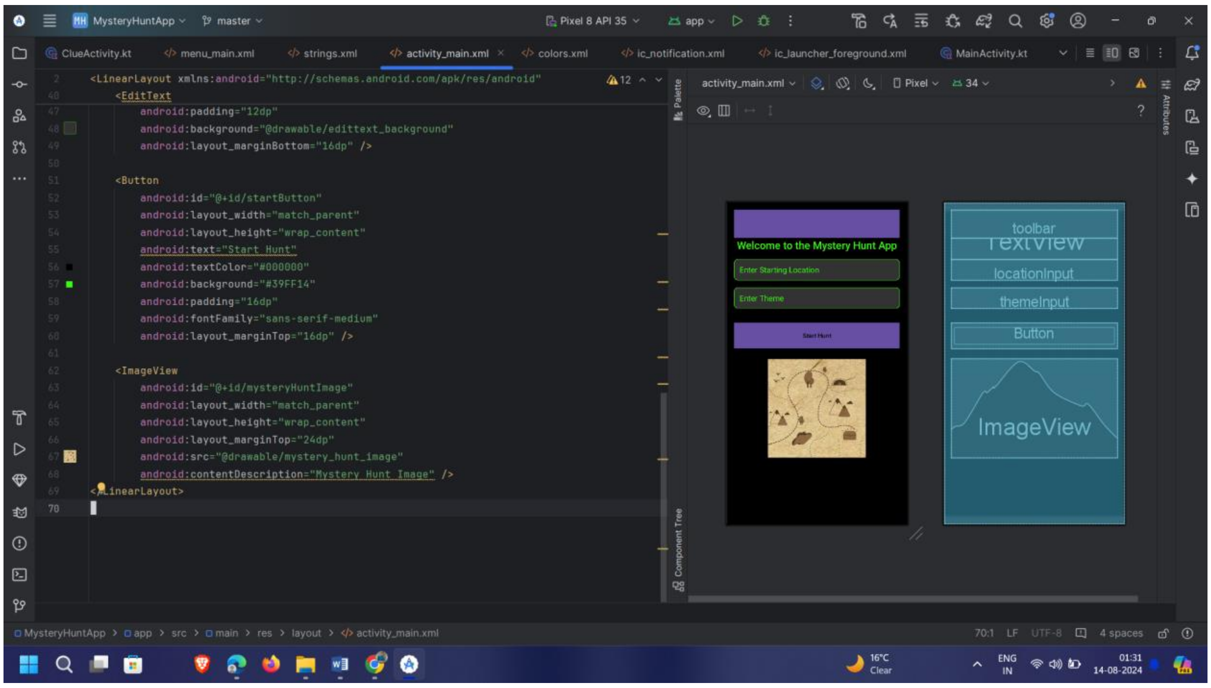
Mobile Programming Assignment 3

StudentId:1229429 Name: Harsh Soni

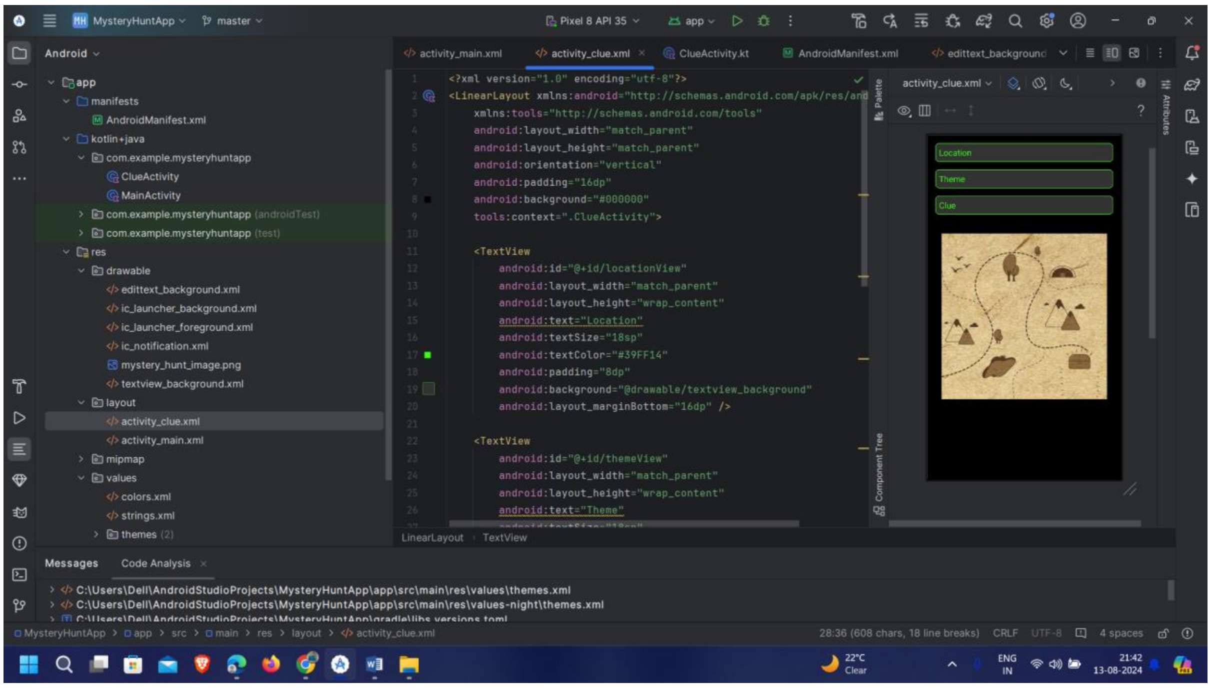
Github Link: https://github.com/harsh74780/MysteryHuntApp.git



```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    <EditText
        android:padding="12dp"
        android:background="@drawable/edittext_background"
        android:layout_marginBottom="16dp" />

    <Button
        android:id="@+id/startButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Hunt"
        android:textColor="#000000"
        android:background="#39FF14"
        android:padding="16dp"
        android:fontFamily="sans-serif-medium"
        android:layout_marginTop="16dp" />

    <ImageView
        android:id="@+id/mysteryHuntImage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:src="@drawable/mystery_hunt_image"
        android:contentDescription="Mystery Hunt Image" />
</LinearLayout>
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="#000000"
    tools:context=".ClueActivity">

    <TextView
        android:id="@+id/locationView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Location"
        android:textSize="18sp"
        android:textColor="#39FF14"
        android:padding="8dp"
        android:background="@drawable/textview_background"
        android:layout_marginBottom="16dp" />

    <TextView
        android:id="@+id/themeView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Theme"
```

```
> C:\Users\Dell\AndroidStudioProjects\MysteryHuntApp\app\src\main\res\values\themes.xml
> C:\Users\Dell\AndroidStudioProjects\MysteryHuntApp\app\src\main\res\values-night\themes.xml
> C:\Users\Dell\AndroidStudioProjects\MysteryHuntApp\gradle\libs.versions.toml
```

MainActivity.kt

```
package com.example.mysteryhuntapp

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.Button
import android.widget.EditText
import android.util.Log

class MainActivity : AppCompatActivity() {

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

        val locationInput: EditText = findViewById(R.id.locationInput)
        val themeInput: EditText = findViewById(R.id.themeInput)
        val startButton: Button = findViewById(R.id.startButton)

        startButton.setOnClickListener {
            val location = locationInput.text.toString()
            val theme = themeInput.text.toString()

            Log.d("MainActivity", "Location: $location, Theme: $theme")

            val intent = Intent(this, ClueActivity::class.java).apply {
                putExtra("location", location)
                putExtra("theme", theme)
            }
            startActivity(intent)
        }
    }
}
```

ClueActivity.kt

```
package com.example.mysteryhuntapp

import android.Manifest
import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.content.pm.PackageManager
import android.os.Build
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import androidx.core.content.ContextCompat
import android.widget.TextView
import android.util.Log

class ClueActivity : AppCompatActivity() {
    private val channelId = "mysteryHuntChannel"
    private val requestCodePostNotifications = 1

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

        createNotificationChannel()

        val location = intent.getStringExtra("location")
        val theme = intent.getStringExtra("theme")

        Log.d("ClueActivity", "Received Location: $location, Theme:
$theme")

        val locationView: TextView = findViewById(R.id.locationView)
        val themeView: TextView = findViewById(R.id.themeView)
        val clueView: TextView = findViewById(R.id.clueView)

        locationView.text = location
        themeView.text = theme
        clueView.text = generateClue(location, theme)

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            if (ContextCompat.checkSelfPermission(this,
Manifest.permission.POST_NOTIFICATIONS) !=
PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.POST_NOTIFICATIONS),
requestCodePostNotifications)
            } else {
                sendNotification(clueView.text.toString())
            }
        } else {
            sendNotification(clueView.text.toString())
        }
    }

    private fun generateClue(location: String?, theme: String?): String {
        // Placeholder for generating clues based on location and theme
        return "Find the ancient tree in $location that fits the $theme
theme."
    }

    private fun sendNotification(clue: String) {
        val builder = NotificationCompat.Builder(this, channelId)
            .setSmallIcon(R.drawable.ic_notification)
            .setContentTitle("New Clue Available")
            .setContentText(clue)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setVibrate(longArrayOf(1000, 1000, 1000, 1000, 1000))
            .setLights(0xFF0000FF.toInt(), 3000, 3000)
            .setAutoCancel(true)

        try {
            with(NotificationManagerCompat.from(this)) {
                notify(0, builder.build())
            }
        } catch (e: SecurityException) {
            Log.e("ClueActivity", "Notification permission not granted", e)
        }
    }


    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val name = "Mystery Hunt Channel"
            val descriptionText = "Channel for Mystery Hunt notifications"
            val importance = NotificationManager.IMPORTANCE_HIGH
            val channel = NotificationChannel(channelId, name,
importance).apply {
                description = descriptionText
            }
            val notificationManager: NotificationManager =
                getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
            notificationManager.createNotificationChannel(channel)
        }
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == requestCodePostNotifications) {
            if ((grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                val clueView: TextView = findViewById(R.id.clueView)
                sendNotification(clueView.text.toString())
            } else {
                // Permission denied, show a message to the user
                Log.d("ClueActivity", "Notification permission denied")
            }
        }
    }
}
```