findViewById(R.id.btnView);

    DB = new DBHelper(this);
```

```java
view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(MainActivity.this, UserList.class));
    }
});
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String txtName = name.getText().toString();
        String txtEmail = email.getText().toString();
        String txtDate = date.getText().toString();
        Boolean isSaveData = DB.insertuserdata(txtName,txtEmail,txtDate);
        if (isSaveData==true){
            Toast.makeText(MainActivity.this, "Saved",
Toast.LENGTH_SHORT).show();
        }else {
            Toast.makeText(MainActivity.this, "Not Saved", 
Toast.LENGTH_SHORT).show();
        }
    }
});
}
```

When I click the View button, I catch the user event and move to a new screen using Intent.

The event when clicking the Save button, the data entered in the input cells will be saved in the variables txtName, txtEmail, txtDate and the variable isSaveData with a Boolean data type that calls the insertuserdata() function.

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