Task ##P/C – Spike: Climb Scorer Mobile App

Goals:

This spike task engaged the design of an interactive Android mobile app that models the scoring system from a climbing wall. The app should record the climber's score, calculate the climber's score, and display the score based on its current hold position while also reflecting the basic Android app development principle of UI design, state management, orientation, and localization.

Learning objectives:

- Develop a single-activity Android application with user interactions and events in Kotlin.
- Create scoring logic that adds up total points across three zones for climbing.
- Manage rotation for portrait/landscape using the onSaveInstanceState() and onRestoreInstanceState() methods.
- Localize the app through English and Vietnamese strings.xml files.
- Log and debug the app workflow with Logcat.
- Provide user feedback via color feedback, toast messages, and responsive layouts.

Tools and Resources Used

- Android Studio (Electric Eel or newer)
- Kotlin programming language
- XML Layouts for UI design
- ConstraintLayout for portrait mode
- LinearLayout for landscape mode
- Logcat for runtime debugging
- Toast for user feedback
- GitHub Classroom for version control and submission
- APA 7 Referencing for report compliance

Knowledge Gaps and Solutions

Gap 1: Managing Score and Hold Logic Across Zones

Problem:
Initially, it was challenging to manage the score system that varied depending on which zone (blue, green, or red) the climber was in.

Solution:
Implemented conditional logic using Kotlin’s when statement:

```
when (hold) {
    in 1 ≤ .. ≤ 3 -> score += 1
    in 4 ≤ .. ≤ 6 -> score += 2
    in 7 ≤ .. ≤ 9 -> score += 3
}
```

```
when (hold) {
    in 1 ≤ .. ≤ 6 -> score -= 3
}
```

-> These cleanly separates the scoring logic per zone and ensures scalability. The score is capped at 18 using conditional checks, preventing negative or overflow values.

Outcome:
Reliable, consistent scoring regardless of gameplay order.

Gap 2: Handling Orientation Changes (Rotation)

Problem:
During rotation (portrait and landscape), all variables such as score and hold reset to zero because Android recreates the activity.

Solution:
Implemented onSaveInstanceState() and onRestoreInstanceState():

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    outState.putInt("score", score)
    outState.putInt("hold", hold)
    outState.putBoolean("hasFallen", hasFallen)
    outState.putBoolean("reachedTop", reachedTop)
}

override fun onRestoreInstanceState(savedInstanceState: Bundle) {
    super.onRestoreInstanceState(savedInstanceState)
    score = savedInstanceState.getInt("score")
    hold = savedInstanceState.getInt("hold")
    hasFallen = savedInstanceState.getBoolean("hasFallen")
    reachedTop = savedInstanceState.getBoolean("reachedTop")
    updateUI()
}
```

This ensures game data persists across orientation changes, satisfying ULO2 and ULO3 requirements.

Outcome:
Smooth transition between portrait and landscape layouts without losing progress.

Gap 3: Implementing Localisation

Problem:
The app initially only supported English.

Solution:
Added a Vietnamese translation file (res/values-vi/strings.xml) to enable bilingual usability.

This is the default (English) file:

```
<resources>
    <string name="app_name">Assignment1</string>
    <string name="score_text">Score: 0</string>
    <string name="currenthold">You are currently at hold: 0</string>
    <string name="climb">Climb</string>
    <string name="fall">Fall</string>
    <string name="reset">Reset</string>
</resources>
```

This is the Vietnamese file:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Assignment1</string>
    <string name="score_text">Điểm: 0</string>
    <string name="currenthold">Bạn hiện đang ở hold: 0</string>
    <string name="climb">Leo</string>
    <string name="fall">Ngã</string>
    <string name="reset">Đặt lại</string>
</resources>
```

Outcome:
The app automatically displays Vietnamese when the device's language is set to Vietnamese.

Gap 4: Visual Feedback and UI Design

Problem:
The original UI provided minimal feedback and lacked color transitions or visual cues.

Solution:
Added color-coded feedback and toast messages:

- Blue zone (holds 1–3): holo_blue_bright
- Green zone (holds 4–6): holo_green_light
- Red zone (holds 7–9): holo_red_light
- Toast messages confirm user actions (Climb, Fall, Reset).

Code:

```
private fun updateUI() {
    scoreDisplay.text = "Score: $score"

    val colorScore = when (hold) {
        in 1 ≤ .. ≤ 3 -> android.R.color.holo_blue_bright
        in 4 ≤ .. ≤ 6 -> android.R.color.holo_green_light
        in 7 ≤ .. ≤ 9 -> android.R.color.holo_red_light
        else -> android.R.color.black
    }

    scoreDisplay.setTextColor(ContextCompat.getColor(this, colorScore))
    currentHold.setTextColor(ContextCompat.getColor(this, colorScore))
}
```

Outcome:
Improved clarity and user engagement.

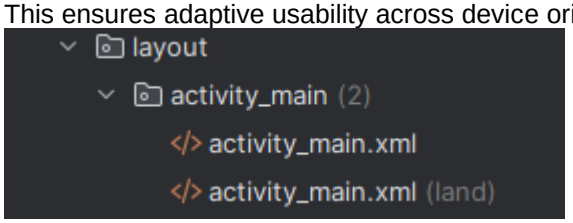
Gap 5: Landscape Layout Implementation

Problem:
Portrait and landscape layouts initially used the same XML design, violating the assignment's layout diversity requirement.

Solution:
Created two layout files:

- res/layout/activity_main.xml -> ConstraintLayout (portrait)
- res/layout-land/activity_main.xml -> LinearLayout (landscape)

This ensures adaptive usability across device orientations.



Open Issues and Recommendations

Issue	Description	Recommendation
Limited graphics	The app currently uses only text elements.	Add vector icons or images representing climbing holds for realism.
Scoring animation	Score updates appear abruptly.	Add a simple fade or scale animation for smoother visual feedback.
Testing	Only tested on Android Emulator.	Test on various physical devices for consistent layout performance.

```
<?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:id="@+id/main"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/scoreDisplay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/score_text"
        android:textSize="32sp"
        android:textStyle="bold"
        android:padding="16dp"
        android:textColor="@android:color/black"/>

    <TextView
        android:id="@+id/currentHold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/currenthold"
        android:textSize="32sp"
        android:padding="16dp"/>

    <Button
        android:id="@+id/btnClimb"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/climb"
        android:layout_margin="8dp"/>

    <Button
        android:id="@+id/btnFall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="@string/fall"
        android:layout_margin="8dp"/>

    <Button
        android:id="@+id/btnReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/reset"
        android:layout_margin="8dp"/>

</LinearLayout>
```

File - D:\Android Projects\Assignment1\app\src\main\res\layout-land\activity_main.xml

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <androidx.constraintlayout.widget.ConstraintLayout
 3     xmlns:android="http://schemas.android.com/apk/res
   /android"
 4     xmlns:app="http://schemas.android.com/apk/res-
   auto"
 5     xmlns:tools="http://schemas.android.com/tools"
 6     android:layout_width="match_parent"
 7     android:layout_height="match_parent"
 8     android:padding="16dp"
 9     tools:context=".MainActivity">
10
11     <TextView
12         android:id="@+id/scoreDisplay"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="@string/score_text"
16         android:textColor="@android:color/black"
17         android:textSize="32sp"
18         android:textStyle="bold"
19         app:layout_constraintTop_toTopOf="parent"
20         app:layout_constraintStart_toStartOf="parent"
21         app:layout_constraintEnd_toEndOf="parent"
22         app:layout_constraintBottom_toTopOf="@+id/
   btnClimb" />
23
24
25     <TextView
26         android:id="@+id/currentHold"
27         android:layout_width="wrap_content"
28         android:layout_height="wrap_content"
29         android:layout_marginTop="80dp"
30         android:textSize="32sp"
31         android:text="@string/currenthold"
32         app:layout_constraintEnd_toEndOf="parent"
33         app:layout_constraintHorizontal_bias="0.499"
34         app:layout_constraintStart_toStartOf="parent"
35         app:layout_constraintTop_toTopOf="parent" />
36
37     <Button
38         android:id="@+id/btnClimb"
```

```
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:layout_marginTop="96dp"
42          android:text="@string/climb"
43          app:layout_constraintEnd_toEndOf="parent"
44          app:layout_constraintHorizontal_bias="0.0"
45          app:layout_constraintStart_toStartOf="parent"
46          app:layout_constraintTop_toBottomOf="@id/
    scoreDisplay" />
47
48      <Button
49          android:id="@+id/btnFall"
50          android:layout_width="wrap_content"
51          android:layout_height="wrap_content"
52          android:layout_marginTop="112dp"
53          android:layout_marginEnd="380dp"
54          android:text="@string/fall"
55          app:layout_constraintEnd_toEndOf="parent"
56          app:layout_constraintTop_toBottomOf="@id/
    scoreDisplay" />
57
58      <Button
59          android:id="@+id/btnReset"
60          android:layout_width="wrap_content"
61          android:layout_height="wrap_content"
62          android:layout_marginTop="96dp"
63          android:layout_marginEnd="16dp"
64          android:text="@string/reset"
65          app:layout_constraintEnd_toEndOf="parent"
66          app:layout_constraintTop_toBottomOf="@+id/
    scoreDisplay" />
67
68
69  </androidx.constraintlayout.widget.ConstraintLayout>
70
```

```kotlin
package com.example.assignment1

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import android.widget.TextView
import android.widget.Button
import android.widget.Toast
import androidx.core.content.ContextCompat
import android.util.Log

class MainActivity : AppCompatActivity() {

    private lateinit var scoreDisplay: TextView
    private lateinit var currentHold: TextView
    private lateinit var btnClimb: Button
    private lateinit var btnFall: Button
    private lateinit var btnReset: Button

    private var score = 0
    private var hold = 0
    private var hasFallen = false
    private var reachedTop = false

    override fun onCreate(savedInstanceState: Bundle
?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        scoreDisplay = findViewById(R.id.scoreDisplay
)
        currentHold = findViewById(R.id.currentHold)
        btnClimb = findViewById(R.id.btnClimb)
        btnFall = findViewById(R.id.btnFall)
        btnReset = findViewById(R.id.btnReset)

        btnClimb.setOnClickListener {
            if (hasFallen || reachedTop) {
```

```
40                  return@setOnClickListener
41              }
42
43              hold++
44              currentHold.text = "You are currently at
   hold: $hold"
45
46              when (hold) {
47                  in 1..3 -> score += 1
48                  in 4..6 -> score += 2
49                  in 7..9 -> score += 3
50              }
51
52              if (hold >= 9) {
53                  reachedTop = true
54                  Log.i("Game", "Reached the top!")
55                  Toast.makeText(this, "You reached to
   the top!!", Toast.LENGTH_SHORT).show()
56              }
57
58              if (score >= 18) {
59                  score = 18
60              }
61
62              updateUI()
63          }
64
65          btnFall.setOnClickListener {
66              if (reachedTop || hasFallen || hold == 0
   ) {
67                  return@setOnClickListener
68              }
69
70              when (hold) {
71                  in 1..6 -> score -= 3
72              }
73
74              hold -= 3
75              currentHold.text = "You are currently at
   hold: $hold"
76
```

```
 77              if (score <= 0) {
 78                  score = 0
 79              }
 80
 81              hasFallen = true
 82              Log.i("Game", "Player has fallen!")
 83              Toast.makeText(this, "Player has fallen
    ...", Toast.LENGTH_SHORT).show()
 84
 85              updateUI()
 86          }
 87
 88          btnReset.setOnClickListener {
 89              score = 0
 90              hold = 0
 91              reachedTop = false
 92              hasFallen = false
 93              Log.i("Game", "Reset!")
 94              currentHold.text = "You are currently at
     hold: $hold"
 95              Toast.makeText(this, "Game Reset!",
    Toast.LENGTH_SHORT).show()
 96              updateUI()
 97          }
 98
 99          updateUI()
100      }
101
102      override fun onSaveInstanceState(outState:
    Bundle) {
103          super.onSaveInstanceState(outState)
104          outState.putInt("score", score)
105          outState.putInt("hold", hold)
106          outState.putBoolean("hasFallen", hasFallen)
107          outState.putBoolean("reachedTop", reachedTop
    )
108      }
109
110      override fun onRestoreInstanceState(
    savedInstanceState: Bundle) {
111          super.onRestoreInstanceState(
```

```
111 savedInstanceState)
112         score = savedInstanceState.getInt("score")
113         hold = savedInstanceState.getInt("hold")
114         hasFallen = savedInstanceState.getBoolean("
    hasFallen")
115         reachedTop = savedInstanceState.getBoolean("
    reachedTop")
116         updateUI()
117     }
118
119     private fun updateUI() {
120         scoreDisplay.text = "Score: $score"
121
122         val colorScore = when (hold) {
123             in 1..3 -> android.R.color.
    holo_blue_bright
124             in 4..6 -> android.R.color.
    holo_green_light
125             in 7..9 -> android.R.color.
    holo_red_light
126             else -> android.R.color.black
127         }
128
129         scoreDisplay.setTextColor(ContextCompat.
    getColor(this, colorScore))
130         currentHold.setTextColor(ContextCompat.
    getColor(this, colorScore))
131     }
132 }
133
```

```xml
<resources>
    <string name="app_name">Assignment1</string>
    <string name="score_text">Score: 0</string>
    <string name="currenthold">You are currently at hold: 0</string>
    <string name="climb">Climb</string>
    <string name="fall">Fall</string>
    <string name="reset">Reset</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Assignment1</string>
    <string name="score_text">Điểm: 0</string>
    <string name="currenthold">Bạn hiện đang ở hold: 0</string>
    <string name="climb">Leo</string>
    <string name="fall">Ngã</string>
    <string name="reset">Đặt lại</string>
</resources>
```