Assignment 1 Report

Student Name: Huu Nhan Le

Student ID: 104171133

1.Acknowledgement

The slides and labs in the modules of COS30017 course
Kotlin Bootcamp for Programmers
No Generative AI tools were used for this task

2. Introduction

In the assignment 1 of the COS30017 unit, I will create an app to score for a climbing club. The app must have several features like layout for both portrait and landscape mode, multilingual support, etc. This report will cover the time logs, key design decisions of the developers, the problems and their solutions, the implementation and the debugging/testing processes, demonstration, as well as a general reflection on what could be enhanced in the future.

3. Time Logs

Day 1-2 (27/01 -28/01): Review the general Kotlin language and guide to Android Studio through Canvas and mostly Google official slides for Android developers
Day 2-4 (28/01 -29/01): Do the labs in week 2 to 4 on Canvas like Calculator or Light Me Up to build the knowledge and skills necessary for the first assignment
Day 5 (30/01): Design the the UI/UX for this app, both landscape and potrait mode, as well as the duolingual support for English and Vietnamese
Day 6 (31/01): Code the “back-end” or MainActivity.kt to complete the functionality of the app
Day 7 (01/02): Add in the logging, as well as any missing requirements from the assignments, and test/debug the app thoroughly

4. Key Design Decisions

4.1. Layout Type

The two layout types are used for this app are Linear Layout for portrait mode and Constraint Layout for landscape mode. Those two are chosen because they have been demonstrated in the labs on Canvas so I would be more familiar with them. The Relative Layout or Framerate Layout may be more useful if I actually want to use an image for the background of the app without it being stretched. However, as they are not instructed yet, they are more risky to implement in this assignment.

4.2. Languages

I decided to use English(US) and Vietnamese as the two languages supported in the mountain climbing app. I feel they basically explain themselves, as English(US) already exists with a strings.xml file and Vietnamese is the native language for me and my instructor. Therefore, it is better to go with those two languages.

4.3. UI/UX

The user interface of the app will be kept very simple. Obviously, there will be three buttons (Climb, Fall, Reset) and a TextView displaying the score, per the requirements of the assignment. Other than that, I also include a TextView for the name of the app and a basic background for the app. This is to simplify the process of maintaining the consistency between the portrait and the landscape designs.

4.4. Save the Score on Rotation

To achieve this feature, I choose to use the recommended method saveInstanceState. More specifically, the function onSaveInstanceState(outState: Bundle) will be utilized to save relevant information like current score and hold to restore upon rotating the screen. Similar to many crucial design decisions before, this is due to it having been instructed in the lab Life Skills on Canvas.

4.5. Logging

As log is required to implement in the app for debugging purposes, I divide them into three main categories with its appropriate tag. The first one is the LIFECYLCE tag, included in the onStart(), onResume() functions. The second one is the TRACKING tag, used to keep track of which button has just been clicked and what the score and hold after that action. The final one is the BUTTONS tag, showing whether the two buttons Climb and Fall are enabled or not.

5. Problems and Solutions

5.1. Background Image Stretching

Intially I used an image related to mountain climbing for the background of the app. But as explained above, I would have to either switch to Relative or Framerate layout to properly scale and crop the image with ImageView or use multiple pictures suitable for hdpi, mdpi, etc. Both ways are too cumbersome, so I decided to just use a simple gradient background of pink and purpose to negate the stretching effect of the android:background attribute.

5.2. Falling Once

The assignment specifically demanded that after a climber has fallen, they cannot climb further. Therefore, I created a private boolean variable called hasFallen with the initial value of False. After the Fall button is clicked, this variable will be set to True, disabling both the Climb and Fall buttons in the process. Only after the Reset button is clicked will this variable be set to False again, thus enabling the Climb button

5.3. Buttons Enabling

In addition to the request in 5.2, the climber also can’t fall unless they reach the first hold or can’t climb or fall once they reach the final hold. To consolidate both this request and the 5.2 one, I created a function called updateButtons with a when expression which disables or enables the button depending on the current hold of the climbing and the value of the hasFallen variable.

5.4. Minimum API

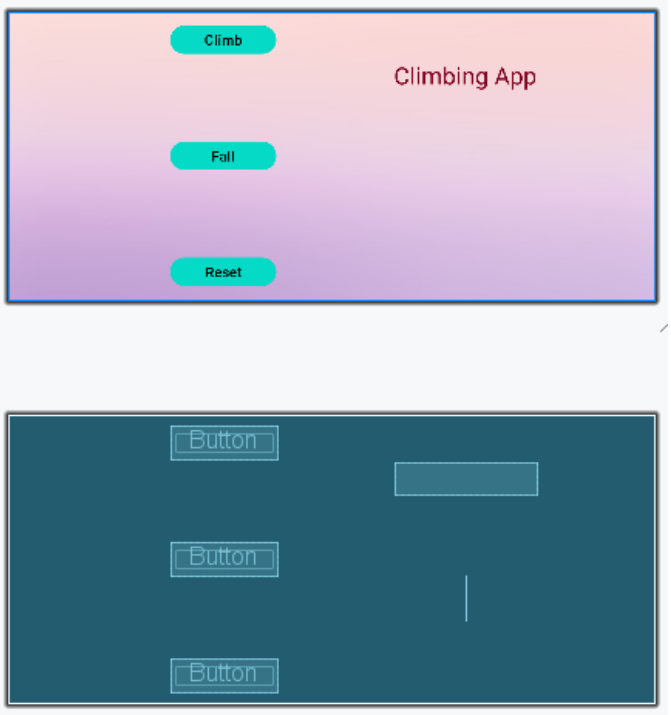
This project was created with the latest API 35 per the request of the instructor. However, I can’t run the Android Emulator on my Windows laptop and the device I have is a Realme C30S with Android 12 OS. So I have to go to the Modules setting of the project and change the Min SDK version to 31 to be able to test the app on my device.

6. UI/UX

6.1. Portrait Mode

In this mode, the root will be a LinearLayout with a vertical organization and a background of the image in the drawable folder. Next, there will be a TextView with the @string/app_name for text attribute, center for gravity, 34sp for textSize, #800020 for textColor, 100dp for layout_marginTop to separate the app name from the top of the screen a bit. Right below is another TextView with the id “score”, occupying most of the phone screen, with the center value for gravity attribute and 60sp for textSize attribute. And near the bottom of the screen, you can see three buttons lying side by side. This is because all the buttons are grouped in a LinearLayout with the horizontal orientation, wrap_content for layout_height, and 100dp for layout_marginBottom to create some distance from the bottom of the screen. All three buttons contain the appropriate text in strings.xml in text attribute and especially @drawable/button_selector in backgroundTint to change the color of the buttons based on their enabled state.

6.2. Landscape Mode



All of the main elements in the portrait mode are still kept the same in the landscape mode. The main difference is the root of the activity_main.xml in the layout-land folder is a ConstraintLayout. In addition, the three buttons, the two TextView for score and app name have constraints to match the guideline of the constraint layout.

6.3. Multilingual Support

```xml
<resources>
    <string name="app_name">Climbing App</string>
    <string name="climb_button">Climb</string>
    <string name="fall_button">Fall</string>
    <string name="reset_button">Reset</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ứng dụng leo núi</string>
    <string name="climb_button">Leo</string>
    <string name="fall_button">Ngã</string>
    <string name="reset_button">Chơi Lại</string>
</resources>
```

There are two strings.xml files in the values and values-vi folders respectively. Based on the original strings.xml file, I create a new value resource file, choose the locale qualifier then Vietnamese language. All four strings like app_name, climb_button, fall_button, reset_button must be present in both strings.xml files, with proper translation for the values inside the elements.

7. Functionality

7.1. Buttons

```
private fun climbAction(): Int {
    trackingHold +=1 //User has climbed one additional hold

    when (trackingHold)
    {
        in 1 ≤ .. ≤ 3 -> trackingScore +=1
        in 4 ≤ .. ≤ 6 -> trackingScore +=2
        in 7 ≤ .. ≤ 9 -> trackingScore +=3
    }

    if (trackingScore >= 18) trackingScore = 18 //Additional measure to guarantee sc
    Log.i( tag: "TRACKING",  msg: "Climb, Score: $trackingScore, Hold: $trackingHold")

    return trackingScore
}
```

For the climb button, a variable used to keep track of the climber's hold, trackingHold, will be incremented when the climbAction function is called. Additionally, depending on if the climb reaches which zone, the number of points will be added to the trackingScore, the variable keeping track of the climber's score accordingly. There is also a condition to make sure the trackingScore can't exceed 18 before returning it

```
private fun fallAction(): Int {
    trackingScore -= 3
    hasFallen = true

    if (trackingScore < 0) trackingScore = 0 //Additional measure to guarantee scor
    Log.i( tag: "TRACKING",  msg: "Fall, Score: $trackingScore, Hold: $trackingHold")

    return trackingScore
}
```

For the fall button, I used the fallAction function for it. The trackingScore variable will be minus three and the hasFallen will be set to the true value. And similar to the climbAction function, there is also a condition to guarantee the function can't return a negative value.

```
private fun resetAction(): Int {
    trackingScore = 0
    trackingHold = 0
    hasFallen = false //reset hasFallen to false to enable climb button

    Log.i( tag: "TRACKING",  msg: "Reset, Score: $trackingScore, Hold: $trackingHold")

    return trackingScore
}
```

The resetAction function is implemented for the reset button in the app. It will set the value of trackingScore and trackingHold to 0 and hasFallen to false.

```
private fun updateButtons() {

    when (trackingHold) {
        0 -> { //safeguard to make sure user
            score.setTextColor(Color.GRAY)

            climbButton.setEnabled(true)
            fallButton.setEnabled(false)
        }
        in 1 ≤ .. ≤ 3 -> {
            score.setTextColor(Color.BLUE) //

            if (hasFallen) {
                climbButton.setEnabled(false)
                fallButton.setEnabled(false)
            } else {
                climbButton.setEnabled(true)
                fallButton.setEnabled(true)
            }
        }
```

Finally, the most crucial function in MainActivity.kt is updateButtons. Firstly, it will change the color of the score to red, green, or blue based on the hold of the climber. Secondly, it will enable or disable the climb and fall buttons depending on the climber's hold and the value of hasFallen.

7.2. saveInstanceState

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    outState.putInt("SCORE", trackingScore)
    outState.putInt("HOLD", trackingHold)
    outState.putBoolean("FALLEN", hasFallen)
    Log.i( tag: "LIFECYCLE",  msg: "saveInstanceState: score - $trackingScore, hold - $trackingHold, hasFallen - $hasFallen")
}
```

```
savedInstanceState?.let {
    trackingScore = savedInstanceState.getInt( key: "SCORE") ,
    trackingHold = savedInstanceState.getInt( key: "HOLD") //1
    hasFallen = savedInstanceState.getBoolean( key: "FALLEN")
    score.text = trackingScore.toString()
    updateButtons()
}
```

In the onSaveInstanceState function, the current value of trackingScore, trackingHold and hasFallen will be inserted into the mapping of the outState bundle. Then if the screen is rotated, the onCreate function with savedInstanceState will be called again. The savedInstanceState will be checked if it is null or not before restoring all the values mapped in the bundle and updating the score and the state of the buttons.

7.3. Logging

```
2025-02-08 00:10:54.041 30150-30150 LIFECYCLE           com.example.assigment1           I  onCreate
2025-02-08 00:10:54.376 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: false
2025-02-08 00:10:54.437 30150-30150 LIFECYCLE           com.example.assigment1           I  onStart
2025-02-08 00:10:54.442 30150-30150 LIFECYCLE           com.example.assigment1           I  onResume
2025-02-08 00:11:03.036 30150-30150 TRACKING            com.example.assigment1           I  Climb, Score: 1, Hold: 1
2025-02-08 00:11:03.043 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: true
2025-02-08 00:11:04.158 30150-30150 TRACKING            com.example.assigment1           I  Climb, Score: 2, Hold: 2
2025-02-08 00:11:04.160 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: true
2025-02-08 00:11:09.862 30150-30150 LIFECYCLE           com.example.assigment1           I  onPause
2025-02-08 00:11:09.866 30150-30150 LIFECYCLE           com.example.assigment1           I  onStop
2025-02-08 00:11:09.872 30150-30150 LIFECYCLE           com.example.assigment1           I  saveInstanceState: score - 2, hold - 2, hasFal
2025-02-08 00:11:09.875 30150-30150 LIFECYCLE           com.example.assigment1           I  onDestroy
2025-02-08 00:11:09.965 30150-30150 LIFECYCLE           com.example.assigment1           I  onCreate
2025-02-08 00:11:10.191 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: true
2025-02-08 00:11:10.191 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: true
2025-02-08 00:11:10.204 30150-30150 LIFECYCLE           com.example.assigment1           I  onStart
2025-02-08 00:11:10.210 30150-30150 LIFECYCLE           com.example.assigment1           I  onResume
2025-02-08 00:11:14.087 30150-30150 TRACKING            com.example.assigment1           I  Climb, Score: 3, Hold: 3
2025-02-08 00:11:14.090 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: true
2025-02-08 00:11:16.139 30150-30150 TRACKING            com.example.assigment1           I  Fall, Score: 0, Hold: 3
2025-02-08 00:11:16.153 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: false, Fall button: false
2025-02-08 00:11:17.833 30150-30150 TRACKING            com.example.assigment1           I  Reset, Score: 0, Hold: 0
2025-02-08 00:11:17.839 30150-30150 BUTTONS             com.example.assigment1           I  Climb button: true, Fall button: false
```

As written above, in this assignment, I mainly used the log for three purposes: the lifecycle of the app; the action, the score and the hold of the climber; the state of the climb and fall buttons. To achieve this, I used the Log.i() statement with a tag and a message in many functions in the MainActivity.kt file. And the image above is the logging result of this climbing app.

8. Debugging/Testing

MainActivity.kt | MainActivityTest.kt | layout\activity_main.xml | land\activity_main.xml

```
27  class MainActivityTest {                                                    ⚠ 7
29      @Rule
30      @JvmField
31      var mActivityScenarioRule = ActivityScenarioRule(MainActivity::class.java)
32
33      @Test
34      fun mainActivityTest() {
35          onView(allOf(withId(R.id.climb_button))).perform(click())
36
37          onView(allOf(withId(R.id.score))).check(matches(withText( text: "1")))
38
39          onView(allOf(withId(R.id.fall_button), withText( text: "Fall"))).check(matches(isEnabled()))
40      }
41  }
42
```

Run | MainActivityTest

Status: 1 passed | 1 tests, 59 s 171 ms

Filter tests:

Tests	Duration	realme RMX3690
Test Results	3 s	1/1
MainActivityTest	3 s	1/1
mainActivityTest	3 s	✓

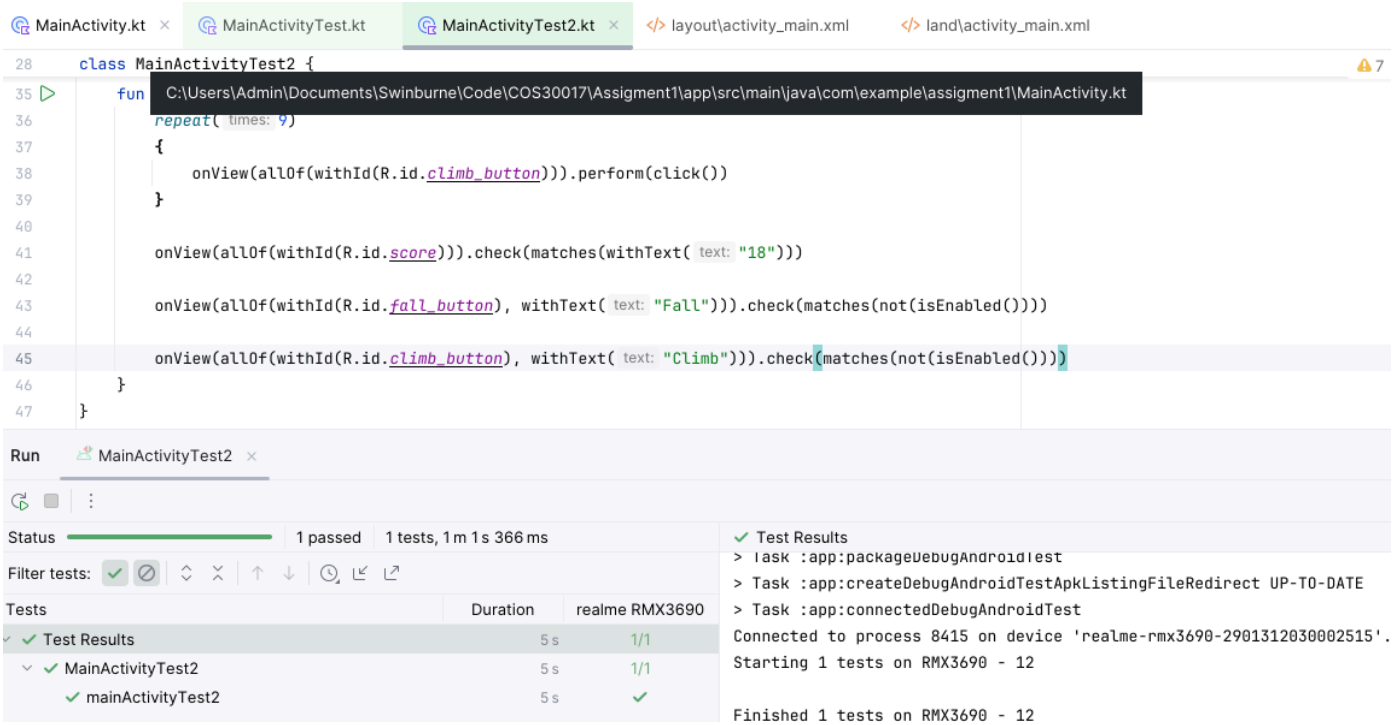
Test Results

```
Connected to process 2168 on device 'realme-rmx3690-2901312030002515'.
Starting 1 tests on RMX3690 - 12

Finished 1 tests on RMX3690 - 12

BUILD SUCCESSFUL in 58s
60 actionable tasks: 6 executed, 54 up-to-date
```

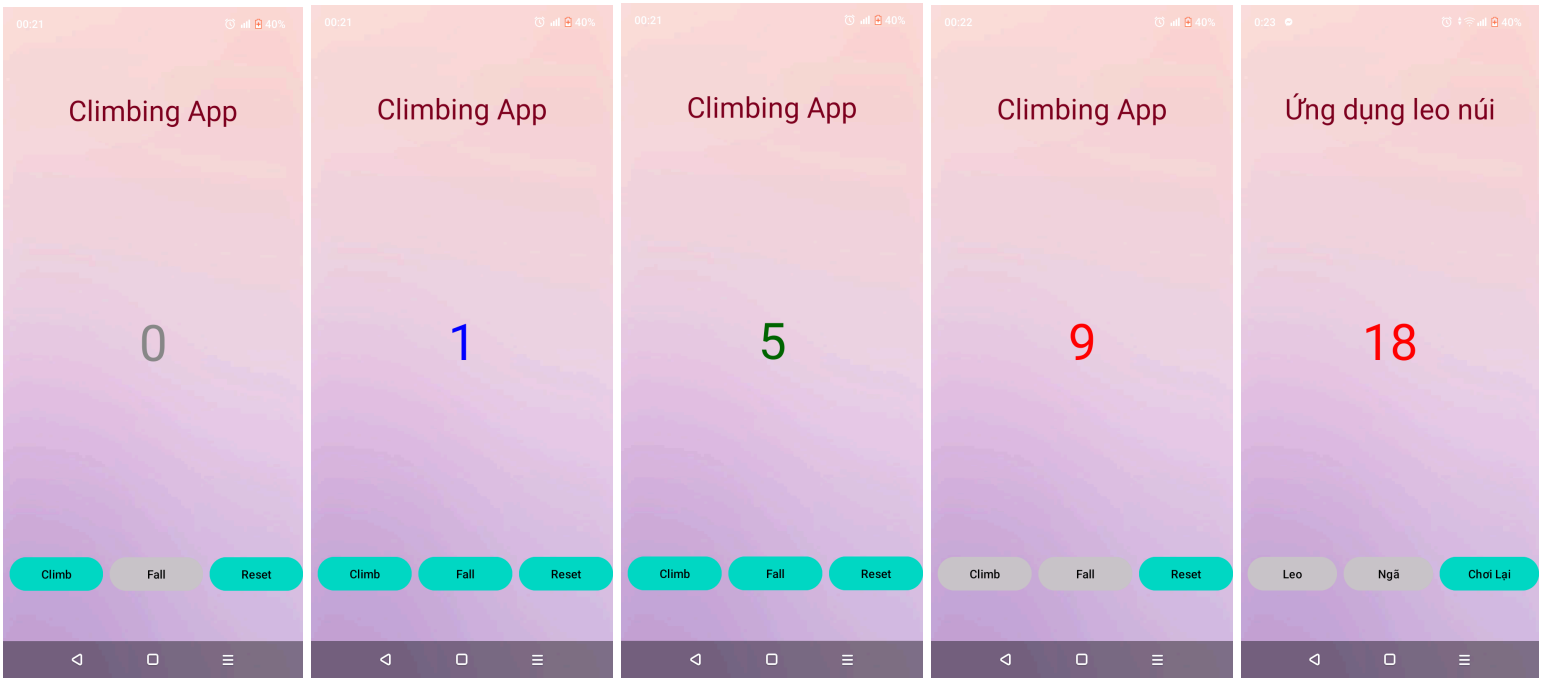
I used 2 Espresso tests to check if the app is working properly. The first test, mainActivityTest(), is conducted to click the climb button once after the app starts, then check if the score TextView is 1 and the fall button is enabled.



The second Espresso test, MainActivityTest2, is for the scenario in which the climb button has been clicked 9 times. Next, it checks if the score TextView is 18, and both the climb and fall buttons are disabled.

9. Process and Demo

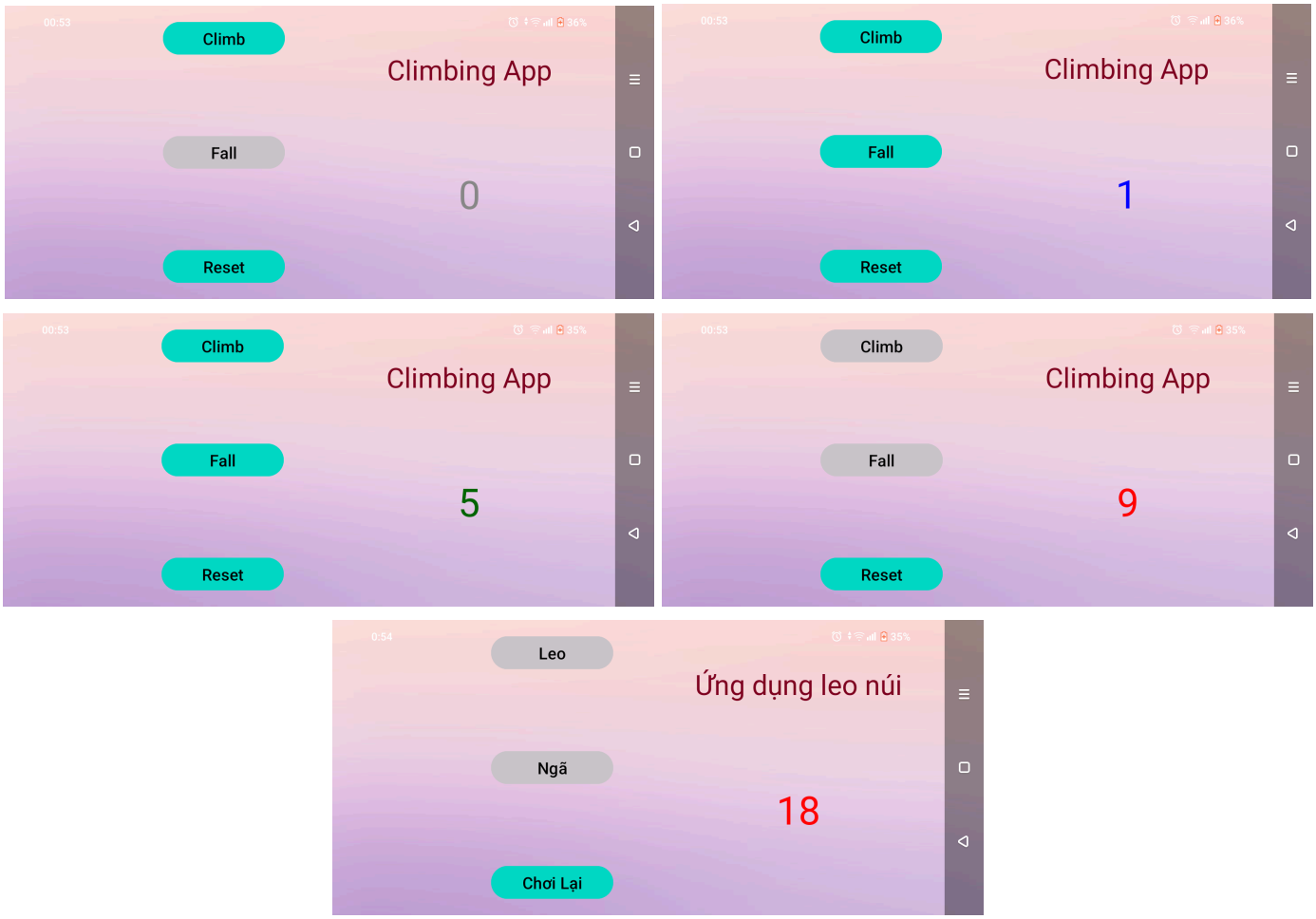
9.1. Portrait Mode



The device I used to test the climbing app is a Realme C30S phone with Android 12 OS and a 720x1600px screen. From left to right, the first screenshot is the initial activity of the app when it just starts, with the fall button disabled as the climber hasn't reached the first hold. The second one is for when the climb button is clicked once, so the fall button is enabled; while the third screenshot is for when the climber has reached the fourth hold, so the score is switched to green. The fourth screenshot occurs when the climber reaches the seventh hold (the climb button is clicked four times) and then he/she falls (the fall button is clicked), reducing the score from 12 to 9 and disabling the fall and the climb buttons. The final screenshot is when the climber has climbed all nine holds, so the climb and fall buttons can't be

clicked anymore. In addition, the final screenshot is captured when the system language is Vietnamese, demonstrating the duolingual support feature.

9.2. Landscape Mode



All the five screenshots in this section occur in the scenarios similar to the 8.1 section, except it is in landscape mode.

10. Reflection

I feel like I have done very well in this assignment, as I have fulfilled all the requirements listed in the Canvas. I believed the code in MainActivity.kt worked quite great and was easier to write when I divided the functionalities of the app into different functions like climbAction, fallAction, updateButtons. In the future, as mentioned earlier, I want to use Relative Layout or Framerate Layout to use a mountain-climbing-related picture as the background for the app, because those two layout types allows me to use the scaleType attribute of the ImageView to automatically crop and scale the image. Furthermore, the current Kotlin code can also be shortened with more advanced functions and features in Kotlin. I also want to include a progress bar in the app to track what hold the climber is on.

11. Github Repository Link

https://github.com/2025-HX01-COS30017-HCM/assignment-1-JJWilson-75