# Climbing Mountain App Report

The GitHub Link: https://github.com/nminh2209/Mobileappp/tree/main/Assignment%201

## Abstract

The "Climbing Mountain App" is an Android application that simulates a climbing experience by tracking user progress, managing falls, and supporting multilingual functionality. This report provides a detailed analysis of the application's features, technical implementation, UI/UX considerations, and potential improvements. Additionally, it aligns the project with key Unit Learning Outcomes (ULOs) to demonstrate its educational and technical relevance.

## Introduction

The development of mobile applications presents unique challenges compared to traditional personal computing or web-based environments. Factors such as screen size, memory constraints, and processor limitations significantly influence design choices. The "Climbing Mountain App" is designed to optimize user interaction while considering these hardware-imposed restrictions. This report examines the application's design, implementation, and alignment with relevant Unit Learning Outcomes.

## Features

### Screenshots of The Program:

A screenshot of a phone

Description automatically generated

English Version

A screenshot of a phone

Description automatically generated

Japanese Version

A screenshot of a phone

Description automatically generated

Vietnamese Version

### Climbing Mechanism

* Users progress incrementally up to a maximum of nine holds.
* Score increments are based on hold levels:
  + Holds 1-3: +1 point per hold (Blue zone)
  + Holds 4-6: +2 points per hold (Green zone)
  + Holds 7-9: +3 points per hold (Red zone)
* Score display color changes dynamically to indicate the zone.

### Falling Mechanism

* If a user falls before reaching hold 9:
  + 3 points are deducted.
  + The fall is recorded, preventing further climbing.
* Falling is not an option upon reaching hold 9.

### Reset Functionality

* Resets climbing progress and score to zero.
* Allows users to restart the climb.

### Localization Support

* Supports English, Japanese, and Vietnamese.
* Language selection updates the UI dynamically.

## Technical Implementation

### The Whole Program

package com.example.climbingmoutainapp  
  
import android.content.res.Configuration  
  
import android.graphics.Color  
import android.os.Bundle  
import android.util.Log  
import android.view.View  
import android.widget.Button  
import android.widget.TextView  
import androidx.appcompat.app.AppCompatActivity  
import java.util.Locale  
  
  
class MainActivity : AppCompatActivity() {  
 private var currentScore = 0  
 private var currentHold = 0  
 private var hasFallen = false  
 private lateinit var scoreText: TextView  
 private lateinit var climbButton: Button  
 private lateinit var fallButton: Button  
 private lateinit var resetButton: Button  
  
 companion object {  
 private const val STATE\_SCORE = "currentScore"  
 private const val STATE\_HOLD = "currentHold"  
 private const val STATE\_FALLEN = "hasFallen"  
 private const val TAG = "ClimbingScoreApp"  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
  
 val langEnglish = findViewById<Button>(R.id.*langEnglish*)  
 val langJapanese = findViewById<Button>(R.id.*langJapanese*)  
 val langVietnamese = findViewById<Button>(R.id.*langVietnamese*)  
  
 langEnglish.setOnClickListener **{** v: View? **->** setLocale("en") **}** langJapanese.setOnClickListener **{** v: View? **->** setLocale("ja") **}** langVietnamese.setOnClickListener **{** v: View? **->** setLocale("vi") **}** initializeViews()  
 restoreState(savedInstanceState)  
 setupButtons()  
 updateScoreDisplay()  
  
 Log.d(TAG, "App initialized with score: $currentScore")  
 }  
  
 private fun setLocale(languageCode: String) {  
 val locale = Locale(languageCode)  
 Locale.setDefault(locale)  
 val resources = *resources* val config: Configuration = resources.*configuration* config.setLocale(locale)  
 resources.updateConfiguration(config, resources.*displayMetrics*)  
 recreate()  
 }  
  
  
 private fun initializeViews() {  
 scoreText = findViewById(R.id.*scoreText*)  
 climbButton = findViewById(R.id.*climbButton*)  
 fallButton = findViewById(R.id.*fallButton*)  
 resetButton = findViewById(R.id.*resetButton*)  
 }  
  
 private fun setupButtons() {  
 climbButton.setOnClickListener **{** if (!hasFallen && currentHold < 9) {  
 currentHold++  
 currentScore += when (currentHold) {  
 in 1..3 -> 1 // Blue zone  
 in 4..6 -> 2 // Green zone  
 in 7..9 -> 3 // Red zone  
 else -> 0  
 }  
 Log.d(TAG, "Climbed to hold $currentHold, score: $currentScore")  
 updateScoreDisplay()  
 }  
 **}** fallButton.setOnClickListener **{** if (currentHold > 0 && currentHold < 9 && !hasFallen) {  
 currentScore = *maxOf*(0, currentScore - 3)  
 hasFallen = true  
 Log.d(TAG, "Fall recorded, new score: $currentScore")  
 updateScoreDisplay()  
 }  
 **}** resetButton.setOnClickListener **{** currentScore = 0  
 currentHold = 0  
 hasFallen = false  
 Log.d(TAG, "Score reset")  
 updateScoreDisplay()  
 **}** }  
  
 private fun updateScoreDisplay() {  
 scoreText.*text* = currentScore.toString()  
 scoreText.setTextColor(when (currentHold) {  
 in 1..3 -> Color.*BLUE* in 4..6 -> Color.*GREEN* in 7..9 -> Color.*RED* else -> Color.*BLACK* })  
 }  
  
 override fun onSaveInstanceState(outState: Bundle) {  
 super.onSaveInstanceState(outState)  
 outState.putInt(STATE\_SCORE, currentScore)  
 outState.putInt(STATE\_HOLD, currentHold)  
 outState.putBoolean(STATE\_FALLEN, hasFallen)  
 Log.d(TAG, "State saved")  
 }  
  
 private fun restoreState(savedInstanceState: Bundle?) {  
 savedInstanceState?.*let* **{** currentScore = **it**.getInt(STATE\_SCORE)  
 currentHold = **it**.getInt(STATE\_HOLD)  
 hasFallen = **it**.getBoolean(STATE\_FALLEN)  
 Log.d(TAG, "State restored")  
 **}** }  
}

### MainActivity Class

class MainActivity : AppCompatActivity() {  
 private var currentScore = 0  
 private var currentHold = 0  
 private var hasFallen = false  
 private lateinit var scoreText: TextView  
 private lateinit var climbButton: Button  
 private lateinit var fallButton: Button  
 private lateinit var resetButton: Button

The MainActivity class extends AppCompatActivity and implements core functionalities, including UI interactions and state management.

### Instance Variables

* currentScore: Tracks the user's score.
* currentHold: Tracks the hold level.
* hasFallen: Boolean flag to prevent additional actions after a fall.
* scoreText, climbButton, fallButton, resetButton: UI components.

### Lifecycle Methods

* onCreate(Bundle?)

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
  
 val langEnglish = findViewById<Button>(R.id.*langEnglish*)  
 val langJapanese = findViewById<Button>(R.id.*langJapanese*)  
 val langVietnamese = findViewById<Button>(R.id.*langVietnamese*)  
  
 langEnglish.setOnClickListener **{** v: View? **->** setLocale("en") **}** langJapanese.setOnClickListener **{** v: View? **->** setLocale("ja") **}** langVietnamese.setOnClickListener **{** v: View? **->** setLocale("vi") **}** initializeViews()  
 restoreState(savedInstanceState)  
 setupButtons()  
 updateScoreDisplay()  
  
 Log.d(TAG, "App initialized with score: $currentScore")  
}

* + Initializes UI elements.
  + Restores previous state if available.
  + Configures button click listeners.
* onSaveInstanceState(Bundle)

override fun onSaveInstanceState(outState: Bundle) {  
 super.onSaveInstanceState(outState)  
 outState.putInt(STATE\_SCORE, currentScore)  
 outState.putInt(STATE\_HOLD, currentHold)  
 outState.putBoolean(STATE\_FALLEN, hasFallen)  
 Log.d(TAG, "State saved")  
}

* + Saves the current score, hold level, and fall status to persist through configuration changes.

### Event Handlers

* setLocale(String): Dynamically updates the app language.
* initializeViews(): Initializes UI components.
* setupButtons(): Defines button actions for climbing, falling, and resetting.
* updateScoreDisplay(): Updates UI elements based on user progress.

### Functionalities:

Language Settings

private fun setLocale(languageCode: String) {  
 val locale = Locale(languageCode)  
 Locale.setDefault(locale)  
 val resources = *resources* val config: Configuration = resources.*configuration* config.setLocale(locale)  
 resources.updateConfiguration(config, resources.*displayMetrics*)  
 recreate()  
}

The setLocale function dynamically updates the app’s language by changing the locale settings and refreshing the UI.

Setup Buttons:

private fun setupButtons() {  
 climbButton.setOnClickListener **{** if (!hasFallen && currentHold < 9) {  
 currentHold++  
 currentScore += when (currentHold) {  
 in 1..3 -> 1 // Blue zone  
 in 4..6 -> 2 // Green zone  
 in 7..9 -> 3 // Red zone  
 else -> 0  
 }  
 Log.d(TAG, "Climbed to hold $currentHold, score: $currentScore")  
 updateScoreDisplay()  
 }  
 **}** fallButton.setOnClickListener **{** if (currentHold > 0 && currentHold < 9 && !hasFallen) {  
 currentScore = *maxOf*(0, currentScore - 3)  
 hasFallen = true  
 Log.d(TAG, "Fall recorded, new score: $currentScore")  
 updateScoreDisplay()  
 }  
 **}** resetButton.setOnClickListener **{** currentScore = 0  
 currentHold = 0  
 hasFallen = false  
 Log.d(TAG, "Score reset")  
 updateScoreDisplay()  
 **}**}

* Climb Button: Moves the player up, increases the score based on zones, and updates the UI.
* Fall Button: Deducts points, prevents further climbing, and updates the UI
* Reset Button: Resets the game, allowing a fresh start.

Handling:

* private fun updateScoreDisplay() {  
   scoreText.*text* = currentScore.toString()  
   scoreText.setTextColor(when (currentHold) {  
   in 1..3 -> Color.*BLUE* in 4..6 -> Color.*GREEN* in 7..9 -> Color.*RED* else -> Color.*BLACK* })  
  }  
    
  override fun onSaveInstanceState(outState: Bundle) {  
   super.onSaveInstanceState(outState)  
   outState.putInt(STATE\_SCORE, currentScore)  
   outState.putInt(STATE\_HOLD, currentHold)  
   outState.putBoolean(STATE\_FALLEN, hasFallen)  
   Log.d(TAG, "State saved")  
  }  
    
  private fun restoreState(savedInstanceState: Bundle?) {  
   savedInstanceState?.*let* **{** currentScore = **it**.getInt(STATE\_SCORE)  
   currentHold = **it**.getInt(STATE\_HOLD)  
   hasFallen = **it**.getBoolean(STATE\_FALLEN)  
   Log.d(TAG, "State restored")  
   **}**}
* The updateScoreDisplay, onSaveInstanceState, and restoreState functions serve key roles in managing the app's UI and ensuring that game progress is not lost during activity lifecycle changes.
* updateScoreDisplay() updates the UI dynamically with the current score and zone-based color changes.
* onSaveInstanceState() stores the game state in a Bundle before the activity is paused or destroyed.
* restoreState() retrieves the stored data and reinstates the previous game state.

## Unit Testing

package com.example.climbingmoutainapp  
  
import android.widget.TextView  
import android.widget.Button  
import org.junit.Test  
import org.junit.runner.RunWith  
import org.robolectric.Robolectric  
import org.robolectric.RobolectricTestRunner  
import org.robolectric.annotation.Config  
import org.junit.Assert.\*  
  
@RunWith(RobolectricTestRunner::class)  
@Config(sdk = [33])  
class MainActivityTest {  
 @Test  
 fun testScoring() {  
 val activity = Robolectric.buildActivity(MainActivity::class.*java*)  
 .create()  
 .resume()  
 .get()  
  
 val scoreText = activity.findViewById<TextView>(R.id.*scoreText*)  
 val climbButton = activity.findViewById<Button>(R.id.*climbButton*)  
 val fallButton = activity.findViewById<Button>(R.id.*fallButton*)  
  
 // Log initial state  
 *println*("Initial score: ${scoreText.*text*}")  
 assertNotNull("Score TextView should not be null", scoreText)  
 assertNotNull("Climb Button should not be null", climbButton)  
 assertNotNull("Fall Button should not be null", fallButton)  
  
 // Test initial state  
 assertEquals("Initial score should be 0", "0", scoreText.*text*.toString())  
  
 // Test climbing  
 climbButton.performClick()  
 *println*("Score after climb: ${scoreText.*text*}")  
 assertEquals("Score should be 1 after climbing", "1", scoreText.*text*.toString())  
  
 // Test falling  
 fallButton.performClick()  
 *println*("Score after fall: ${scoreText.*text*}")  
 assertEquals("Score should be 0 after falling", "0", scoreText.*text*.toString())  
 }  
}

The application includes a unit test to ensure the proper functionality of the climbing and falling mechanisms.

## Debugging and Logging

* Utilizes Log.d(TAG, message) to log key events such as climbing progress, falls, and resets.
* Enhances debugging efficiency by tracking user interactions.

## Layout File:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center"  
 android:padding="16dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:layout\_marginBottom="16dp">  
  
 <Button  
 android:id="@+id/langEnglish"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="EN"  
 android:layout\_margin="4dp"/>  
  
 <Button  
 android:id="@+id/langJapanese"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="JP"  
 android:layout\_margin="4dp"/>  
  
 <Button  
 android:id="@+id/langVietnamese"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="VN"  
 android:layout\_margin="4dp"/>  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/scoreText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="48sp"  
 android:text="0"  
 android:layout\_marginBottom="32dp"/>  
  
 <Button  
 android:id="@+id/climbButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/climb"/>  
  
 <Button  
 android:id="@+id/fallButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/fall"/>  
  
 <Button  
 android:id="@+id/resetButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/reset"/>  
</LinearLayout>

## UI/UX Considerations

* Color-coded scoring zones for intuitive visual feedback.
* Interactive buttons enhance user engagement.
* Language selection improves accessibility.
* Score persistence across device rotations.

## Potential Improvements

1. **Persistent Storage**: Use SharedPreferences or a database to save user progress across app restarts.
2. **Animation Effects**: Implement smooth UI animations for climbing and falling actions.
3. **Leaderboard System**: Introduce a scoring system with global or local leaderboards.
4. **Sound Effects**: Add auditory feedback for climbing, falling, and button clicks.
5. **Expanded Language Support**: Broaden localization to additional languages.

## Conclusion

The Climbing Mountain App effectively demonstrates core principles of mobile application development by considering hardware limitations, user interaction, and multilingual accessibility. The project aligns with mobile computing best practices and supports Unit Learning Outcomes through its design and implementation. Future enhancements, such as persistent storage, animations, and leaderboards, can further improve user engagement and functionality.

## References

Mobile SDK: [*https://developer.salesforce.com/docs/platform/mobile-sdk/overview*](https://developer.salesforce.com/docs/platform/mobile-sdk/overview)

Calling activity's methods from a separate class: [*https://teamtreehouse.com/community/calling-activitys-methods-from-a-separate-class*](https://teamtreehouse.com/community/calling-activitys-methods-from-a-separate-class)

For UI: <https://xuanthulab.net/co-ban-ve-cach-tao-va-su-dung-activity-trong-android.html>

From Android Developers: <https://developer.android.com/?hl=vi>