AMRITA VISHWA VIDYAPEETHAM

19CSE213 – Project Report

Android Logger

By,

Jai Chiranjeeva – 21122

Manasa S – 21138

Jahnavi – 21146

ABSTRACT

Logem , an android application built using flutter & kotlin which is useful to read the system logs by a developer or an user with ease which have a functionality to clear and also save logs. This application is built using fluuter widgets which can be obtained by dart language to display the current logs on the screen to the user.

Users can also interact with the application as the can even pause printing the logs to check any suspicious activity happening in you OS , this is achieved with the help of various android packages which gives us the logs , kotlin packages which are used to connect to android packages to get logs efficiently. Overall, the application provides an intuitive and interactive approach to get system process information and data with time stamps and severity.

INTRODUCTION

This application deals with the core android modules which may have severe consequences if we don’t deal with them properly , like the permissions to the logs . logs may contain sensitive information which may be vulnerable to hackers seeking for this information from your device . Flutter is a cross-platform supported but brining kotlin to grab system logs of particularly android which is a native code makes this application functionalities limited to android only.

In this application , there where various parts of the code which collectively builds the application . As this is a flutter application , we create widgets which represents text , images , floating button , appbar. Which will be having functionalities of clearing logs ,displaying them , pausing them for a while and also saving them to a file in your internal storage to share with others.

This project in flutter seeks make it easy to the user to read the logs rather than he connecting his device to a pc and then grabbing a log through some adb commands which is a complex process for a naïve user to perform when he is facing some difficulties with the OS he is running on.

How this works?

As we have to get the logs from OS , we need methods from kotlin as well as flutter to display.

Methods defined and used:

Flutter :

“_setlog()”: This method gets the log data from kotlin code

“getlog()”: Based on the pasue button behaviour , this will know whether to set a log now or not

“printlogs()” : Display logs to the user

“initstate()” : A magic method of dart which is invoked art the start of application

Kotlin :

“svelogs(log: String)” : this method gets the present log string and saves the data to a file for user.

“getLogs()” : Executes the androidOS command with its libraries and fetches the log data to the source.

CODE:

Main.dart:

```
import 'dart:async';

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';


void main() {

  runApp(MyApp());

}


String ss = "", sb = "", p = "";

String res = "";

String xyz = "file";

String plog = '';

bool pause = false;


class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  MyHomePageState createState() => MyHomePageState();
}

class MyHomePageState extends State<MyHomePage> {
  static const platform = MethodChannel('com.example.logem/grab');

  int i = 1;
  Future<void> _setlog() async {
    if (true) {
      p = res;
      res = "${await platform.invokeMethod('getLatestLog')}\n";

      i += 1;
    if (p != res) {
      ss = res + ss;
      if (i > 55) {
        ss = sb;
        i = 1;
      } else if (i > 45) {
        sb += ss;
      }
      xyz = xyz;
      setState(() {
        plog = ss;
      });
    }
  }
}

Future<void> _printlogs() async {
  const interval = Duration(milliseconds: 200);
  Timer.periodic(interval, (Timer t) async {
    if (!pause) {
      _setlog();
    }
  });
```

```
}

Future<void> _pa() async {
  setState(() {
    pause = !pause;
  });
}

@override
void initState() {
  _printlogs();
  super.initState();
}

@override
Widget build(BuildContext context) {
  return Container(
    height: double.infinity,
    width: 100,
    child: Scaffold(
      appBar: AppBar(
        title: const Text('Logem'),
        actions: <Widget>[
    // action button
    IconButton(
      icon: const Icon(Icons.cleaning_services_rounded),
      onPressed: () {
        ss = "";
      },
    ),
    // action button
    IconButton(
      icon: const Icon(Icons.save),
      onPressed: () async {
        String dmp =
            await platform.invokeMethod('svelogs', {"log": plog});
      },
    ),
  ],
),
body: SingleChildScrollView(
  child: Align(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Text(
                    plog,
                    style: TextStyle(
                        color: Colors.black,
                        decoration: TextDecoration.none,
                        fontSize: 15.0),
                  ),
                ],
              ),
            ),
          ),
          floatingActionButton: FloatingActionButton(
            onPressed: _pa,
            backgroundColor: Colors.blue,
            child: const Icon(Icons.pause_circle_outline_outlined),
          ),
        ),
      );
    }
  }
```

Pubspec.yaml:

```
name: logem

description: A new Flutter project.

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish_to: 'none' # Remove this line if you wish to publish to pub.dev


# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as
versionCode.

# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is
used as CFBundleVersion.

# Read more about iOS versioning at

#
https://developer.apple.com/library/archive/documentation/General/Reference/I
nfoPlistKeyReference/Articles/CoreFoundationKeys.html
```

```
# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1


environment:

  sdk: '>=2.19.6 <3.0.0'


# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below
to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter



  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino_icons: ^1.0.2
  flutter_logs: ^2.1.10

  logger: ^1.0.0

  kotlin_flavor: ^0.2.0

  permission_handler: ^10.2.0

  #android_logger: ^0.0.2


dev_dependencies:

  flutter_test:

    sdk: flutter


  # The "flutter_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter_lints: ^2.0.0


# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec


# The following section is specific to Flutter packages.
```

```
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true
```

<u>MainActivity.kt:</u>

```
package com.example.logem

import io.flutter.embedding.android.FlutterActivity



import android.content.Intent
import android.util.Log
import android.content.IntentFilter
import android.os.Build.VERSION
import android.os.Build.VERSION_CODES
import android.icu.text.SimpleDateFormat
import androidx.annotation.NonNull

import io.flutter.embedding.engine.FlutterEngine

import io.flutter.plugin.common.MethodChannel

import java.io.File

import java.io.BufferedReader

import java.io.InputStreamReader

import java.util.Stack

//import java.util.Date




class MainActivity : FlutterActivity() {

    private val CHANNEL = "com.example.logem/grab"

    override fun configureFlutterEngine(@NonNull flutterEngine:
FlutterEngine) {

        super.configureFlutterEngine(flutterEngine)

        MethodChannel(flutterEngine.dartExecutor.binaryMessenger,
CHANNEL).setMethodCallHandler {

            // Note: this method is invoked on the main thread.

            call, result ->
        if(call.method == "getLatestLog") {
          val logg = getLogs()
          result.success(logg)
         }
         else if(call.method == "svelogs") {
          val str=call.argument<String>("log");
          val ahs=svelogs(str.toString())
          result.success(ahs)
         }
         else{
          result.notImplemented()
        }


    }


}
```

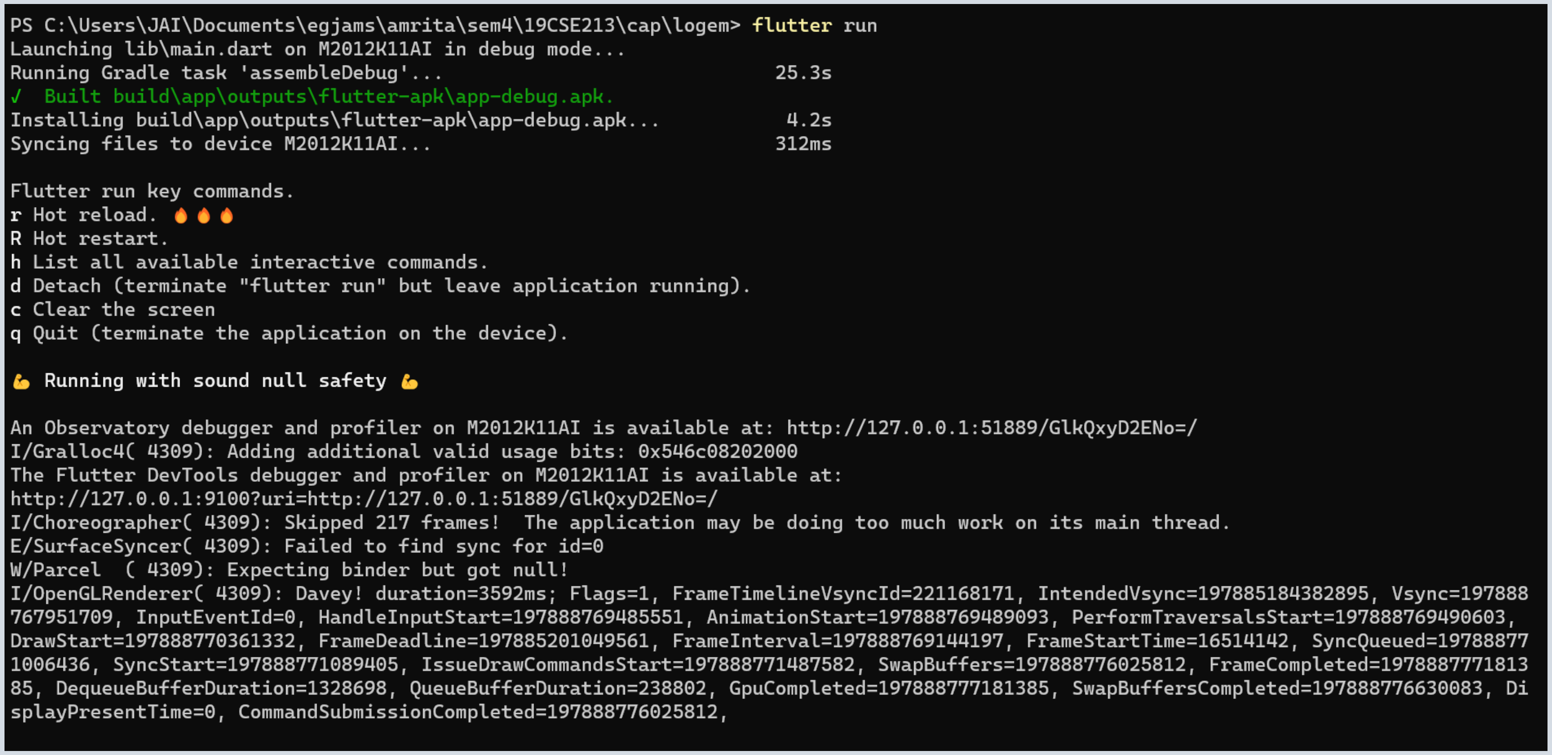
```
private fun svelogs(slog:String): String{
    try{


        val file = File("/storage/emulated/0/"+"logs.txt")
        if(!file.exists())
        file.createNewFile()


        file.writeText(slog)
    }
    catch(e: Exception){


    }
    return "";
}




    private fun getLogs(): String {


        try {
            var process = Runtime.getRuntime().exec("logcat -t 1")

            var bufferedReader = process.inputStream.bufferedReader()

            var grab =  Stack<String>();

            bufferedReader.useLines { lines ->

                lines.forEach {


                    grab.push(it)


                }


                    return grab.peek()+"\n"

            }

        } catch (e: Exception) {

            return "ERROR \n \n \n \n \n \n \n \n "

        }


    }


}
```

AndroidManifest.xml:

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.logem"
    xmlns:tools="http://schemas.android.com/tools">


    <uses-permission android:name="android.permission.READ_LOGS"
        tools:ignore="ProtectedPermissions" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.FOREGROUND_SERVICE" />




   <application
        android:label="logem"
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden|keyboard|screen
Size|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout
|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize"
            android:requestLegacyExternalStorage="true">
            <!-- Specifies an Android theme to apply to this Activity as soon
as
                 the Android process has started. This theme is visible to the
user
                 while the Flutter UI initializes. After that, this theme continues
                 to determine the Window background behind the Flutter UI. --
>
            <meta-data
              android:name="io.flutter.embedding.android.NormalTheme"
              android:resource="@style/NormalTheme"
```

```
             />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

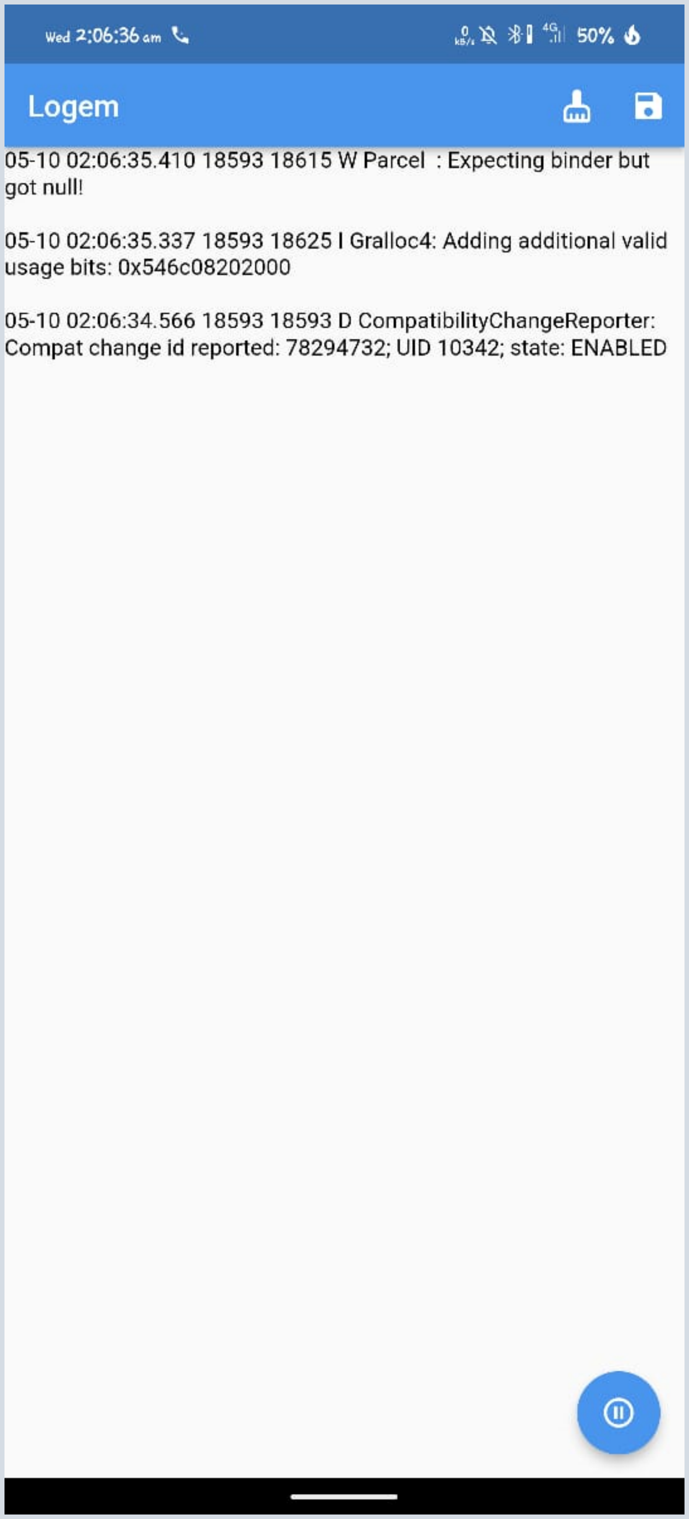
Compilation :



```
PS C:\Users\JAI\Documents\egjams\amrita\sem4\19CSE213\cap\logem> flutter run
Launching lib\main.dart on M2012K11AI in debug mode...
Running Gradle task 'assembleDebug'...                             25.3s
√  Built build\app\outputs\flutter-apk\app-debug.apk.
Installing build\app\outputs\flutter-apk\app-debug.apk...           4.2s
Syncing files to device M2012K11AI...                              312ms

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

💪 Running with sound null safety 💪

An Observatory debugger and profiler on M2012K11AI is available at: http://127.0.0.1:51889/GlkQxyD2ENo=/
I/Gralloc4( 4309): Adding additional valid usage bits: 0x546c08202000
The Flutter DevTools debugger and profiler on M2012K11AI is available at:
http://127.0.0.1:9100?uri=http://127.0.0.1:51889/GlkQxyD2ENo=/
I/Choreographer( 4309): Skipped 217 frames!  The application may be doing too much work on its main thread.
E/SurfaceSyncer( 4309): Failed to find sync for id=0
W/Parcel  ( 4309): Expecting binder but got null!
I/OpenGLRenderer( 4309): Davey! duration=3592ms; Flags=1, FrameTimelineVsyncId=221168171, IntendedVsync=197885184382895, Vsync=197888
767951709, InputEventId=0, HandleInputStart=197888769485551, AnimationStart=197888769489093, PerformTraversalsStart=197888769490603,
DrawStart=197888770361332, FrameDeadline=197885201049561, FrameInterval=197888769144197, FrameStartTime=16514142, SyncQueued=19788877
1006436, SyncStart=197888771089405, IssueDrawCommandsStart=197888771487582, SwapBuffers=197888776025812, FrameCompleted=1978887771813
85, DequeueBufferDuration=1328698, QueueBufferDuration=238802, GpuCompleted=197888777181385, SwapBuffersCompleted=197888776630083, Di
splayPresentTime=0, CommandSubmissionCompleted=197888776025812,
```

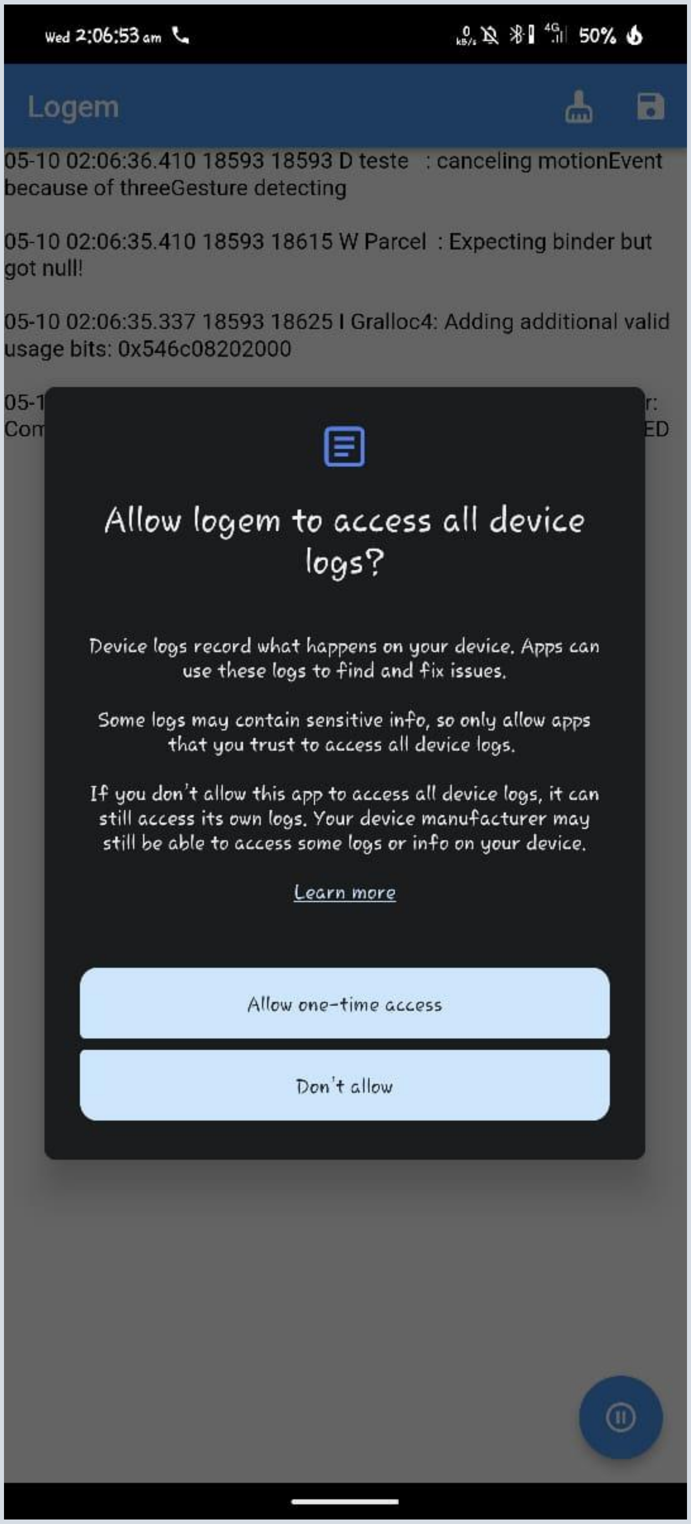
Interfaces :

→ Home :



```
05-10 02:06:35.410 18593 18615 W Parcel  : Expecting binder but got null!

05-10 02:06:35.337 18593 18625 I Gralloc4: Adding additional valid usage bits: 0x546c08202000

05-10 02:06:34.566 18593 18593 D CompatibilityChangeReporter: Compat change id reported: 78294732; UID 10342; state: ENABLED
```

➔ Permissions :



```
05-10 02:06:36.410 18593 18593 D teste   : canceling motionEvent
because of threeGesture detecting

05-10 02:06:35.410 18593 18615 W Parcel  : Expecting binder but
got null!

05-10 02:06:35.337 18593 18625 I Gralloc4: Adding additional valid
usage bits: 0x546c08202000

05-1                                                              r:
Com                                                               ED
```

Allow logem to access all device logs?

Device logs record what happens on your device. Apps can use these logs to find and fix issues.

Some logs may contain sensitive info, so only allow apps that you trust to access all device logs.

If you don't allow this app to access all device logs, it can still access its own logs. Your device manufacturer may still be able to access some logs or info on your device.

And also , we working videos added in ppt.

CONCLUSION

In conclusion, this application in Flutter provides an effective way for getting the logcat information of your system through an interacting and simple user interface.

The project utilizes depends on many libraries for its gui , functionalities and the base point of interacting with the OS to get data from it with some system calls efficiently, and its use of Flutter's material library ensures a responsive interface for the user and user-friendly experience for both novice and advanced users.

Through the development and evaluation of the application, we have demonstrated that a graphical interface can provide a more intuitive and accessible approach to the user particularly who is a novice user.

The project's design and implementation have highlighted the importance of developing user-friendly tools that provide a clear and concise visualization of complex systems, such as operating system logcat information.

THANK YOU